Neural Operators of Backstepping Controller and Observer Gain Functions for Reaction-Diffusion PDEs

Miroslav Krstic, Luke Bhan, Yuanyuan Shi

University of California, San Diego, USA

Abstract

Unlike ODEs, whose models involve system matrices and whose controllers involve vector or matrix gains, PDE models involve functions in those roles—functional coefficients, dependent on the spatial variables, and gain functions dependent on space as well. The designs of gains for controllers and observers for PDEs, such as PDE backstepping, are mappings of system model functions into gain functions. These infinite-dimensional nonlinear operators are given in an implicit form through PDEs, in spatial variables, which need to be solved to determine the gain function for each new functional coefficient of the PDE. The need for solving such PDEs can be eliminated by learning and approximating the said design mapping in the form of a neural operator. Learning the neural operator requires a sufficient number of prior solutions for the design PDEs, offline, as well as the training of the operator. In recent work, we developed the neural operators for PDE backstepping designs for first-order hyperbolic PDEs. Here we extend this framework to the more complex class of parabolic PDEs. The key theoretical question is whether the controllers are still stabilizing, and whether the observers are still convergent, if they employ the approximate functional gains generated by the neural operator. We provide affirmative answers to these questions, namely, we prove stability in closed loop under gains produced by neural operators. We illustrate the theoretical results with numerical tests and publish our code on github. The neural operators are three orders of magnitude faster in generating gain functions than PDE solvers for such gain functions. This opens up the opportunity for the use of this neural operator methodology in adaptive control and in gain scheduling control for nonlinear PDEs.

1 Introduction

ML as a tool for learning control methodologies. In the recent manuscript [10] we introduced a learning-based control *framework* which devises a new role for machine learning (ML): learn an entire control design *methodology*, in the form of a mapping from the plant model to the controller gains, or even to the control inputs.

This framework is neither model-free nor methodology-agnostic. On the contrary, it is method-specific. For a particular method (LQR, pole placement, MPC, backstepping, etc.), after a large number of training calculations of the controller gains on a sample set of plant models, an ML-approximated mapping of the method is learned. Once learned as a plant-to-gains mapping, the control design for a *new*/next plant (outside of the training set) does not require another solution of the design equations (Riccati, Bezout, etc.) but merely entails an "evaluation of the learnt map" to obtain the control gains.

One would argue that no dire need exists for LQR or other linear finite-dimensional designs for such a learning-based capability, where an entire methodology is "encoded" into a neural mapping. The cost of the solution of a design problem (say, a Riccati equation, even of a high dimension) is not prohibitive, even online with current technology. Indeed, we are not motivated by design challenges in finite dimensions but for PDEs.

In PDE control, the design problems are not matrix equations. They are PDEs themselves (or harder problems, such as operator Riccati equations). Since the infinite-dimensional state of a PDE is a function of *spatial* variables, the controller gain is also a function of spatial variables. Finding the gain typically entails solving a PDE in space (but not in time). It is therefore of interest, in PDE control, to have a capability where producing the control gain functions is just an evaluation of a neural mapping that has already learned the design methodology on a large set of previously offline-solved control design problems for a sample set of PDEs in a certain class.

Neural operators for approximating mappings of functions into functions. Just as the control designs for linear finite-dimensional systems are matrix-to-matrix mappings (A, B into gain K), control designs for PDEs are function-to-function mappings (spatially-dependent coefficients into

Email addresses: krstic@ucsd.edu (Miroslav Krstic), lbhan@ucsd.edu (Luke Bhan), yyshi@eng.ucsd.edu (Yuanyuan Shi).

gains). Our inspiration for encoding PDE control methodologies into machine learning comes from recent advances in the mathematics of machine learning. Motivated by the tasks of finding solution/flow maps (from the initial conditions into future states) for physical PDEs (such as the difficult Navier-Stokes PDEs), research teams led by George Karniadakis [55,56], Anima Anandkumar and Andrew Stuart [53,54], and George Pappas and Manfred Morari [41,73], have developed neural approximation methods, termed "neural operators," with provable properties for nonlinear operators acting on functions and producing functions. These approaches are not simply discretizing PDEs and finding solution maps to the resulting large ODE solution problems. In the language of distributed parameter systems, they are not "early lumping" methods of learning solution maps. They approximate (non-discretized) function-to-function nonlinear operators and provide guarantees of the accuracy of approximation in terms of the required sizes of the training sets and neural networks.

The value of such a capability in PDE control cannot be understated. With a theoretically rigorous and numerically powerful capability like this, specific PDE control methods, for specific classes of PDEs, can be learned once and encoded as neural operators, ready to produce the control gain functions for any new functional coefficients of the same classes of PDEs.

In a theoretically rigorous field like PDE control, a computational capability with rigorous approximation guarantees has a value primarily if it allows the retention of the theoretical properties proven for the "exact design". This is indeed what we show in the paper [10] in which we introduce the framework: approximate neural operator representations of a particular PDE control method—PDE backstepping—preserves its stability guarantees in spite of the control gains not being generated by solving the design PDEs but by the gains being generated from the learned "neural model" of PDE backstepping.

Extension of PDE backstepping neural operators from hyperbolic [10] to parabolic PDEs. Hyperbolic PDEs involve only the first derivatives in space and time. This makes them (all else being equal) the "simplest" PDE class for control. Delay systems combine ODEs with delays—the simplest form of a PDE. While the simplest among PDEs, hyperbolic PDEs are not necessarily easy to control. They can be unstable, with many unstable eigenvalues, and only one input acting at the boundary of a domain. This mix of simplicity within the PDE family, with the non-triviality for control, makes hyperbolic PDEs the ideal entry point for any new study in PDE control, including the introduction of a new framework for learning-based control in our [10]. The framework is depicted in Figure 1. The *learning* and *implementation* portions of the framework in Figure 1 are depicted in Figure 2.

Control design problems for hyperbolic PDEs are hyperbolic

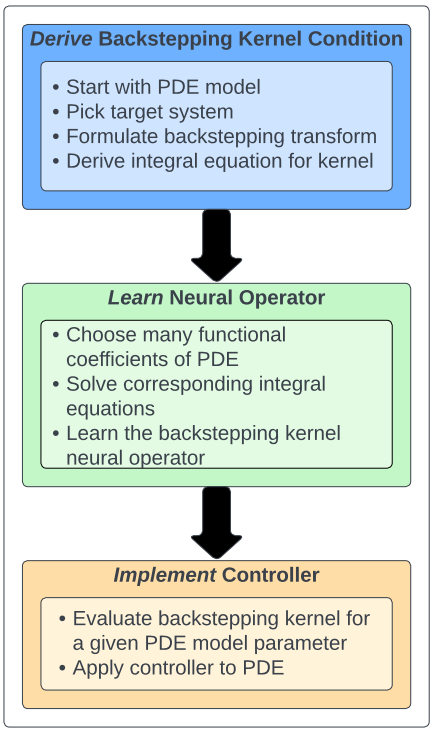

Fig. 1. An algorithmic representation of our design paradigm of employing neural operators in boundary control of PDEs. Three major step clusters are performed: (1) derivation of the integral equations for the backstepping kernels, performed only once; (2) learning of the mapping from the plant parameter functions into the backstepping kernel functions, also performed only once; and (3) implementation of the controller for specific plant parameters. The task in the top box has been completed in [46, 77]. In this paper, the task in the middle box is introduced and stability guarantees for the task in the bottom box are provided.

PDEs themselves, namely, equations with only first derivatives in multiple spatial variables. Parabolic PDEs, with their first derivative in time but second derivatives in space, are the natural next challenge for learning the PDE backstepping methodology using neural operators. This is what we undertake in this paper. The chief difficulty with learning backstepping kernel operators for parabolic PDEs is that the kernels are governed by second-order PDEs, which raises the difficulty for solving such PDEs and for proving the sufficient smoothness of their solutions so that the neural operator (NO) approximations have guarantee of sufficient accuracy for preserving stabilization. At the intuitive level, with more derivatives, and more boundary conditions, the nonlinear operator from the reaction function to the gain in parabolic PDEs is a more complex operator than the nonlinear operator from the recirculation function to the gain in hyperbolic PDEs. There are hyperbolic cases where the kernel mapping can be written (though not solved) using the Laplace transform. That is never the case with parabolic PDEs; the design problem is never of spatial dimension lower than two.

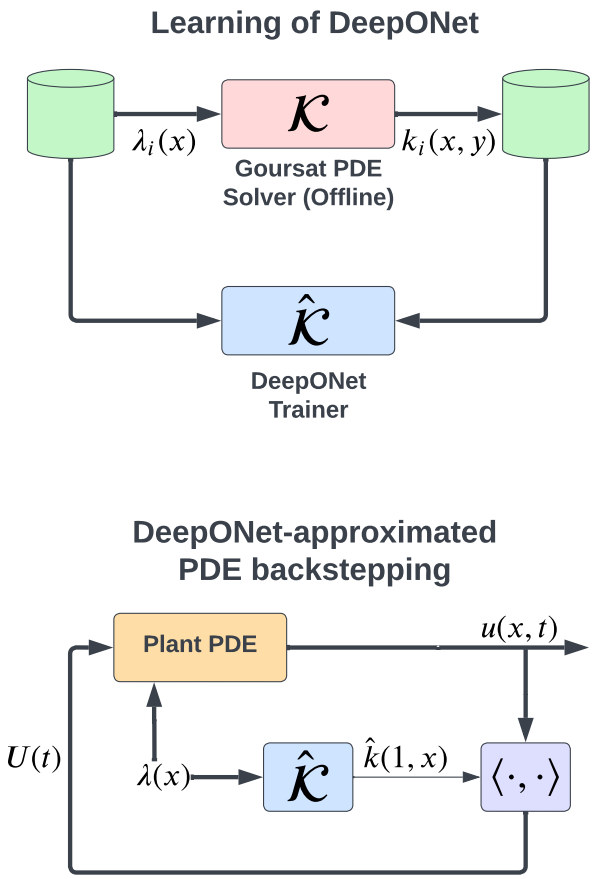

Fig. 2. Stages (2) and (3) of the framework in Figure 1. TOP: The process of learning the PDE backstepping design operator $\mathscr{K}:\lambda\mapsto k$ involves many solutions of a kernel PDE $k_{xx}-k_{yy}=\lambda k$ in the Goursat form, for different functions $\lambda_i(x)$ and then training of a neural operator $\hat{\mathscr{K}}:\lambda\mapsto\hat{k}$. BOTTOM: Feedback implementation of PDE backstepping control with gain kernel $\hat{k}(1,x)$ generated by the DeepONet $\hat{\mathscr{K}}$.

We consider parabolic PDE systems of the form

$$u_t(x,t)=u_{xx}(x,t)+\lambda(x)u(x,t), \qquad x\in[0,1) \tag{1}$$
$$u(0,t)=0 \tag{2}$$
$$u(1,t)=U(t). \tag{3}$$

Our goal is the design of a PDE backstepping boundary control

$$U(t)=\int_0^1 k(1,y)u(y,t)dy, \tag{4}$$

as well as an observer with the (collocated) boundary sensing of $u_x(1,t)$. By "design" we mean to find the gain function k in the control law (4), namely, to find the output k of the function-to-function mapping $\mathscr{K}:\lambda\mapsto k$, depicted in Figure 3. This paper's objective is to learn the design operator $\mathscr{K}$ with a neural operator approximation $\hat{\mathscr{K}}$ (top of Figure 2) and to employ the resulting approximate gain $\hat{k}$ in the control law (bottom of Figure 2).

actuation	opposite boundary	sensing	
$u(1,t)=U(t)$	$u(0,t)=0$	$u_x(0,t)$	anti-col
$\boldsymbol{u(1,t)=U(t)}$	$\boldsymbol{u(0,t)=0}$	$\boldsymbol{u_x(1,t)}$	**col**
$u(1,t)=U(t)$	$u_x(0,t)=0$	$u(0,t)$	anti-col
$u(1,t)=U(t)$	$u_x(0,t)=0$	$u(1,t)$	col
$u_x(1,t)=U(t)$	$u(0,t)=0$	$u_x(0,t)$	anti-col
$u_x(1,t)=U(t)$	$u(0,t)=0$	$u_x(1,t)$	col
$u_x(1,t)=U(t)$	$u_x(0,t)=0$	$u(0,t)$	anti-col
$u_x(1,t)=U(t)$	$u_x(0,t)=0$	$u(1,t)$	col

Table 1
Eight possible combinations of boundary actuation, sensing, and boundary condition at the opposite end of $[0,1]$. We focus on the simplest combination—in the second row.

Since parabolic PDEs in one dimension have two boundary conditions, and also boundary actuation and boundary sensing can be employed at either boundary, a total of sixteen combinations of boundary actuation, boundary sensing, and boundary condition on the unactuated boundary are possible. Taking the symmetry between the boundaries $x=0$ and $x=1$ into account, the total number of truly distinct combinations is eight. They are listed in Table 1.

We are able to solve all eight problems but, in this paper, pursue the simplest of the eight combinations for pedagogical reasons. The case with Dirichlet boundary conditions, $u(0,t)=0, u(1,t)=U(t)$ is, notationally, the simplest case. It allows the reader to most quickly grasp the utility and the technical steps in employing neural operators in the control of parabolic PDEs.

All the results in the paper—a full-state controller, an observer, and an output-feedback law (as well as the seven additional combinations not pursued in the paper)—can be extended to the more general class of parabolic PDE systems,

$$\begin{aligned} v_t(x,t)=&\,\varepsilon(x)v_{xx}(x,t)+b(x)v_x(x,t)\\ &+\lambda(x)v(x,t)+g(x)v(0,t)\\ &+\int_0^x f(x,y)v(y,t)dy, \quad x\in(0,L). \end{aligned} \tag{5}$$

The domain $[0,L]$ is easily normalized to $[0,1]$, the diffusion $\varepsilon(x)$ is easily normalized to unity, and the advection term $b(x)v_x(x,t)$ is easily eliminated with a scaling transformation. We forego pursuing this myriad of alternatives for pedagogical reasons—they are overwhelming but standard. Likewise, we forgo the treatment of the terms $g(x)v(0,t)+\int_0^x f(x,y)v(y,t)dy$ because it adds complications but it is standard as well.

The parabolic PDE with unity diffusion and with spatially-varying reaction $\lambda(x)$ is a perfect introduction to the possibilities that neural operators present for PDE backstepping control where the computation of the gain kernel $k(x,y)$ for each new $\lambda(x)$ can be avoided by developing a neural op-

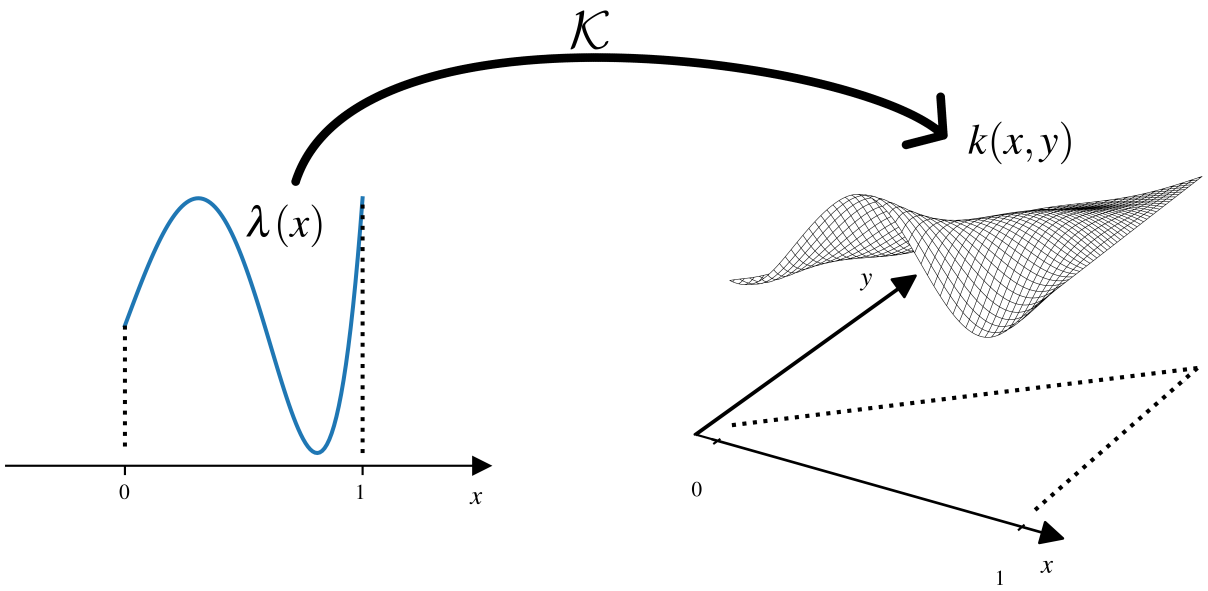

Fig. 3. The PDE backstepping design operator $\mathscr{K} : \lambda \mapsto k$, where $\lambda(x)$ is the spatially-varying reaction coefficient of the PDE, whereas $k(x,y)$ is the kernel function of the backstepping transformation, producing the feedback gain function $k(1,y)$ in the feedback law $U(t) = \int_0^1 k(1,y)u(y,t)dy$.

erator approximation of the functional mapping (nonlinear operator) $\mathscr{K} : \lambda \mapsto k$, which is depicted in Figure 3.

We opt to present in the paper the results for the combination in the second row of Table 1 because this combination allows us to "kill two birds with one stone" in our exposition. For this particular actuator-sensor combination, which is collocated (and the simplest of the four collocated combinations), the same kernel is used to obtain the gain functions for both the controller and the observer. This relieves the reader of the burden of following multiple approximations of the kernel, multiple neural operators, multiple training processes for those operators, and multiple theorems that guarantee the approximability of those multiple operators. The concept of encoding the methodologies of controller, observer, and output-feedback design into a neural operator is grasped through a single operator $\mathscr{K} : \lambda \mapsto k$. And the duplication in the exposition is avoided. A reader who possesses the skills in calculations and the stamina can work out the remaining combinations in Table 1.

PDE Backstepping. Even though PDE backstepping was first developed for parabolic systems [77], it is best to begin its study from the easier, hyperbolic case [46]. Control of hyperbolic PDEs has grown into a rich area because, in the hyperbolic case, one can stabilize a coupled system with fewer inputs than PDEs. A *pair* of coupled hyperbolic PDEs was stabilized with a single boundary input in [18], an extension to $n+1$ hyperbolic PDEs with a single input was introduced in [28], an extension to $n+m$ PDEs with boundary actuation on m "homodirectional" PDEs in [35,36], an extension to cascades with ODEs in [29], and an extension to "sandwiched" ODE-PDE-ODE systems in [94,95]. Redesigns robust to delays are provided in [3]. PDE backstepping-based output-feedback regulation with disturbances is proposed in [26,27].

For parabolic PDEs, backstepping for full-state feedback stabilization was developed in [77] and for observer design in [78]. A complex extension from linear to nonlinear parabolic PDEs, using infinite Volterra series, was provided in [88, 89]. Backstepping was combined with differential flatness in [61]. The first solutions to the null-controllability problem for parabolic PDEs were provided, using backstepping, in [19, 31]. Sampled-data and event-triggered versions of backstepping for parabolic PDEs appeared in [30, 38, 39, 71]. Work on cascades of parabolic PDEs with other systems has included heat-ODE cascades [5,43], delay-parabolic cascades [44], and ODE-heat-ODE sandwich systems [93]. A backstepping design for a moving-boundary PDE-ODE Stefan system was presented in [42]. Coupled parabolic PDEs introduce special challenges and have been tackled in [4,65,90]. Extensions from multiple 1D parabolic PDEs to PDEs in 2D and higher dimensions, such as in the book [60] are arguably even more challenging and have been pursued for Navier-Stokes and magnetohydrodynamic systems in [87, 92] on channel domains, as well as for reaction-diffusion systems on balls of arbitrary dimensions [91]. Adaptive control designs for parabolic PDEs were introduced in [45,79,80], extended in [40], and extended to the hyperbolic case in [8]. For coupled hyperbolic PDEs with unknown parameters, the parabolic designs in [45, 79, 80] inspired a comprehensive collection of adaptive control designs in the book [1]. Applications of backstepping to PDE models of traffic are introduced in [96,97].

Advances in learning-based control. What we present here is one more among many directions in learning-based control. For the benefit of the reader from PDE control, we highlight a few results from this vast and growing literature. Stability of learning-based MPC was established in [2, 72] and followed, for nonlinear systems, by efforts on joint learning of the controller and(or) Lyapunov functions [13–15, 21, 22]. In addition, [64, 83] have explored how learning-based control affects systems with known Lyapunov functions, [12, 23, 68] studied learning of stability certificates and stable controllers from data, and [6] developed a provably stable data-driven algorithm based on system measurements and prior system knowledge.

For reinforcement learning (RL) [9], the focus has been on learning the system dynamics and providing closed-loop guarantees in *finite-time* for both linear [16,24,48] and nonlinear systems [7,37,49,76]. For model-free RL, [32,62,66, 100] proved the convergence of policy optimization to the optimal controller for LTI systems, [63, 67] for LTV systems, [82] for partially observed linear systems. For a review of policy optimization (PO) methods for LQR, H_∞ control, risk-sensitive control, LQG, and output feedback synthesis, see [34]. For nonlinear systems, [17, 20, 75] investigated PO with stability guarantees from CLFs. In addition to PO, [11, 51, 84, 85] proved stability and convergence of actor-critic methods [51, 85] and Q-learning [84]. In CPS, learning-based control was developed for partially observable systems [57].

Learning-based control in games and for MAS is pursued in [33, 58, 59, 69, 70, 86, 98, 99]. Convergence is shown to Nash equilibria in zero-sum linear quadratic games [99], continuous games [59], Stackelberg games [33], Markov games [58, 98], and multi-agent learning over networked systems [69, 70].

We focus on learning-based control for PDE systems. In our previous work [74], we demonstrate the empirical success of using NOs for accelerating PDE backstepping observers. Our recent work [10] represents the first step towards using NOs for *provably* bypassing gain computations and directly learning the controller with closed-loop stabilization guarantee, in hyperbolic PDE systems.

Neural operators—a brief summary. Neural operators are neural network-parameterized maps for learning relationships between function spaces. They consist of three components: an encoder, an approximator, and a reconstructor [50]. The encoder is an interpolation from an infinite-dimensional function space to a finite-dimensional vector representation. The approximator aims to mimic the infinite map using a finite-dimensional representation of both the domain function space and the target function space. The reconstructor then transforms the approximation output into the infinite-dimensional target function space. The implementation of both the approximator and the reconstructor is generally coupled and can take many forms. For example, the original DeepONet [56] contains a "branch" net that represents the approximation network and a "trunk" net that builds a basis for the target function space. The outputs of the two networks are then taken in linear combination with each other to form the operator. FNO [54] utilizes the approximation network in the Fourier domain where the reconstruction is done on a basis of the trigonometric polynomials. LOCA [41] integrates the approximation network and reconstruction step with a unified attention mechanism. NOMAD [73] extends the linear reconstructor map in DeepONet to a nonlinear map that is capable of learning on nonlinear submanifolds in function spaces.

With the basic notions and notation for NOs given in Appendix A, we state next the key technical result that enables our use of NOs to learn the PDE backstepping kernel mappings. The result is quoted in its general/abstract form. It is specialized to the PDE control setting in our Theorem 4.

Theorem 1 (DeepONet universal approximation theorem [25, Theorem 2.1]). *Let $X \subset \mathbb{R}^{d_x}$ and $Y \subset \mathbb{R}^{d_y}$ be compact sets of vectors $x \in X$ and $y \in Y$, respectively. Let $\mathscr{U} : X \to U \subset \mathbb{R}^{d_u}$ and $\mathscr{V} : Y \to V \subset \mathbb{R}^{d_v}$ be sets of continuous functions $u(x)$ and $v(y)$, respectively. Let $\mathscr{U}$ be also compact. Assume the operator $\mathscr{G} : \mathscr{U} \to \mathscr{V}$ is continuous. Then, for all $\varepsilon > 0$, there exist $m^*, p^* \in \mathbb{N}$ such that for each $m \geq m^*$, $p \geq p^*$, there exist $\theta^{(k)}, \vartheta^{(k)}$, neural networks $f^{\mathscr{N}}(\cdot;\theta^{(k)}), g^{\mathscr{N}}(\cdot;\vartheta^{(k)}), k = 1,\ldots,p$, and $x_j \in X, j = 1,\ldots,m$, with corresponding $\mathbf{u}_m = (u(x_1), u(x_2), \cdots, u(x_m))^{\mathrm{T}}$, such that*

$$|\mathscr{G}(u)(y) - \mathscr{G}_{\mathbb{N}}(\mathbf{u}_m)(y)| < \varepsilon \tag{6}$$

for all functions $u \in \mathscr{U}$ and all values $y \in Y$ of $\mathscr{G}(u) \in \mathscr{V}$.

In the sequel, we denote the DeepONet neural operator values $\mathscr{G}_{\mathbb{N}}(\mathbf{u}_m)(y)$ compactly as $\hat{\mathscr{G}}(u)(y)$ and the operators themselves as $\hat{\mathscr{G}}$.

Paper outline and contributions. In Section 2 we recap the basic PDE backstepping approach from [77]. Recalling in Section 3 the twice continuous differentiability of the backstepping kernel function we establish the existence of a neural operator with an arbitrary accuracy for a set of continuously differentiable reaction coefficients not exceeding a certain size in the supremum norm. Sections 4 and 5 contain our main results. In Section 4 we prove the stability of a feedback law employing a DeepONet approximation of the backstepping gain. In Section 5 we prove the convergence of a backstepping observer that employs a DeepONet approximation of the observer gain. In Section 6 we combine the DeepONet-based full-state feedback and observer, to obtain a DeepONet-based output feedback controller with an actuator-sensor pair collocated at the $x = 1$ boundary. In Section 7 we illustrate the theoretical results with numerical tests.

This paper's contribution relative to the inaugural work on backstepping for parabolic PDEs [77] is in providing a methodology for capturing the backstepping design in the form of a neural operator and avoiding the need for the solution of kernel PDEs, after the neural operator is once synthesized. This capability is highly valuable in future work in the adaptive control of parabolic PDEs and gain-scheduling for semilinear parabolic PDEs. In relation to our recent work on neural operator approximated backstepping control of hyperbolic PDEs, this paper extends this methodology, including stability guarantees, to a more difficult class of PDE systems and kernel operators. Additionally, compared to [10] where only full-state feedback is considered, in this paper, we solve problems in observer design and output-feedback control, with a convergence guarantee for the DeepONet-approximated backstepping observer.

2 Basic Backstepping Design for Reaction-Diffusion PDE

We employ the following backstepping transformation,

$$w(x,t) = u(x,t) - \int_0^x k(x,y)u(y,t)dy, \tag{7}$$

to convert (1), (2), (3) into the target system

$$w_t = w_{xx} \tag{8}$$
$$w(0,t) = 0 \tag{9}$$
$$w(1,t) = 0 \tag{10}$$

with the help of feedback (4). We could as well pursue the target system $w_t = w_{xx} - cw, c > 0$, but we forego this design flexibility for the sake of simplicity.

To convert (1), (2), (3) into (8), (9), (10), k needs to satisfy

$$k_{xx}(x,y) - k_{yy}(x,y) = \lambda(y)k(x,y), \quad \forall (x,y) \in \check{\mathscr{T}} \tag{11}$$
$$k(x,0) = 0 \tag{12}$$
$$k(x,x) = -\frac{1}{2}\int_0^x \lambda(y)dy \tag{13}$$

where $\check{\mathscr{T}} = \{0 < y \leq x < 1\}$ and $\mathscr{T} = \{0 \leq y \leq x \leq 1\}$.

The following assumption is important.

Assumption 2 $\lambda \in C^1([0,1])$.

3 Accuracy of Approximation of Backstepping Kernel Operator with DeepONet

Theorem 3 (proven in [77,81]) *For every $\lambda \in C^1([0,1])$, the PDE system* (11)*,* (12)*,* (13) *has a unique $C^2(\mathscr{T})$ solution with the property*

$$|k(x,y)| \leq \bar{\lambda} e^{2\bar{\lambda}x}, \tag{14}$$

for all $x \in [0,1]$, where $\bar{\lambda} = \sup_{x\in[0,1]} |\lambda(x)|$.

This theorem is proven by representing the PDE system (11), (12), (13) as an integral equation

$$G(\xi,\eta) = -\frac{1}{4}\int_\eta^\xi \lambda\left(\frac{s}{2}\right)ds + \frac{1}{4}\int_\eta^\xi \int_0^\eta \lambda\left(\frac{\sigma - s}{2}\right)G(\sigma,s)dsd\sigma, \tag{15}$$

where

$$\xi = x + y, \quad \eta = x - y, \quad x = \frac{\xi+\eta}{2}, \quad y = \frac{\xi-\eta}{2} \tag{16}$$
$$G(\xi,\eta) = k(x,y) = k\left(\frac{\xi+\eta}{2}, \frac{\xi-\eta}{2}\right). \tag{17}$$

The change of variables (16) converts the domain $\mathscr{T}$ for (x,y) into the larger triangular domain $\mathscr{T}_1 = \{0 \leq \eta \leq \xi \leq 1\} \cup \{1 \leq \xi \leq 2 - \eta \leq 2\}$ for (ξ,η). The integral equation (15) is one of the useful approaches in generating solutions for $k(x,y)$ for the purpose of training the neural approximation of the operator $\lambda \mapsto k$.

Next, denote the set of functions

$$\underline{K} = \left\{ k \in C^2(\mathscr{T}) \middle| k(x,0) = 0, \forall x \in [0,1] \right\} \tag{18}$$

and let the operator $\mathscr{K} : C^1[0,1] \to \underline{K}$ be defined by

$$k(x,y) =: \mathscr{K}(\lambda)(x,y). \tag{19}$$

Additionally, let the operator $\mathscr{M} : C^1[0,1] \to \underline{K} \times C^1[0,1] \times C^0(\mathscr{T})$ be defined by

$$(k(x,y), \kappa_1(x), \kappa_2(x,y)) =: \mathscr{M}(\lambda)(x,y), \tag{20}$$

where

$$\kappa_1(x) = 2\frac{d}{dx}(k(x,x)) + \lambda(x) \tag{21}$$
$$\kappa_2(x,y) = k_{xx}(x,y) - k_{yy}(x,y) - \lambda(y)k(x,y). \tag{22}$$

Based on Theorem 3, $\mathscr{M}$ is a continuous operator. By applying Theorem 1, we get the following key result for the approximation of a backstepping kernel by a DeepONet (top of Figure 2).

Theorem 4 *For all $B_\lambda, B_{\lambda'} > 0$ and $\varepsilon > 0$, there exists a neural operator $\hat{\mathscr{M}}$ such that, for all $(x,y) \in \mathscr{T}$,*

$$\left|\mathscr{M}(\lambda)(x,y) - \hat{\mathscr{M}}(\lambda)(x,y)\right| < \varepsilon \tag{23}$$

holds for all Lipschitz λ with the properties that $\|\lambda\|_\infty \leq B_\lambda, \|\lambda'\|_\infty \leq B_{\lambda'}$, namely, there exists a neural operator $\hat{\mathscr{K}}$

such that $\hat{\mathscr{K}}(\lambda)(x,0) \equiv 0$ *and*

$$\begin{aligned}&\left|\mathscr{K}(\lambda)(x,y)-\hat{\mathscr{K}}(\lambda)(x,y)\right|\\&+\left|2\frac{d}{dx}\left(\mathscr{K}(\lambda)(x,x)-\hat{\mathscr{K}}(\lambda)(x,x)\right)\right|\\&+\left|(\partial_{xx}-\partial_{yy})\left(\mathscr{K}(\lambda)(x,y)-\hat{\mathscr{K}}(\lambda)(x,y)\right)\right.\\&\left.-\lambda(y)\left(\mathscr{K}(\lambda)(x,y)-\hat{\mathscr{K}}(\lambda)(x,y)\right)\right|<\varepsilon. \qquad (24)\end{aligned}$$

4 Stabilization under DeepONet Gain Feedback

The following theorem establishes the properties of the feedback system at the bottom of Figure 2.

Theorem 5 *Let* $B_\lambda, B_{\lambda'} > 0$ *be arbitrarily large and consider the system* (1), (2), (3) *with any* $\lambda \in C^1([0,1])$ *whose derivative* λ' *is Lipschitz and which satisfies* $\|\lambda\|_\infty \le B_\lambda$ *and* $\|\lambda'\|_\infty \le B_{\lambda'}$. *There exists a sufficiently small* $\varepsilon^*(B_\lambda, B_{\lambda'}) > 0$ *such that the feedback law*

$$U(t) = \int_0^1 \hat{k}(1,y)u(y,t)dy, \qquad (25)$$

with all NO gain kernels $\hat{k} = \hat{\mathscr{K}}(\lambda)$ *of approximation accuracy* $\varepsilon \in (0, \varepsilon^*)$ *in relation to the exact backstepping kernel* $k = \mathscr{K}(\lambda)$ *ensures that the closed-loop system satisfies the exponential stability bound*

$$\|u(t)\| \le Me^{-(t-t_0)/2}\|u_0\|, \quad \forall t \ge t_0, \qquad (26)$$

where

$$M(\varepsilon,\bar{\lambda}) = \left(1+\bar{\lambda}e^{2\bar{\lambda}}\right)\left(1+\bar{\lambda}e^{2\bar{\lambda}}+\varepsilon\right)e^{\bar{\lambda}e^{2\bar{\lambda}}+\varepsilon}. \qquad (27)$$

PROOF. *Approximate backstepping transform and perturbed target system.* Take the backstepping transformation

$$\hat{w}(x,t) = u(x,t) - \int_0^x \hat{k}(x,y)u(y,t)dy, \qquad (28)$$

where $\hat{k} = \hat{\mathscr{K}}(\lambda)$. With the control law (25), the target system becomes

$$\begin{aligned}\hat{w}_t(x,t) &= \hat{w}_{xx}(x,t) + \delta_{k0}(x)u(x,t)\\&\quad + \int_0^x \delta_{k1}(x,y)u(y,t)dy \qquad (29)\\ \hat{w}(0,t) &= 0 \qquad (30)\\ \hat{w}(1,t) &= 0, \qquad (31)\end{aligned}$$

with

$$\begin{aligned}\delta_{k0}(x) &= 2\frac{d}{dx}\left(\hat{k}(x,x)\right)+\lambda(x)\\&= -2\frac{d}{dx}\left(\tilde{k}(x,x)\right) \qquad (32)\\ \delta_{k1}(x,y) &= \partial_{xx}\hat{k}(x,y) - \partial_{yy}\hat{k}(x,y) - \lambda(y)\hat{k}(x,y)\\&= -\partial_{xx}\tilde{k}(x,y) + \partial_{yy}\tilde{k}(x,y) + \lambda(y)\tilde{k}(x,y), \qquad (33)\end{aligned}$$

where

$$\tilde{k} = k - \hat{k} = \mathscr{K}(\lambda) - \hat{\mathscr{K}}(\lambda). \qquad (34)$$

With (24), we get

$$\begin{aligned}\|\delta_{k0}\|_\infty &\le \varepsilon \qquad (35)\\ \|\delta_{k1}\|_\infty &\le \varepsilon. \qquad (36)\end{aligned}$$

Inverse approximate backstepping transformation. Since the state u appears under the integral in the $\hat{w}$-system (29), in the Lyapunov analysis we need the inverse backstepping transformation

$$u(x,t) = \hat{w}(x,t) + \int_0^x \hat{l}(x,y)\hat{w}(y,t)dy. \qquad (37)$$

It is shown in [47] that the direct and inverse backstepping kernels satisfy in general the relationship

$$\hat{l}(x,y) = \hat{k}(x,y) + \int_y^x \hat{k}(x,\xi)\hat{l}(\xi,y)dy. \qquad (38)$$

The inverse kernel satisfies the following conservative bound

$$\|\hat{l}\|_\infty \le \|\hat{k}\|_\infty e^{\|\hat{k}\|_\infty}. \qquad (39)$$

Since $\|k-\hat{k}\|_\infty < \varepsilon$, we have that $\|\hat{k}\|_\infty \le \|k\|_\infty + \varepsilon$. With (14) we get

$$\begin{aligned}\|\hat{k}\|_\infty &\le \bar{k} + \varepsilon \qquad (40)\\ \bar{k}(\bar{\lambda}) &:= \bar{\lambda}e^{2\bar{\lambda}}, \qquad (41)\end{aligned}$$

and hence

$$\|\hat{l}\|_\infty \le \left(\bar{\lambda}e^{2\bar{\lambda}}+\varepsilon\right)e^{\bar{\lambda}e^{2\bar{\lambda}}+\varepsilon}. \qquad (42)$$

Lyapunov analysis. The Lyapunov functional

$$V = \frac{1}{2}\|\hat{w}\|^2 \qquad (43)$$

has a derivative

$$\dot{V} = -\|\hat{w}_x\|^2 + \Delta_0 + \Delta_1, \qquad (44)$$

where

$$\begin{aligned}\Delta_0(t) &= \int_0^1 \hat{w}(x,t)\delta_{k0}(x)u(x,t)dx \qquad (45)\\ \Delta_1(t) &= \int_0^1 \hat{w}(x,t)\int_0^x \delta_{k1}(x,y)u(y,t)dydx. \qquad (46)\end{aligned}$$

With several straightforward majorizations, we get

$$\begin{aligned}\Delta_0 &\le \|\delta_{k0}\|_\infty \left(1+\|\hat{l}\|_\infty\right)\|\hat{w}\|^2 \\ &= \|\delta_{k0}\|_\infty \left(1+\|\hat{l}\|_\infty\right) 2V. \end{aligned} \tag{47}$$

and

$$\begin{aligned}\Delta_1 &= \int_0^1 \hat{w}(x)\int_0^y \hat{w}(y)\int_y^x \delta_k(x,\sigma)\hat{l}(\sigma,y)d\sigma dydx \\ &\quad + \int_0^1 \hat{w}(x)\int_0^x \delta(x,y)\hat{w}(y)dydx \\ &\le \|\delta_{k1}\|_\infty \left(1+\|\hat{l}\|_\infty\right)\|\hat{w}\|^2 \\ &= \|\delta_{k1}\|_\infty \left(1+\|\hat{l}\|_\infty\right) 2V. \end{aligned} \tag{48}$$

From (44), (47), (48), (42), and Poincare's inequality, we get

$$\dot{V} \le -\frac{1}{2}(1-\delta^*)V, \tag{49}$$

where

$$\delta^*(\varepsilon,\bar{\lambda}) = 2\varepsilon\left(1+\bar{\lambda}e^{2\bar{\lambda}}+\varepsilon\right)e^{\bar{\lambda}e^{2\bar{\lambda}}+\varepsilon} \tag{50}$$

is an increasing function of $\varepsilon,\bar{\lambda}$, with the property that $\delta^*(0,\bar{\lambda})=0$. Hence, there exists $\varepsilon^*(\bar{\lambda})$ such that, for all $\varepsilon\in[0,\varepsilon^*]$,

$$\dot{V} \le -\frac{1}{4}V, \tag{51}$$

namely, $V(t)\le V_0 e^{-(t-t_0)/4}$. From the direct and inverse backstepping transformations it follows that

$$\frac{1}{1+\|\hat{l}\|_\infty}\|u\| \le \sqrt{2V} \le \left(1+\|\hat{k}\|_\infty\right)\|u\|. \tag{52}$$

In conclusion,

$$\|u(t)\| \le \left(1+\|\hat{l}\|_\infty\right)\left(1+\|\hat{k}\|_\infty\right)e^{-(t-t_0)/2}\|u_0\|. \tag{53}$$

With (40), (41), (42), the proof is completed. □

5 Observer Design

State estimators (observers) with boundary measurements can be formulated with four measurement choices on the interval $[0,1]$: the measured quantities can be $u(0,t), u_x(0,t), u(1,t), u_x(1,t)$. That leads to many possible problem formulations. The possibilities multiply once we note that, on the opposite boundary from the one at which measurement is conducted, one can have either a Dirichlet or Neumann (or even Robin) boundary condition. Our objective in this paper is not to solve all the possible problems. We are concerned only with illustrating how NOs can be combined with PDE observers. Hence, our choice among the many possibilities is the simplest choice, of the highest pedagogical value.

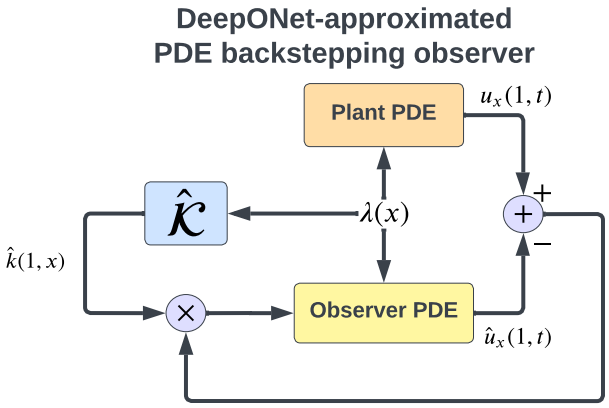

Fig. 4. The PDE backstepping observer (54), (55), (56) uses boundary measurement of the flux $u_x(1,t)$. The gain $\hat{k}(1,t)$ is produced with the DeepONet $\hat{\mathscr{K}}$.

Since our goals with observers are twofold—to estimate the unmeasured state but also to use it in output-feedback control for stabilization—our choice of measurement needs to be consistent with the actuation choice we have already pursued in this note, namely, Dirichlet actuation of $u(1,t)=U(t)$. So, we cannot use $u(1,t)$ for measurement but we can use $u(0,t), u_x(0,t), u_x(1,t)$. We make the last among these three choices. We let the output $u_x(1,t)$ be measured.

Our choice of $u_x(1,t)$ for measurement, as indicated in the observer diagram in Figure 4, is motivated by the fact that, with this measurement, an observer can be built using the same kernel $k(x,y)$ as for the control law. In other words, with a single training of a neural operator $\hat{\mathscr{K}}$, we obtain gains for both a controller and an observer—we kill two birds (pedagogically speaking) with one stone. We don't have to expend an undue amount of the reader's effort on the verification of the conditions of the DeepONet approximation theorem. It is enough for the reader to see once how this is done. The rest of the effort is better spent on illustrating the uses of this approximation capability in observers and output-feedback controllers.

Theorem 6 *Let $B_\lambda, B_{\lambda'} > 0$ be arbitrarily large and consider the system* (1), (2), (3) *with any $\lambda \in C^1([0,1])$ whose derivative λ' is Lipschitz and which satisfies $\|\lambda\|_\infty \le B_\lambda$ and $\|\lambda'\|_\infty \le B_{\lambda'}$. There exists a sufficiently small $\varepsilon^*(B_\lambda, B_{\lambda'}) > 0$ such that the observer*

$$\begin{aligned}\hat{u}_t(x,t) &= \hat{u}_{xx}(x,t)+\lambda(x)\hat{u}(x,t) \\ &\quad -\hat{k}(1,x)\left[u_x(1,t)-\hat{u}_x(1,t)\right] \end{aligned} \tag{54}$$

$$\hat{u}(0,t) = 0 \tag{55}$$

$$\hat{u}(1,t) = U(t), \tag{56}$$

with all NO gain kernels $\hat{k}=\hat{\mathscr{K}}(\lambda)$ of approximation accuracy $\varepsilon\in(0,\varepsilon^*)$ in relation to the exact backstepping kernel $k=\mathscr{K}(\lambda)$ ensure that the observer error system, for all

$u_0, \hat{u}_0 \in L^2[0,1]$, *satisfies the exponential stability bound*

$$\|u(t)-\hat{u}(t)\| \le M\mathrm{e}^{-(t-t_0)/2}\|u_0-\hat{u}_0\|, \quad \forall t \ge t_0, \tag{57}$$

where $M(\varepsilon,\bar{\lambda})$ *is defined in* (27).

PROOF. We start by postulating a PDE backstepping observer in the form

$$\begin{aligned}\hat{u}_t(x,t) &= \hat{u}_{xx}(x,t)+\lambda(x)\hat{u}(x,t)\\ &\quad +p_1(x)\left[u_x(1,t)-\hat{u}_x(1,t)\right] \qquad (58)\\ \hat{u}(0,t) &= 0 \qquad (59)\\ \hat{u}(1,t) &= U(t). \qquad (60)\end{aligned}$$

The observer error $\tilde{u}(x,t)=u(x,t)-\hat{u}(x,t)$ is governed by the system

$$\begin{aligned}\tilde{u}_t(x,t) &= \tilde{u}_{xx}(x,t)+\lambda(x)\tilde{u}(x,t)\\ &\quad -p_1(x)\tilde{u}_x(1,t) \qquad (61)\\ \tilde{u}(0,t) &= 0 \qquad (62)\\ \tilde{u}(1,t) &= 0. \qquad (63)\end{aligned}$$

The backstepping transformation

$$\tilde{u}(x,t)=\tilde{w}(x,t)-\int_x^1 p(x,y)\tilde{w}(y,t)dy \tag{64}$$

converts (61), (62), (63) into

$$\begin{aligned}\tilde{w}_t(x,t) &= \tilde{w}_{xx}(x,t) \qquad (65)\\ \tilde{w}(0,t) &= 0 \qquad (66)\\ \tilde{w}(1,t) &= 0 \qquad (67)\end{aligned}$$

provided $p(x,y)$ satisfies

$$p(x,y)=k(y,x) \tag{68}$$

with k that is governed by (11), (12), (13), and with the observer gain

$$p_1(x)=-k(1,x). \tag{69}$$

It is crucial to note in (68) that the arguments x and y have been commuted in $k(\cdot,\cdot)$. The commuting of the spatial arguments of the backstepping kernel is akin to transposing matrices in going between designs for controllers and observers in finite-dimensional LTI systems. The commuted order of the arguments x and y continues in the rest of the proof. The observer (58), (59), (60) is next rewritten as

$$\begin{aligned}\hat{u}_t(x,t) &= \hat{u}_{xx}(x,t)+\lambda(x)\hat{u}(x,t)\\ &\quad -k(1,x)\left[u_x(1,t)-\hat{u}_x(1,t)\right] \qquad (70)\\ \hat{u}(0,t) &= 0 \qquad (71)\\ \hat{u}(1,t) &= U(t) \qquad (72)\end{aligned}$$

and the transformation (64) as

$$\tilde{u}(x,t)=\tilde{w}(x,t)-\int_x^1 k(y,x)\tilde{w}(y,t)dy. \tag{73}$$

Theorem 4 applies to the kernel $k(y,x)$ of the observer backstepping transformation and the observer gains $-k(1,x)$. The observer (70), (71), (72) is henceforth implemented with the approximate kernel $\hat{k}$ as in (54), (55), (56) whereas the backstepping transformation (73) is applied with $\hat{k}$ as

$$\tilde{u}(x,t)=\omega(x,t)-\int_x^1 \hat{k}(y,x)\omega(y,t)dy. \tag{74}$$

The target system under the approximate kernel $\hat{k}$ becomes

$$\begin{aligned}\omega_t(x,t) &= \omega_{xx}(x,t)+\Omega_0(x,t)+\Omega_1(x,t) \qquad (75)\\ \omega(0,t) &= 0 \qquad (76)\\ \omega(1,t) &= 0, \qquad (77)\end{aligned}$$

where

$$\Omega_0(x,t)=\delta_{k0}(x)\omega(x,t)+\int_x^1 \hat{l}(y,x)\delta_{k0}(y)\omega(y,t)dy \tag{78}$$

$$\begin{aligned}\Omega_1(x,t) &= \int_x^1 \Big(\delta_{k1}(y,x)\omega(y,t)\\ &\quad +\hat{l}(y,x)\int_y^1 \delta_{k1}(s,y)\omega(s,t)ds\Big)dy \qquad (79)\end{aligned}$$

and δ_{k0}, δ_{k1} are defined in (32), (33), with bounds (35), (36). Note that the arguments in δ_{k1} have been commuted in the integral in (75). Similar as in the proof of Theorem 5, the Lyapunov functional

$$V=\frac{1}{2}\|\omega\|^2 \tag{80}$$

has a derivative

$$\dot{V} \le -\frac{1}{4}V, \tag{81}$$

namely, $V(t) \le V_0\mathrm{e}^{-(t-t_0)/4}$, provided $\varepsilon \in [0,\varepsilon^*]$, with ε^* obtained from (50). The result (54) follows from (80), (74), (40), and the inverse backstepping transformation

$$\omega(x,t)=\tilde{u}(x,t)+\int_x^1 \hat{l}(y,x)\tilde{u}(y,t)dy \tag{82}$$

whose kernel $\hat{l}$ satisfies the bound (42). □

6 Collocated Output-Feedback Stabilization

In this section we put together the observer (54), (55), (56), along with the observer-based controller

$$U(t)=\int_0^1 \hat{k}(1,x)\hat{u}(x,t)dx \tag{83}$$

to stabilize the system (1), (2), (3) by output feedback.

The backstepping transformations

$$\check{w}(x,t) = \hat{u}(x,t) - \int_0^x \hat{k}(x,y)\hat{u}(y,t)dy \tag{84}$$

$$\tilde{u}(x,t) = \omega(x,t) - \int_x^1 \hat{k}(y,x)\omega(y,t)dy. \tag{85}$$

transform the overall system into the cascade

$$\begin{aligned}\check{w}_t(x,t) &= \check{w}_{xx}(x,t) + \delta_{k0}(x)\check{w}(x,t)dy \\ &+\delta_{k0}(x,y)\int_0^x \hat{l}(x,y)\check{w}(y,t)dy \\ &+\int_0^x \delta_{k1}(x,y)\check{w}(y,t)dy \\ &+\int_0^x \delta_{k1}(x,y)\int_0^y \hat{l}(y,\eta)\check{w}(\eta,t)d\eta dy \\ &-\left(\hat{k}(1,x) - \int_0^x \hat{k}(x,y)\hat{k}(1,y)dy\right)\omega_x(1,t)\end{aligned} \tag{86}$$

$$\check{w}(0,t) = 0 \tag{87}$$

$$\check{w}(1,t) = 0 \tag{88}$$

$$\begin{aligned}\omega_t(x,t) &= \omega_{xx}(x,t) + \delta_{k0}(x)\omega(x,t) \\ &+\int_x^1 \hat{l}(y,x)\delta_{k0}(y)\omega(y,t)dy \\ &+\int_x^1 (\delta_{k1}(y,x)\omega(y,t) \\ &+\hat{l}(y,x)\int_y^1 \delta_{k1}(s,y)\omega(s,t)ds\Big)dy\end{aligned} \tag{89}$$

$$\omega(0,t) = 0 \tag{90}$$

$$\omega(1,t) = 0. \tag{91}$$

Both the ω-subsystem (89)–(91), which is autonomous, and the $\check{w}$-subsystem (86)–(88), which is driven by the output $\omega_x(1,t)$ of the ω-subsystem, are exponentially stable in $L^2[0,1]$ and higher norms for sufficiently small ε. However, because the trace term $\omega_x(1,t)$ in the last line of (86) cannot be easily bounded even by an H^2 norm of ω, we do not pursue a stability analysis of the composite system, i.e., we leave the "separation principle" unproven for the observer-based feedback (54), (55), (56), (83) acting on the system (1), (2), (3). The technical challenge has nothing to do with the NO implementation of the kernel $\hat{k}$, as the challenge does not arise due to the perturbation kernels δ_{k0}, δ_{k1}. The challenge is due to the unbounded nature of the *output mapping* $\omega(t) \mapsto \omega_x(1,t)$, a challenge not encountered in ODEs but only in PDE control with boundary sensing or actuation.

The result given next, which is of a slightly more complicated form, is provable but we give it without a proof because the calculations are very, very lengthy and partly duplicate the calculations in the previous sections. The actuation-sensing setup is from the last row of Table 1, namely, a collocated Neumann actuation and Dirichlet sensing. Stability established is in the H^1 norm $\|u(t)\|_{H^1} + \|\hat{u}(t)\|_{H^1}$.

Theorem 7 *Consider the system*

$$u_t(x,t) = u_{xx}(x,t) + \lambda(x)u(x,t), \qquad x \in [0,1) \tag{92}$$

$$u(0,t) = 0 \tag{93}$$

$$u_x(1,t) = U(t) \tag{94}$$

with a measured Dirichlet output $u(1,t)$, along with the collocated observer-based Neumann-actuated controller

$$\begin{aligned}\hat{u}_t(x,t) &= \hat{u}_{xx}(x,t) + \lambda(x)\hat{u}(x,t) \\ &+\kappa(x)\left[u(1,t) - \hat{u}(1,t)\right]\end{aligned} \tag{95}$$

$$\hat{u}(0,t) = 0 \tag{96}$$

$$\hat{u}_x(1,t) = U(t) - \hat{k}(1,1)\left(u(1,t) - \hat{u}(1,t)\right) \tag{97}$$

$$U(t) = \hat{k}(1,1)u(1,t) + \int_0^1 \kappa(x)\hat{u}(x,t)dx, \tag{98}$$

where the gain function of both the controller and the observer is given by

$$\kappa(x) := \hat{k}_\xi(\xi,x)\big|_{\xi=1}. \tag{99}$$

For all $B_\lambda, B_{\lambda'} > 0$ and for all $\lambda \in C^1([0,1])$ whose derivative is Lipschitz and which satisfies $\|\lambda\|_\infty \le B_\lambda$ and $\|\lambda'\|_\infty \le B_{\lambda'}$, there exists a sufficiently small $\varepsilon^*(B_\lambda, B_{\lambda'}) > 0$ such that for all NO gain kernels $\hat{k} = \mathscr{K}(\lambda)$ of approximation accuracy $\varepsilon \in (0,\varepsilon^*)$ in relation to the exact backstepping kernel $k = \mathscr{K}(\lambda)$ there exists sufficiently large $M(\varepsilon,\bar{\lambda}) > 0$ such that the above observer-based feedback ensures that the closed-loop system, for all $u_0, \hat{u}_0 \in H^1[0,1]$, satisfies the exponential stability bound

$$\|u(t)\|_{H^1} + \|\hat{u}(t)\|_{H^1} \le Me^{-(t-t_0)/4}\left(\|u_0\|_{H^1} + \|\hat{u}_0\|_{H^1}\right) \tag{100}$$

for all $t \ge t_0$.

The proof is based on the backstepping transformations (84), (85) into a perturbed version of the target system

$$\begin{aligned}\check{w}_t(x,t) &= \check{w}_{xx}(x,t) \\ &+\left(\kappa(x) - \int_0^x \kappa(y)\hat{k}(x,y)dy\right)\omega(1,t)\end{aligned} \tag{101}$$

$$\check{w}(0,t) = 0 \tag{102}$$

$$\check{w}(1,t) = 0 \tag{103}$$

$$\omega_t(x,t) = \omega_{xx}(x,t) \tag{104}$$

$$\omega(0,t) = 0 \tag{105}$$

$$\omega(1,t) = 0. \tag{106}$$

The perturbation terms are as in (86) and (89), employing the functions δ_{k0} and δ_{k1} (which are uniformly bounded by ε). The Lyapunov analysis employs the H^1 norms of $\check{w}$ and ω, along with Agmon's inequality to bound the perturbation term $\omega(1,t)$ in the $\check{w}$-system using the norm $\|\omega_x\|$.

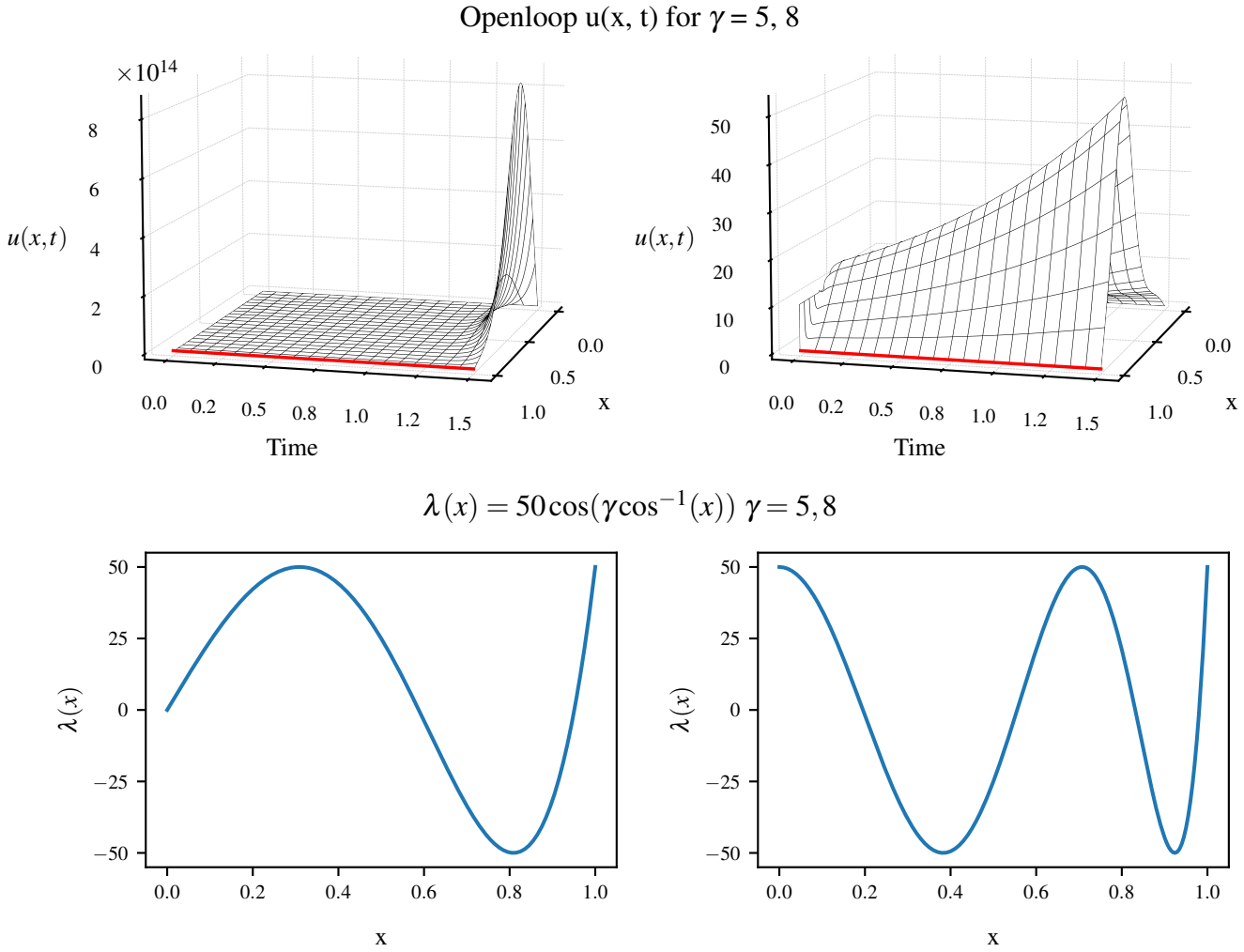

Fig. 5. Open-loop instability (top) for the two respective reaction coefficients $\lambda(x)$ shown on the bottom row.

7 Numerical Results: Full-State Feedback, Observer, and Output Feedback

In Figure 5, we show that the system is open-loop unstable for the reaction term $\lambda(x) = 50\cos(\gamma\cos^{-1}(x))$ for $\gamma = 5, 8$. The increased oscillation in larger γ yields a lower rate of instability as shown on the right. We simulate the PDE and its control using the finite difference scheme in Appendix B.

In Figure 6, we demonstrate both the analytical and learned DeepONet kernels for the two γ values corresponding to Figure 5. To learn the mapping $\mathscr{K} : \lambda(x) \mapsto k(x,y)$, we construct a dataset of 900 different $\lambda(x)$ as the Chebyshev polynomials define above with $\gamma \sim$ uniform $(4,9)$. We choose λ of this form due to the rich set of kernel functions generated by varying only a single parameter. To effectively utilize the DeepONet without modifying the grid, we stack $\lambda(x)$ repeatedly n_y times over the y axis to make a 2D input to the network. Then, we capitalize on the 2D mapping by implementing a CNN for the DeepONet branch network. In the future, one can explore neural operators on irregular girds along the direction of [52]. For training, the relative L_2 error is $3.5e-2$ and the testing error is $3.6e-2$. With the learned neural operator, we achieve speedups on the magnitude of 10^3 compared to an efficient finite difference implementation. In Figure 7, we demonstrate closed-loop stability with the neural operator approximated gain function for the control feedback law. Additionally, we see the error is largest at the beginning achieving a maximum in both cases of approximately 10%.

In Figure 8, we test the observer (55), (55), (56) with a DeepONet-approximated kernel trained as above using $\lambda(x) = 20\cos(5\cos^{-1}(x))$ with $\gamma \sim$ uniform $(4,9)$. Additionally, we apply a boundary signal of $U(t) = 7\sin(16\pi t) + 10\cos(2\pi t)$ to generate a challenging and rich PDE motion for estimation. The true system state begins with initial conditions $u(x,0) = 10$ while the DeepONet observer has initial conditions of $\hat{u}_{NO}(x,0) = 20$. Despite this, the observer approximates the PDE well with a peak

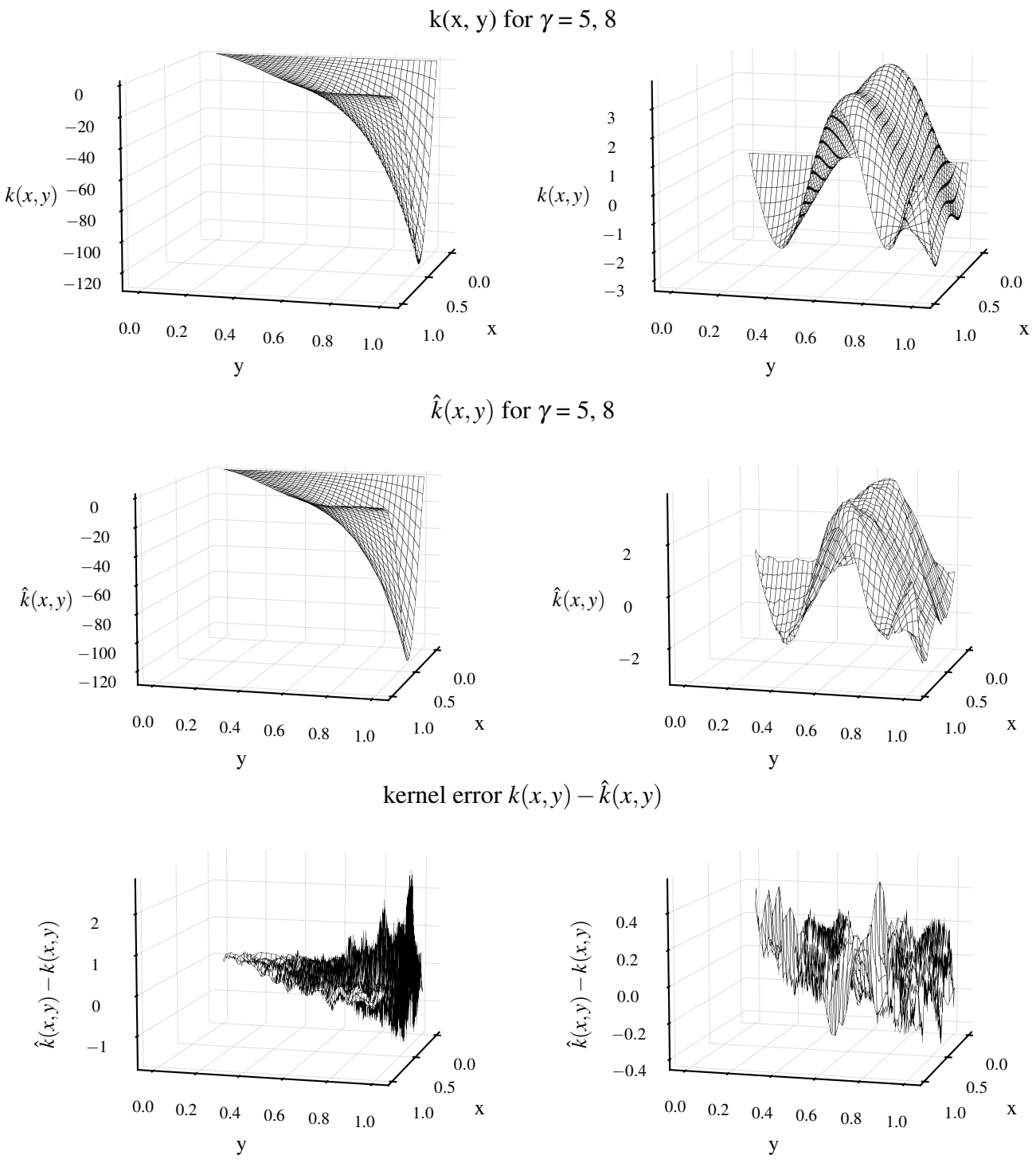

Fig. 6. Examples of the kernel $k(x,y)$ (top row), learned kernel $\hat{k}(x,y)$ (middle row), and the kernel error $k(x,y) - \hat{k}(x,y)$ (bottom row). The two respective $\lambda(x)$ values correspond to the same respective values as in Fig 5.

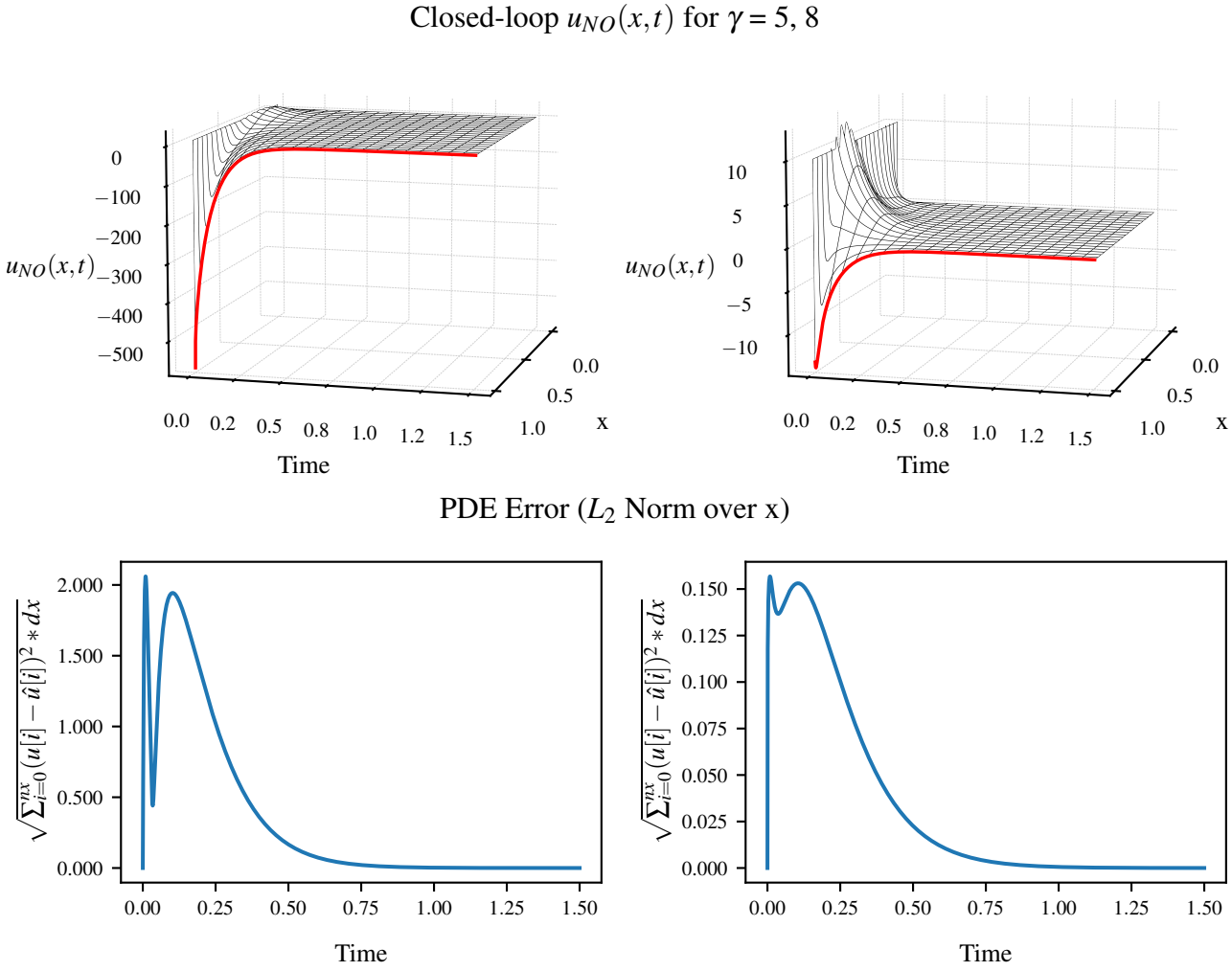

Fig. 7. For the two respective $\lambda(x)$ values as in Fig 5, the top row showcases closed-loop solutions with the learned kernel $\hat{k}(x,y)$, whereas the bottom row shows the closed-loop PDE error between applying the original kernel $k(x,y)$ and the learned kernel $\hat{k}(x,y)$.

error of less than 5% compared to the analytical observer while maintaining the same 10^3x speedup over the finite difference scheme.

8 Conclusions

In this paper, we build on the framework introduced in [10], and depicted in Figures 1 and 2, and extend the neural operator-supported PDE backstepping methodology from the hyperbolic to the harder parabolic case. We limit ourselves to the reaction-diffusion parabolic class for the clarity of exposition.

With the foundation laid for hyperbolic and parabolic PDE backstepping designs, which free the user from having to solve kernel PDEs in real time and result in a *thousandfold speedup*, the road is open to developing this methodology for two important control domains in which the backstepping kernels constantly evolve in the course of implementation: (1) gain scheduling for nonlinear PDEs, where the kernel depends on the current state of the PDE; and (2) adaptive control of PDEs whose functional coefficients are unknown, have to be adaptively estimated online, and the kernel has to be continuously updated. In both applications, the solving of the k-PDE online is eliminated with the aid of the predetermined neural operator $\hat{\mathscr{K}}$.

Appendix

A Neural networks notation

An n-layer neural network (NN) $f^{\mathscr{N}} : \mathbb{R}^{d_1} \to \mathbb{R}^{d_n}$ is given by

$$f^{\mathscr{N}}(x,\theta) := (l_n \circ l_{n-1} \circ \ldots \circ l_2 \circ l_1)(x,\theta) \tag{A.1}$$

where layers l_i start with $l_0 = x \in \mathbb{R}^{d_1}$ and continue as

$$l_{i+1}(l_i,\theta_{i+1}) := \sigma(W_{i+1}l_i + b_{i+1}), \quad i = 1,\ldots,n-1 \tag{A.2}$$

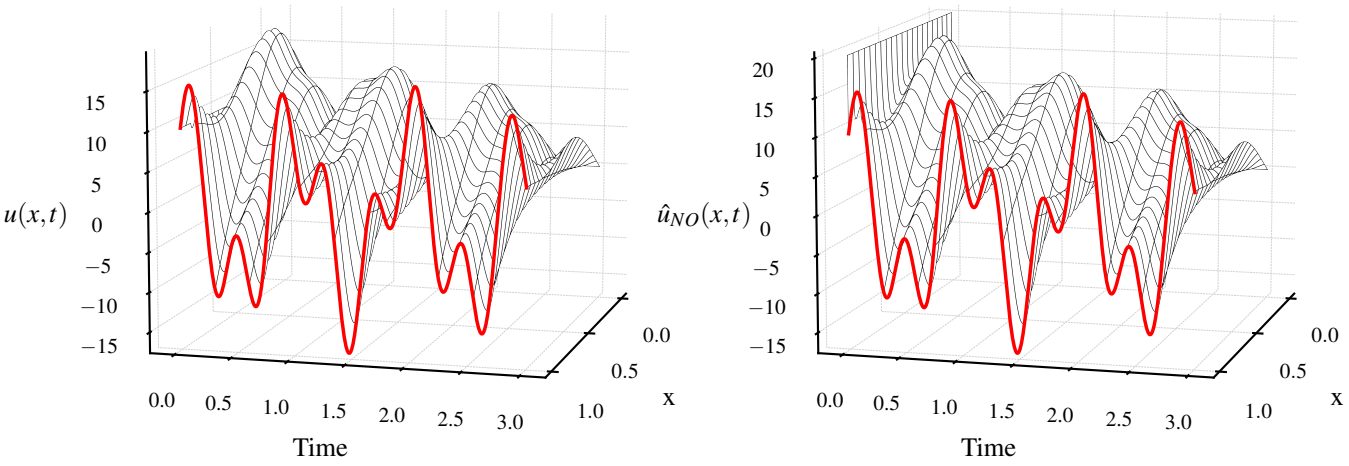

Fig. 8. Left: PDE solution with $\lambda(x) = 20\cos(5\cos^{-1}(x))$ and $U(t) = 7\sin(16\pi t) + 10\cos(2\pi t)$. Right: the observer with the neural operator-learned kernel. Note the difference between the plant initial condition $u(x,0) = 10$ and the twice as large initial condition of the exact and neural operator observers, $\hat{u}_{\text{NO}}(x,0) = \hat{u}(0,x) = 20$. The peak error between the analytical observer (not shown) and the neural operator observer is around 0.3.

σ is a nonlinear activation function, and weights $W_{i+1} \in \mathbb{R}^{d_{i+1}\times d_i}$ and biases $b_{i+1} \in \mathbb{R}^{d_{i+1}}$ are parameters to be learned, collected into $\theta_i \in \mathbb{R}^{d_{i+1}(d_i+1)}$, and then into $\theta = [\theta_1^{\mathrm{T}}, \ldots, \theta_n^{\mathrm{T}}]^{\mathrm{T}} \in \mathbb{R}^{\sum_{i=1}^{n-1} d_{i+1}(d_i+1)}$. Let $\vartheta^{(k)}, \theta^{(k)} \in \mathbb{R}^{\sum_{i=1}^{k-1} d_{k,(i+1)}(d_{k,i}+1)}$ denote a sequence of NN weights.

A neural operator (NO) for approximating a nonlinear operator $\mathscr{G} : \mathscr{U} \mapsto \mathscr{V}$ is defined as

$$\mathscr{G}_{\mathbb{N}}(\mathbf{u}_m)(y) = \sum_{k=1}^{p} g^{\mathscr{N}}(\mathbf{u}_m; \vartheta^{(k)}) f^{\mathscr{N}}(y; \theta^{(k)}) \tag{A.3}$$

where $\mathscr{U}, \mathscr{V}$ are function spaces of continuous functions $u \in \mathscr{U}, v \in \mathscr{V}$. $\mathbf{u}_m$ is the evaluation of function u at points $x_i = x_1, \ldots, x_m$, p is the number of chosen basis components in the target space, $y \in Y$ is the location of the output function $v(y)$ evaluations, and $g^{\mathscr{N}}$, $f^{\mathscr{N}}$ are NNs termed branch and trunk networks. Note, $g^{\mathscr{N}}$ and $f^{\mathscr{N}}$ are not limited to feedforward NNs (A.1), but can also be of convolutional or recurrent.

B FD Scheme for Goursat-Form Kernel PDE

For the PDE in (1), (2), (3), we utilize the following finite difference scheme adapted from [81]:

$$k_j^{i+1} = -k_j^{i-1} + k_{j+1}^i + k_{j-1}^i + h^2\lambda_j \frac{k_{j+1}^i + k_{j-1}^i}{2} \tag{B.1}$$

$$k_i^{i+1} = k_i^i + \frac{h}{2}\lambda_i \tag{B.2}$$

$$k_{i+1}^{i+1} = k_i^i - \frac{h}{4}(\lambda_i + \lambda_{i+1}), \qquad k_1^{j+1} = 0 \tag{B.3}$$

with $k_i^j = k((i-1)h, (j-1)h), i = 2, \ldots, N, j = 2, \ldots, i-1, \lambda_i = \bar{\lambda}((i-1)h), h = 1/N$ where N is the number of spatial steps.

References

[1] H. Anfinsen and O. Aamo. *Adaptive Control of Hyperbolic PDEs*. Springer, 2019.

[2] A. Aswani, H. Gonzalez, S. S. Sastry, and C. Tomlin. Provably safe and robust learning-based model predictive control. *Automatica*, 49(5):1216–1226, 2013.

[3] J. Auriol, F. Bribiesca-Argomedo, D. Saba, M. D. Loreto, and F. Di Meglio. Delay-robust stabilization of a hyperbolic PDE-ODE system. *Automatica*, 95:494–502, 2018.

[4] A. Baccoli, A. Pisano, and Y. Orlov. Boundary control of coupled reaction–diffusion processes with constant parameters. *Automatica*, 54:80–90, 2015.

[5] N. Bekiaris-Liberis and M. Krstic. Compensating the distributed effect of diffusion and counter-convection in multi-input and multi-output lti systems. *IEEE Transactions on Automatic Control*, 56(3):637–643, 2010.

[6] J. Berberich, C. W. Scherer, and F. Allgöwer. Combining prior knowledge and data for robust controller design. *IEEE Transactions on Automatic Control*, pages 1–16, 2022.

[7] F. Berkenkamp, M. Turchetta, A. Schoellig, and A. Krause. Safe model-based reinforcement learning with stability guarantees. *Advances in neural information processing systems*, 30, 2017.

[8] P. Bernard and M. Krstic. Adaptive output-feedback stabilization of non-local hyperbolic PDEs. *Automatica*, 50:2692–2699, 2014.

[9] D. P. Bertsekas. *Dynamic Programming and Optimal Control*, volume I. Athena Scientific, Belmont, MA, USA, 3rd edition, 2005.

[10] L. Bhan, Y. Shi, and M. Krstic. Neural operators for bypassing gain and control computations in PDE backstepping. arXiv, submitted to *IEEE Transactions on Automatic Control*, 2023.

[11] S. Bhasin, R. Kamalapurkar, M. Johnson, K. Vamvoudakis, F. Lewis, and W. Dixon. A novel actor-critic-identifier architecture for approximate optimal control of uncertain nonlinear systems. *Automatica (Journal of IFAC)*, 49(1):82–92, 2013.

[12] N. Boffi, S. Tu, N. Matni, J.-J. Slotine, and V. Sindhwani. Learning stability certificates from data. In *Conference on Robot Learning*, pages 1341–1350. PMLR, 2021.

[13] Y.-C. Chang, N. Roohi, and S. Gao. Neural lyapunov control. *Advances in neural information processing systems*, 32, 2019.

[14] S. Chen, M. Fazlyab, M. Morari, G. J. Pappas, and V. M. Preciado. Learning lyapunov functions for piecewise affine systems with neural network controllers. *arXiv preprint arXiv:2008.06546*, 2020.

[15] S. Chen, M. Fazlyab, M. Morari, G. J. Pappas, and V. M. Preciado. Learning lyapunov functions for hybrid systems. In *Proceedings of the 24th International Conference on Hybrid Systems: Computation and Control*, pages 1–11, 2021.

[16] X. Chen and E. Hazan. Black-box control for linear dynamical systems. In *Conference on Learning Theory*, pages 1114–1143. PMLR, 2021.

[17] Y. Chow, O. Nachum, E. Duenez-Guzman, and M. Ghavamzadeh. A lyapunov-based approach to safe reinforcement learning. *Advances in neural information processing systems*, 2018.

[18] J. Coron, R. Vazquez, M. Krstic, and G. Bastin. Local exponential H^2 stabilization of a 2×2 quasilinear hyperbolic system using backstepping. *SIAM Journal on Control and Optimization*, 51(3):2005–2035, 2013.

[19] J.-M. Coron and H.-M. Nguyen. Null controllability and finite time stabilization for the heat equations with variable coefficients in space in one dimension via backstepping approach. *Archive for Rational Mechanics and Analysis*, 225:993–1023, 2017.

[20] W. Cui, Y. Jiang, B. Zhang, and Y. Shi. Structured neural-pi control for networked systems: Stability and steady-state optimality guarantees. *arXiv preprint arXiv:2206.00261*, 2022.

[21] H. Dai, B. Landry, L. Yang, M. Pavone, and R. Tedrake. Lyapunov-stable neural-network control, 2021.

[22] C. Dawson, S. Gao, and C. Fan. Safe control with learned certificates: A survey of neural lyapunov, barrier, and contraction methods. *arXiv preprint arXiv:2202.11762*, 2022.

[23] C. De Persis, M. Rotulo, and P. Tesi. Learning controllers from data via approximate nonlinearity cancellation. *IEEE Transactions on Automatic Control*, pages 1–16, 2023.

[24] S. Dean, H. Mania, N. Matni, B. Recht, and S. Tu. Regret bounds for robust adaptive control of the linear quadratic regulator. *Advances in Neural Information Processing Systems*, 31, 2018.

[25] B. Deng, Y. Shin, L. Lu, Z. Zhang, and G. E. Karniadakis. Convergence rate of deeponets for learning operators arising from advection-diffusion equations, 2021.

[26] J. Deutscher. A backstepping approach to the output regulation of boundary controlled parabolic pdes. *Automatica*, 57:56–64, 2015.

[27] J. Deutscher and J. Gabriel. Minimum time output regulation for general linear heterodirectional hyperbolic systems. *International Journal of Control*, 93:1826–1838, 2018.

[28] F. Di Meglio, R. Vazquez, and M. Krstic. Stabilization of a system of $n+1$ coupled first-order hyperbolic linear PDEs with a single boundary input. *IEEE Transactions on Automatic Control*, 58:3097–3111, 2013.

[29] F. Di Meglio, F. B. Argomedo, L. Hu, and M. Krstic. Stabilization of coupled linear heterodirectional hyperbolic pde–ode systems. *Automatica*, 87:281–289, 2018.

[30] N. Espitia, I. Karafyllis, and M. Krstic. Event-triggered boundary control of constant-parameter reaction–diffusion pdes: A small-gain approach. *Automatica*, 128:109562, 2021.

[31] N. Espitia, A. Polyakov, D. Efimov, and W. Perruquetti. Boundary time-varying feedbacks for fixed-time stabilization of constant-parameter reaction–diffusion systems. *Automatica*, 103:398–407, 2019.

[32] M. Fazel, R. Ge, S. Kakade, and M. Mesbahi. Global convergence of policy gradient methods for the linear quadratic regulator. In *International conference on machine learning*, pages 1467–1476. PMLR, 2018.

[33] T. Fiez, B. Chasnov, and L. Ratliff. Implicit learning dynamics in stackelberg games: Equilibria characterization, convergence analysis, and empirical study. In *International Conference on Machine Learning*, pages 3133–3144. PMLR, 2020.

[34] B. Hu, K. Zhang, N. Li, M. Mesbahi, M. Fazel, and T. Başar. Towards a theoretical foundation of policy optimization for learning control policies. *arXiv preprint arXiv:2210.04810*, 2022.

[35] L. Hu, F. Di Meglio, R. Vazquez, and M. Krstic. Control of homodirectional and general heterodirectional linear coupled hyperbolic PDEs. *IEEE Transactions on Automatic Control*, 61(11):3301–3314, 2016.

[36] L. Hu, R. Vazquez, F. Di Meglio, and M. Krstic. Boundary exponential stabilization of 1-dimensional inhomogeneous quasi-linear hyperbolic systems. *SIAM J. Control and Optimization*, 57(2):963–998, 2019.

[37] S. Kakade, A. Krishnamurthy, K. Lowrey, M. Ohnishi, and W. Sun. Information theoretic regret bounds for online nonlinear control. *arXiv preprint*, 2020.

[38] I. Karafyllis, N. Espitia, and M. Krstic. Event-triggered gain scheduling of reaction-diffusion pdes. *SIAM Journal on Control and Optimization*, 59(3):2047–2067, 2021.

[39] I. Karafyllis and M. Krstic. Sampled-data boundary feedback control of 1-d parabolic pdes. *Automatica*, 87:226–237, 2018.

[40] I. Karafyllis, M. Krstic, and K. Chrysafi. Adaptive boundary control of constant-parameter reaction-diffusion PDEs using regulation-triggered finite-time identification. *Automatica*, 103:166–179, 2019.

[41] G. Kissas, J. H. Seidman, L. F. Guilhoto, V. M. Preciado, G. J. Pappas, and P. Perdikaris. Learning operators with coupled attention. *Journal of Machine Learning Research*, 23(215):1–63, 2022.

[42] S. Koga, M. Diagne, and M. Krstic. Control and state estimation of the one-phase Stefan problem via backstepping design. *IEEE Transactions on Automatic Control*, 64(2):510–525, 2019.

[43] M. Krstic. Compensating actuator and sensor dynamics governed by diffusion pdes. *Systems & Control Letters*, 58(5):372–377, 2009.

[44] M. Krstic. Control of an unstable reaction-diffusion PDE with long input delay. *Systems & Control Letters*, 58:773–782, 2009.

[45] M. Krstic and A. Smyshlyaev. Adaptive boundary control for unstable parabolic pdes—part i: Lyapunov design. *IEEE Transactions on Automatic Control*, 53(7):1575–1591, 2008.

[46] M. Krstic and A. Smyshlyaev. Backstepping boundary control for first-order hyperbolic PDEs and application to systems with actuator and sensor delays. *Systems & Control Letters*, 57(9):750–758, 2008.

[47] M. Krstic and A. Smyshlyaev. *Boundary Control of PDEs: A Course on Backstepping Designs*. SIAM, 2008.

[48] S. Lale, K. Azizzadenesheli, B. Hassibi, and A. Anandkumar. Reinforcement learning with fast stabilization in linear dynamical systems. In *International Conference on Artificial Intelligence and Statistics*, pages 5354–5390. PMLR, 2022.

[49] S. Lale, Y. Shi, G. Qu, K. Azizzadenesheli, A. Wierman, and A. Anandkumar. Kcrl: Krasovskii-constrained reinforcement learning with guaranteed stability in nonlinear dynamical systems. *arXiv preprint arXiv:2206.01704*, 2022.

[50] S. Lanthaler, S. Mishra, and G. E. Karniadakis. Error estimates for DeepONets: a deep learning framework in infinite dimensions. *Transactions of Mathematics and Its Applications*, 6(1), 03 2022. tnac001.

[51] F. L. Lewis, D. Vrabie, and K. G. Vamvoudakis. Reinforcement learning and feedback control: Using natural decision methods to design optimal adaptive controllers. *IEEE Control Systems Mag.*, 32(6):76–105, 2012.

[52] Z. Li, D. Z. Huang, B. Liu, and A. Anandkumar. Fourier neural operator with learned deformations for pdes on general geometries, 2022.

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

[54] Z. Li, N. B. Kovachki, K. Azizzadenesheli, B. liu, K. Bhattacharya, A. Stuart, and A. Anandkumar. Fourier neural operator for parametric partial differential equations. In *International Conference on Learning Representations*, 2021.

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

[56] L. Lu, P. Jin, G. Pang, Z. Zhang, and G. E. Karniadakis. Learning nonlinear operators via deeponet based on the universal approximation theorem of operators. *Nature Machine Intelligence*, 3(3):218–229, 2021.

[57] A. A. Malikopoulos. Separation of learning and control for cyber–physical systems. *Automatica*, 151:110912, 2023.

[58] W. Mao and T. Başar. Provably efficient reinforcement learning in decentralized general-sum markov games. *Dynamic Games and Applications*, pages 1–22, 2022.

[59] E. Mazumdar, L. J. Ratliff, and S. S. Sastry. On gradient-based learning in continuous games. *SIAM Journal on Mathematics of Data Science*, 2(1):103–131, 2020.

[60] T. Meurer. *Control of higher–dimensional PDEs: Flatness and backstepping designs*. Springer Science & Business Media, 2012.

[61] T. Meurer and A. Kugi. Tracking control for boundary controlled parabolic pdes with varying parameters: Combining backstepping and differential flatness. *Automatica*, 45(5):1182–1194, 2009.

[62] H. Mohammadi, A. Zare, M. Soltanolkotabi, and M. R. Jovanović. Convergence and sample complexity of gradient methods for the model-free linear–quadratic regulator problem. *IEEE Transactions on Automatic Control*, 67(5):2435–2450, 2021.

[63] D. Muthirayan, D. Kalathil, and P. P. Khargonekar. Meta-learning online control for linear dynamical systems. In *2022 IEEE 61st Conference on Decision and Control (CDC)*, pages 1435–1440, 2022.

[64] H. H. Nguyen, T. Zieger, S. C. Wells, A. Nikolakopoulou, R. D. Braatz, and R. Findeisen. Stability certificates for neural network learning-based controllers using robust control theory. In *2021 American Control Conference (ACC)*, pages 3564–3569, 2021.

[65] Y. Orlov, A. Pisano, A. Pilloni, and E. Usai. Output feedback stabilization of coupled reaction-diffusion processes with constant parameters. *SIAM Journal on Control and Optimization*, 55(6):4112–4155, 2017.

[66] B. Pang, T. Bian, and Z.-P. Jiang. Robust policy iteration for continuous-time linear quadratic regulation. *IEEE Transactions on Automatic Control*, 67(1):504–511, 2022.

[67] B. Pang, Z.-P. Jiang, and I. Mareels. Reinforcement learning for adaptive optimal control of continuous-time linear periodic systems. *Automatica*, 118:109035, 2020.

[68] D. Pfrommer, T. T. Zhang, S. Tu, and N. Matni. Tasil: Taylor series imitation learning. *arXiv preprint arXiv:2205.14812*, 2022.

[69] J. I. Poveda, M. Krstic, and T. Basar. Fixed-time nash equilibrium seeking in time-varying networks. *IEEE Transactions on Automatic Control*, 2022.

[70] G. Qu, A. Wierman, and N. Li. Scalable reinforcement learning of localized policies for multi-agent networked systems. In *Learning for Dynamics and Control*, pages 256–266. PMLR, 2020.

[71] B. Rathnayake, M. Diagne, N. Espitia, and I. Karafyllis. Observer-based event-triggered boundary control of a class of reaction–diffusion pdes. *IEEE Transactions on Automatic Control*, 67(6):2905–2917, 2021.

[72] U. Rosolia and F. Borrelli. Learning model predictive control for iterative tasks. a data-driven control framework. *IEEE Transactions on Automatic Control*, 63(7):1883–1896, 2017.

[73] J. H. Seidman, G. Kissas, P. Perdikaris, and G. J. Pappas. NOMAD: Nonlinear manifold decoders for operator learning. In A. H. Oh, A. Agarwal, D. Belgrave, and K. Cho, editors, *Advances in Neural Information Processing Systems*, 2022.

[74] Y. Shi, Z. Li, H. Yu, D. Steeves, A. Anandkumar, and M. Krstic. Machine learning accelerated pde backstepping observers. In *2022 IEEE 61st Conference on Decision and Control (CDC)*, pages 5423–5428, 2022.

[75] Y. Shi, G. Qu, S. Low, A. Anandkumar, and A. Wierman. Stability constrained reinforcement learning for real-time voltage control. In *2022 American Control Conference (ACC)*, pages 2715–2721. IEEE, 2022.

[76] S. Singh, S. M. Richards, V. Sindhwani, J.-J. E. Slotine, and M. Pavone. Learning stabilizable nonlinear dynamics with contraction-based regularization. *The International Journal of Robotics Research*, 40(10-11):1123–1150, 2021.

[77] A. Smyshlyaev and M. Krstic. Closed-form boundary state feedbacks for a class of 1-d partial integro-differential equations. *IEEE Transactions on Automatic Control*, 49(12):2185–2202, 2004.

[78] A. Smyshlyaev and M. Krstic. Backstepping observers for a class of parabolic pdes. *Systems & Control Letters*, 54(7):613–625, 2005.

[79] A. Smyshlyaev and M. Krstic. Adaptive boundary control for unstable parabolic pdes—part ii: Estimation-based designs. *Automatica*, 43(9):1543–1556, 2007.

[80] A. Smyshlyaev and M. Krstic. Adaptive boundary control for unstable parabolic pdes—part iii: Output feedback examples with swapping identifiers. *Automatica*, 43(9):1557–1564, 2007.

[81] A. Smyshlyaev and M. Krstic. *Adaptive Control of Parabolic PDEs*. Princeton University Press, 2010.

[82] Y. Tang, Y. Zheng, and N. Li. Analysis of the optimization landscape of linear quadratic gaussian (lqg) control. In *Learning for Dynamics and Control*, pages 599–610. PMLR, 2021.

[83] A. J. Taylor, V. D. Dorobantu, M. Krishnamoorthy, H. M. Le, Y. Yue, and A. D. Ames. A control lyapunov perspective on episodic learning via projection to state stability. In *2019 IEEE 58th Conference on Decision and Control (CDC)*, pages 1448–1455. IEEE, 2019.

[84] K. G. Vamvoudakis. Q-learning for continuous-time linear systems: A model-free infinite horizon optimal control approach. *Systems & Control Letters*, 100:14–20, 2017.

[85] K. G. Vamvoudakis and F. L. Lewis. Online actor-critic algorithm to solve the continuous-time infinite horizon optimal control problem. *Automatica*, 46(5):878–888, 2010.

[86] K. G. Vamvoudakis, H. Modares, B. Kiumarsi, and F. L. Lewis. Game theory-based control system algorithms with real-time reinforcement learning: How to solve multiplayer games online. *IEEE Control Systems Magazine*, 37(1):33–52, 2017.

[87] R. Vazquez and M. Krstic. A closed-form feedback controller for stabilization of the linearized 2-d navier–stokes poiseuille system. *IEEE Transactions on Automatic Control*, 52(12):2298–2312, 2007.

[88] R. Vazquez and M. Krstic. Control of 1-d parabolic pdes with volterra nonlinearities, part i: Design. *Automatica*, 44(11):2778–2790, 2008.

[89] R. Vazquez and M. Krstic. Control of 1d parabolic pdes with volterra nonlinearities, part ii: analysis. *Automatica*, 44(11):2791–2803, 2008.

[90] R. Vazquez and M. Krstic. Boundary control of coupled reaction-advection-diffusion systems with spatially-varying coefficients. *IEEE Transactions on Automatic Control*, 62(4):2026–2033, 2016.

[91] R. Vazquez and M. Krstic. Explicit output-feedback boundary control of reaction-diffusion pdes on arbitrary-dimensional balls. *ESAIM: Control, Optimisation and Calculus of Variations*, 22(4):1078–1096, 2016.

[92] R. Vazquez, E. Schuster, and M. Krstic. Magnetohydrodynamic state estimation with boundary sensors. *Automatica*, 44(10):2517–2527, 2008.

[93] J. Wang and M. Krstic. Output feedback boundary control of a heat pde sandwiched between two odes. *IEEE Transactions on Automatic Control*, 64(11):4653–4660, 2019.

[94] J. Wang and M. Krstic. Delay-compensated control of sandwiched ode–pde–ode hyperbolic systems for oil drilling and disaster relief. *Automatica*, 120:109131, 2020.

[95] J. Wang and M. Krstic. Event-triggered output-feedback backstepping control of sandwich hyperbolic pde systems. *IEEE Transactions on Automatic Control*, 67(1):220–235, 2022.

[96] H. Yu and M. Krstic. Traffic congestion control for Aw-Rascle-Zhang model. *Automatica*, 100:38–51, 2019.

[97] H. Yu and M. Krstic. *Traffic Congestion Control by PDE Backstepping*. Springer, 2022.

[98] K. Zhang, S. Kakade, T. Basar, and L. Yang. Model-based multi-agent rl in zero-sum markov games with near-optimal sample complexity. *Advances in Neural Information Processing Systems*, 33:1166–1178, 2020.

[99] K. Zhang, Z. Yang, and T. Basar. Policy optimization provably converges to nash equilibria in zero-sum linear quadratic games. *Advances in Neural Information Processing Systems*, 32, 2019.

[100] F. Zhao, K. You, and T. Başar. Global convergence of policy gradient primal-dual methods for risk-constrained lqrs. *IEEE Transactions on Automatic Control*, 2023.