2023 62nd IEEE Conference on Decision and Control (CDC)
December 13-15, 2023. Marina Bay Sands, Singapore

Neural Operators for Hyperbolic PDE Backstepping Kernels

Luke Bhan, Yuanyuan Shi, and Miroslav Krstic

***Abstract*— We introduce a framework for eliminating the computation of controller gain functions in PDE control. We learn the nonlinear operator from the plant parameters to the control gains with a (deep) neural network. We provide closed-loop stability guarantees (global exponential) under an NN-approximation of the feedback gains. While, in the existing PDE backstepping, finding the gain kernel requires (one offline) solution to an integral equation, the neural operator (NO) approach we propose learns the mapping from the functional coefficients of the plant PDE to the kernel function by employing a sufficiently high number of offline numerical solutions to the kernel integral equation, for a large enough number of the PDE model's different functional coefficients. We prove the existence of a DeepONet approximation, with arbitrarily high accuracy, of the exact nonlinear continuous operator mapping PDE coefficient functions into gain functions. Once proven to exist, learning of the NO is standard, completed "once and for all" (never online) and the kernel integral equation doesn't need to be solved ever again, for any new functional coefficient not exceeding the magnitude of the functional coefficients used for training. Simulation illustrations are provided and code is available on github. This framework, eliminating real-time recomputation of gains, has the potential to be game changing for adaptive control of PDEs and gain scheduling control of nonlinear PDEs.**

I. Introduction

ML/AI has been a disruptive force in a wide class of engineering disciplines leading to questions about whether ML/AI will "takeover" model-based sciences such as physics or conventional control theory. Recently, a new framework has emerged [7], [8], [10], [11] which promises to unite the goals of physics and learning, rather than presenting learning as an alternative or substitute to first-principles physics. This framework, termed neural operators (NO), is formulated as learning mappings from function spaces into function spaces and has achieved success in PDEs with learnable solution/"flow" maps after enough simulations with different initial conditions.

a) Mappings of plant parameters to control gains and learning of those maps: It is worth asking what neural operators can contribute to control theory, namely to the design of controllers, observers, and online parameter estimators. In this work, we explore a first venture in this direction with the capability for future extensions and layout a blueprint to *learn* PDE control design and prove their stability.

Although learning nonlinear maps for various design problems for nonlinear ODEs is worth studying, we focus this initial work one step beyond, on a benchmark PDE control class. Particularly, we focus on an uncomplicated - but unstable - PDE control class. Our choice of basic PDE control is for pedagogical reasons - combining the operator learning with *PDE backstepping* is complex for even the simplest-looking PDE stabilization problems.

b) PDE backstepping control with the gain computation obviated using neural operators: Consider 1D hyperbolic partial integro-differential equation systems of the general form $v_t(x,t) = v_x(x,t) + \lambda(x)v(x,t) + g(x)v(0,t) + \int_0^x f(x,y)v(y,t)dy$ on the unit interval $x \in [0,1]$, which are transformable, using an invertible backstepping "pre-transformation" introduced in [1] into the simple PDE

$$u_t(x,t) = u_x(x,t) + \beta(x)u(0,t) \tag{1}$$
$$u(1,t) = U(t). \tag{2}$$

Our goal is the design of a PDE backstepping boundary control

$$U(t) = \int_0^1 k(1-y)u(y,t)dy. \tag{3}$$

Physically, (1) is a "transport process (from $x = 1$ towards $x = 0$) with recirculation" of the outlet variable $u(0,t)$. Recirculation causes instability when the coefficient $\beta(x)$ is positive and large. This instability is prevented by the backstepping boundary feedback (3) with the gain function $k(\cdot)$ as a kernel in the spatial integration of the measured state $u(y,t)$. (The full state does not need to be measured, as explained in Remark 1 at the end of Section IV.)

Backstepping produces the gain kernel k for a given β. We learn the nonlinear continuous mapping $\mathcal{K} : \beta \mapsto k$ and once $\mathcal{K}$ is learned, the partial differential or integral equation does *not* need to be recomputed for a new β. Instead, for a new β, finding k is simply a "function evaluation" of the learned mapping $\mathcal{K}$. This benefits adaptive control where, at each timestep, the gain estimate $\hat{k}$ needs to be calculated for a new parameter update $\hat{\beta}$ and in gain scheduling for nonlinear PDEs where the gain must be recomputed at each current state.

Naturally, one can just learn the mapping $\mathcal{K}$ and stop. However, in this work, we extend our analysis to investigate whether the NN approximation of the gains $\hat{k}$ will result in a stable PDE. We find that with a large enough data set of solved pairs (β_i, k_i), and a large enough trained (deep) NN, closed-loop stability is guaranteed for a new β, not in the training set.

c) Neural operator literature—a brief summary: Neural operators are NN-parameterized maps for learning relationships between function spaces. They originally gained popularity due to their success in mapping PDE solutions

This work was supported by NSF Grants ECCS-2151525 and ECCS-2210315 as well as AFOSR FA9550-22-1-0265

The authors are with the University of California, San Diego, USA, lbhan@ucsd.edu, yyshi@eng.ucsd.edu, krstic@ucsd.edu

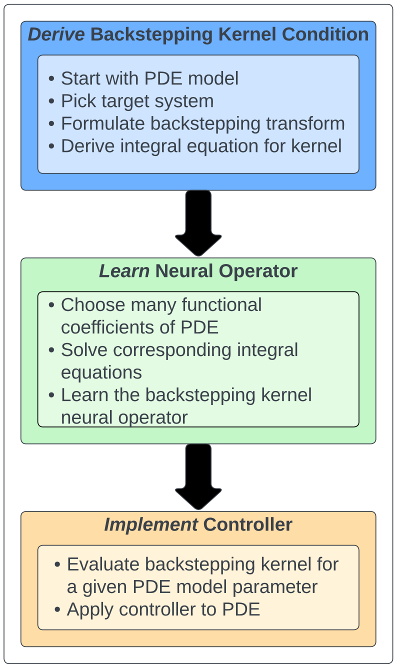

Fig. 1. An algorithmic representation of our design paradigm of employing neural operators in boundary control of PDEs. Three major step clusters are performed: (1) derivation of the integral equations for the backstepping kernels, performed only once; (2) learning of the mapping from the plant parameter 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 [5]. In this paper, the task in the middle box is introduced and stability guarantees for the task in the bottom box are provided.

while remaining discretization-invariant. Generally, nonlinear operators consist of three components: an encoder, an approximator, and a reconstructor [6]. 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 NNs, but can take many different forms. More details can be found in [3], [8]–[10], [12]–[15]

d) Paper outline and contributions: After a brief introduction to the backstepping design in Section II, for system (1), (2), in Section III we prove that the backstepping kernel operator is locally Lipschitz, between the spaces of continuous functions, with which we satisfy a sufficient condition for the existence of a neural operator approximation of a nonlinear operator to arbitrarily high accuracy—stated at the section's end in a formal result and illustrated with an example of approximating the operator $k = \mathcal{K}(\beta)$. In Section IV we present our main results: the closed-loop stabilization (not merely practical but exponential) with a DeepONet-approximated backstepping gain kernel function. In Section V we present simulations illustrate stabilization under DeepONet-approximated gains.

Lastly, we note the stabilization results also hold for any other neural operators with a universal approximation property (shown for LOCA [3] and for FNO on the periodic domain [4]).

e) Notation: We denote convolution operations as

$$(a * b)(x) = \int_0^x a(x-y)b(y)dy \tag{4}$$

In the sequel, we suppresses the arguments x and t wherever clear from the context. For instance, we write (1), (2) compactly as $u_t = u_x + \beta u(0)$ and $u(1) = U$, where, from the context, the boundary values $u(0), u(1)$ depend on t as well.

II. Backstepping Design for a Transport PDE with 'Recirculation'

Consider the PDE system (1), (2). We employ the following backstepping transformation:

$$w = u - k * u, \tag{5}$$

i.e., $w(x,t) = u(x,t) - \int_0^x k(x-y)u(y,t)dy$, to convert the plant into the target system

$$w_t = w_x \tag{6}$$

$$w(1) = 0 \tag{7}$$

with the help of feedback

$$U = (k * u)(1), \tag{8}$$

namely, $U(t) = \int_0^1 k(1-y)u(y,t)dy$. To yield the target system, k must satisfy the integral/convolution equation

$$k(x) = -\beta(x) + \int_0^x \beta(x-y)k(y)dy \tag{9}$$

for $x \in [0,1]$. Note that, while this integral equation is linear in k for a given β, the mapping from β to k is actually nonlinear, due to the product in the convolution of β with k.

III. Accuracy of Approximation of Backstepping Kernel Operator with DeepONet

An n-layer NN $f^{\mathcal{N}} : \mathbb{R}^{d_1} \to \mathbb{R}^{d_n}$ is given by

$$f^{\mathcal{N}}(x,\theta) := (l_n \circ l_{n-1} \circ ... \circ l_2 \circ l_1)(x,\theta) \tag{10}$$

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

$$l_{i+1}(l_i, \theta_{i+1}) := \sigma(W_{i+1} l_i + b_{i+1}), \quad i = 1, \ldots, n-1 \tag{11}$$

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.

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

$$\mathcal{G}_{\mathbb{N}}(\mathbf{u}_m)(y) = \sum_{k=1}^{p} g^{\mathcal{N}}(\mathbf{u}_m; \vartheta^{(k)}) f^{\mathcal{N}}(y; \theta^{(k)}) \tag{12}$$

where $\mathcal{U}, \mathcal{V}$ are function spaces of continuous functions $u \in \mathcal{U}, v \in \mathcal{V}$. $\mathbf{u}_m$ is the evaluation of function u at points $x_i = x_1, ..., x_m$, 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 $g^{\mathcal{N}}$, $f^{\mathcal{N}}$ are NNs termed branch and trunk networks. Note, $g^{\mathcal{N}}$ and $f^{\mathcal{N}}$ are not limited to feedforward NNs 10, but can also be of convolutional or recurrent.

Theorem 1: (DeepONet universal approximation theorem [2, 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 $\mathcal{U} : X \to U \subset \mathbb{R}^{d_u}$ and $\mathcal{V} : Y \to V \subset \mathbb{R}^{d_v}$ be sets of continuous functions $u(x)$ and $v(y)$, respectively. Let $\mathcal{U}$ be also compact. Assume the operator $\mathcal{G} : \mathcal{U} \to \mathcal{V}$ is continuous. Then, for all $\epsilon > 0$, 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 $f^{\mathcal{N}}(\cdot; \theta^{(k)}), g^{\mathcal{N}}(\cdot; \vartheta^{(k)}), k = 1, \ldots, p$, 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

$$|\mathcal{G}(u)(y) - \mathcal{G}_{\mathbb{N}}(\mathbf{u}_m)(y)| < \epsilon \tag{13}$$

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

Definition 1: (backstepping kernel operator). A mapping $\mathcal{K} : \beta \mapsto k$ of $C^0[0,1]$ into itself, where $k = \mathcal{K}(\beta)$ satisfies

$$\mathcal{K}(\beta) = -\beta + \beta * \mathcal{K}(\beta), \tag{14}$$

namely, in the Laplace transform notation,

$$k = \mathcal{K}(\beta) := \mathscr{L}^{-1}\left\{\frac{\mathscr{L}\{\beta\}}{\mathscr{L}\{\beta\} - 1}\right\} \tag{15}$$

is referred to as the *backstepping kernel operator*.

Lemma 1: (Lipschitzness of backstepping kernel operator $\mathcal{K}$). The kernel operator $\mathcal{K} : \beta \mapsto k$ in Definition 1 is Lipschitz. Specifically, for any $B > 0$ the operator $\mathcal{K}$ satisfies

$$||\mathcal{K}(\beta_1) - \mathcal{K}(\beta_2)||_\infty \leq C||\beta_1 - \beta_2||_\infty \tag{16}$$

with the Lipschitz constant

$$C = \mathrm{e}^{3B} \tag{17}$$

for any pair of functions (β_1, β_2) such that $\|\beta_1\|_\infty, \|\beta_2\|_\infty \leq B$, where $\|\cdot\|_\infty$ is the supremum norm over the argument of β and k.

Proof: Start with the iteration $k^0 = -\beta, k^{n+1} = k^0 + \beta * k^n,\ n \geq 0$ and consider the iteration

$$\Delta k^{n+1} = \beta * \Delta k^n, \qquad \Delta k^0 = k^0 = -\beta \tag{18}$$

for the difference $\Delta k^n = k^n - k^{n-1}$, which sums to

$$k = \sum_{n=1}^{\infty} \Delta k^n. \tag{19}$$

Next, for $\bar{\beta} = \|\beta\|_\infty$ and all $x \in [0,1]$,

$$|\Delta k^n(x)| \leq \frac{\bar{\beta}^{n+1} x^n}{n!}, \tag{20}$$

which is established by induction by postulating $\left|\Delta k^{n-1}(x)\right| \leq \frac{\bar{\beta}^n x^{n-1}}{(n-1)!}$ and by computing, from (18),

$$\begin{aligned} |\Delta k^n(x)| &= \left|\int_0^x \beta(x-y)\Delta k^{n-1}(y)dy\right| \\ &\leq \bar{\beta}\int_0^x \frac{\bar{\beta}^n y^{n-1}}{(n-1)!}|dy \leq \frac{\bar{\beta}^{n+1} x^n}{n!}. \end{aligned} \tag{21}$$

And then, (20) and (19) yield

$$|k(x)| \leq \bar{\beta}\mathrm{e}^{\bar{\beta}x}. \tag{22}$$

Next, for $k_1 = \mathcal{K}(\beta_1)$ and $k_2 = \mathcal{K}(\beta_2)$ it is easily verified that

$$k_1 - k_2 = \beta_1 * (k_1 - k_2) - \delta\beta + \delta\beta * k_2 \tag{23}$$

where $\delta\beta = \beta_1 - \beta_2$. Define the iteration

$$\delta k^{n+1} = \beta_1 * \delta k^n \tag{24}$$

$$\delta k^0 = -\delta\beta + \delta\beta * k_2 \tag{25}$$

which verifies $k_1 - k_2 = \sum_{n=1}^{\infty} \delta k^n$. Noting that (22) ensures that $k_2 = \mathcal{K}(\beta_2)$ verifies $|k_2(x)| \leq \bar{\beta}_2 \mathrm{e}^{\bar{\beta}_2 x}$, from (25),

$$|\delta k^0(x)| \leq \left(1 + \bar{\beta}_2\mathrm{e}^{\bar{\beta}_2 x}\right)\overline{\delta\beta} \leq \mu_2\overline{\delta\beta} \tag{26}$$

where $\mu_2 := 1 + \bar{\beta}_2\mathrm{e}^{\bar{\beta}_2}$ and $\overline{\delta\beta} = \|\beta_1 - \beta_2\|_\infty$, it can be shown by induction, by mimicking the chain of inequalities (21), that, for all $x \in [0,1]$,

$$|\delta k^n(x)| \leq \mu_2\overline{\delta\beta}\frac{\bar{\beta}_1^n x^n}{n!} \tag{27}$$

and therefore it follows that, for all $x \in [0,1]$,

$$\begin{aligned} |k_1(x) - k_2(x)| &\leq \left(1 + \bar{\beta}_2\mathrm{e}^{\bar{\beta}_2}\right)\mathrm{e}^{\bar{\beta}_1 x}\|\beta_1 - \beta_2\|_\infty \\ &\leq \mathrm{e}^{3B}\|\beta_1 - \beta_2\|_\infty. \end{aligned} \tag{28}$$

Hence, local Lipschitzness is proven with (17). ■

Corollary 1: (to Theorem 1). Consider the backstepping kernel operator $\mathcal{K}$ in Definition 1. For all $B > 0$ and $\epsilon > 0$, there exist $p^*(B,\epsilon), m^*(B,\epsilon) \in \mathbb{N}$, with an increasing dependence on B and $1/\epsilon$, such that for each $p \geq p^*$ and $m \geq m^*$ there exist $\theta^{(k)}, \vartheta^{(k)}$, neural networks $f^{\mathcal{N}}(\cdot; \theta^{(k)}), g^{\mathcal{N}}(\cdot; \vartheta^{(k)}), k = 1, \ldots, p$, and $x_j \in K_1, j = 1, \ldots, m$, with corresponding $\beta_m = (\beta(x_1), \beta(x_2), \cdots, \beta(x_m))^{\mathrm{T}}$, such that

$$|\mathcal{K}(\beta)(x) - \mathcal{K}_{\mathbb{N}}(\beta_m)(x)| < \epsilon \tag{29}$$

holds for all Lipschitz β with the property that $\|\beta\|_\infty \leq B$.

So the backstepping kernel is approximable, qualitatively, but how many neurons and how much data are needed for a given ϵ? We recall a result on the minimum-sized DeepONet.

Proposition 1: (DeepONet size for kernel operator approximation [2, Theorem 3.3 and Remark 3.4]). If the kernel operator defined in (14) is Lipschitz (or at least Hölder) continuous, a DeepONet that approximates it to a *required* error tolerance $\epsilon > 0$ indicated by (29) employs the number of data point evaluations for β on the order of

$$m \sim \epsilon^{-1}, \tag{30}$$

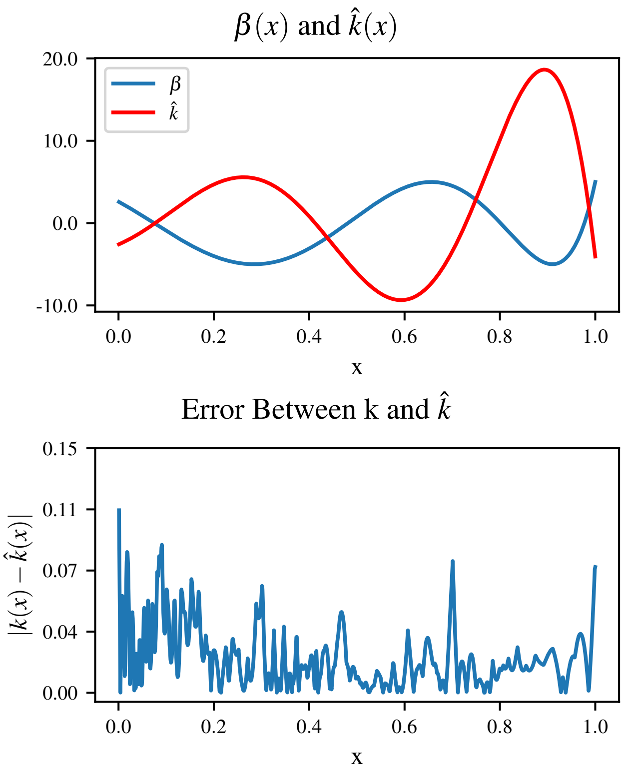

Fig. 2. Examples of β, $\hat{k}$ for Chebyshev polynomials defined as $\beta = 6\cos(\gamma\cos\ ^{1}(x))$ with $\gamma = 7.35$. The γ parameter controls the wave frequency of β and therefore affects the resulting kernel. Additionally, the DeepONet absolute approximation error of $\hat{k}$ and k is shown.

the number of basis components in the interpolation when reconstructing into $C^0[0,1]$ on the order of

$$p \sim \epsilon^{-\frac{1}{2}}, \tag{31}$$

the numbers of layers $L_{g^{\mathcal{N}}}$ in the branch network and of neurons $N_{g^{\mathcal{N}}}$ in each layer of the branch network on the order given, respectively, by

$$N_{g^{\mathcal{N}}} \cdot L_{g^{\mathcal{N}}} \sim \left(\frac{1}{\epsilon}\right)^{\frac{1}{\epsilon}}, \tag{32}$$

and the total size of the trunk network on the order of

$$|\theta^{(k)}| \sim \left(\frac{3}{2}\log\frac{1}{\epsilon}\right)^2. \tag{33}$$

Example 1: In Figure 2 we present two examples of approximation of k using a DeepONet approximation of $\mathcal{K}(\beta)$ for given β_1 and β_2, which are taken as Chebyshev polynomials $\beta(x) = 6\cos(\gamma\cos^{-1}(x))$. They are trained on approximating kernels from 900 samples with $\gamma \in$ uniform$[2,8]$.

IV. Stability under Kernel Approximation with DeepONet

For our stability study under an approximate (imperfect) kernel, we begin with a derivation of the target PDE system under a backstepping transformation employing a DeepONet approximation of the backstepping kernel.

For a given β, let $\hat{k} = \hat{\mathcal{K}}(\beta)$, where $\hat{\mathcal{K}} = \mathcal{K}_{\mathbb{N}}$, denote an NO approximation of the exact backstepping kernel k whose existence is established in Corollary 1 for DeepONet. Let

$$\tilde{k} = k - \hat{k} \tag{34}$$

denote the approximation error. Finally, let the backstepping transformation with the approximate kernel $\hat{k}$ be

$$\hat{w} = u - \hat{k} * u. \tag{35}$$

With routine calculations, employing the approximate backstepping transformation and the feedback

$$U = (\hat{k} * u)(1) \tag{36}$$

we arrive at the target system

$$\hat{w}_t = \hat{w}_x + \delta\hat{w}(0) \tag{37}$$
$$\hat{w}(1) = 0, \tag{38}$$

where the function $\delta(x)$ is defined as

$$\delta = -\tilde{k} + \beta * \tilde{k}. \tag{39}$$

Next, we proceed with a Lyapunov analysis.

Lemma 2: (a Lyapunov estimate). Given arbitrarily large $B > 0$, for all Lipschitz β with $\|\beta\|_\infty \le B$, and for all neural operators $\hat{\mathcal{K}}$ with $\epsilon \in (0, \epsilon^*)$, where

$$\epsilon^*(B) = \frac{c\mathrm{e}^{-c/2}}{1+B} \tag{40}$$

the Lyapunov functional

$$V(t) = \int_0^1 \mathrm{e}^{cx}\hat{w}^2(x,t)dx, \qquad c > 0. \tag{41}$$

satisfies the following estimate along the solutions of the target system (37), (38),

$$V(t) \le V(0)\mathrm{e}^{-c^* t}, \tag{42}$$

for

$$c^* = c - \frac{e^c}{c}\epsilon^2(1+B)^2 > 0. \tag{43}$$

The accuracy required of the NO $\hat{\mathcal{K}}$, and given by (40), is maximized with $c = 2$ and has the value $\epsilon^*(B) = \frac{2}{\mathrm{e}(1+B)}$.

Proof: Several steps of calculation (chain rule, substitution, integration by parts) result in

$$\begin{aligned} \dot{V} &= -\hat{w}^2(0) - c\int_0^1 \mathrm{e}^{cx}\hat{w}^2(x,t)dx \\ &\quad + \hat{w}(0)\int_0^1 \delta(x)\mathrm{e}^{cx}\hat{w}(x)dx \\ &\le -\frac{1}{2}w^2(0) - c\int_0^1 \mathrm{e}^{cx}\hat{w}^2(x,t)dx \\ &\quad + \left(\int_0^1 \delta(x)\mathrm{e}^{cx}\hat{w}(x)dx\right)^2 \end{aligned} \tag{44}$$

With the Cauchy-Schwartz inequality

$$\begin{aligned} &\left(\int_0^1 \delta(x)\mathrm{e}^{cx}\hat{w}(x)dx\right)^2 \\ &\le \int_0^1 \delta^2(x)\mathrm{e}^{cx}dx \int_0^1 \mathrm{e}^{cx}\hat{w}(x)^2dx \end{aligned} \tag{45}$$

we get

$$\dot{V} \leq -\frac{1}{2}w^2(0) - \left(c - \int_0^1 \delta^2(x)e^{cx}dx\right)V \quad (46)$$

The function δ in (39) is bounded by $|\delta(x)| \leq (1+||\beta||_\infty)\,||\tilde{k}||_\infty$ which, in turn, using (29), yields

$$|\delta(x)| \leq (1+\bar{\beta})\epsilon =: \bar{\delta}. \quad (47)$$

Then, substituting this into (37), we obtain:

$$\begin{aligned}\dot{V} &\leq -\frac{1}{2}w^2(0) - \left(c - \epsilon^2\left(1+\bar{\beta}\right)^2 \int_0^1 e^{cx}dx\right)V \\ &\leq -\frac{1}{2}w^2(0) - \left(c - \frac{e^c}{c}\epsilon^2\left(1+\bar{\beta}\right)^2\right)V \\ &\leq -\frac{1}{2}w^2(0) - \left(c - \frac{e^c}{c}\epsilon^2\left(1+B\right)^2\right)V \quad (48)\end{aligned}$$

For $0 \leq \epsilon \leq \epsilon^*$, where ϵ^* is defined in (40), we have

$$\dot{V} \leq -\frac{1}{2}w^2(0) - c^*V \quad (49)$$

for some $c^* > 0$ in (43). ■

The size of the NO and of the dataset needs to increase with $\bar{\beta}$, i.e., with the potential instability in the open-loop system.

Lemma 3: (bound on inverse approximate kernel). The kernel $\hat{l}$ of the inverse to the backstepping transformation (35),

$$u = \hat{w} + \hat{l} * \hat{w}, \quad (50)$$

satisfies, for all $x \in [0,1]$, the estimate

$$|\hat{l}(x)| \leq \left(\bar{\beta} + (1+\bar{\beta})\epsilon\right) e^{(1+\bar{\beta})\epsilon x}. \quad (51)$$

Proof: It is easily shown that $\hat{l}$ obeys the integral equation

$$\hat{l} = -\beta + \delta + \delta * \hat{l}. \quad (52)$$

Using the successive approximation approach, we get that the following bound holds for all $x \in [0,1]$:

$$|\hat{l}(x)| \leq \left(\bar{\beta} + \bar{\delta}\right) e^{\bar{\delta}x}. \quad (53)$$

With (47), we get (51). ■

Theorem 2: (Closed-loop stability robust to DeepONet approximation of backstepping kernel). Let $B > 0$ be arbitrarily large and consider the closed-loop system consisting of (1), (2) with any Lipschitz β such that $\|\beta\|_\infty \leq B$, and the feedback (36) with the NO gain kernel $\hat{k} = \hat{\mathcal{K}}(\beta)$ of arbitrary desired accuracy of approximation $\epsilon \in (0, \epsilon^*)$ in relation to the exact backstepping kernel k, where $\epsilon^*(B)$ is defined in (40). This closed-loop system obeys the exponential stability estimate

$$\|u(t)\| \leq M e^{-c^*t/2}\|u(0)\|, \qquad \forall t \geq 0 \quad (54)$$

with the overshoot coefficient

$$M = \left(1 + \left(\bar{\beta} + (1+\bar{\beta})\epsilon\right) e^{(1+\bar{\beta})\epsilon}\right)\left(1 + \bar{\beta}e^{\bar{\beta}}\right) e^{c/2}. \quad (55)$$

Proof: First, we note that Lemma 2 satisfies

$$\frac{1}{\left(1+\|\hat{l}\|_\infty\right)^2}\|u\|^2 \leq V \leq e^c\left(1+\|\hat{k}\|_\infty\right)^2\|u\|^2. \quad (56)$$

Since, by Lemma 2, $V(t) \leq V(0)e^{-c^*t}$, we get, for all $t \geq 0$,

$$\begin{aligned}\|u(t)\| \leq &\left(1+\|\hat{l}\|_\infty\right)\left(1+\|\hat{k}\|_\infty\right)e^{c/2} \\ &\times e^{-c^*t/2}\|u(0)\|. \quad (57)\end{aligned}$$

Then, noting, with Theorem 1, (22), and Lemma 3 that

$$\begin{aligned}\|\hat{k}\|_\infty &\leq \|k\|_\infty + \epsilon \leq \bar{\beta}e^{\bar{\beta}} + \epsilon \quad (58)\\ \|\hat{l}\|_\infty &\leq \left(\bar{\beta} + (1+\bar{\beta})\epsilon\right) e^{(1+\bar{\beta})\epsilon} \quad (59)\end{aligned}$$

we finally arrive at the exponential stability estimate (54). ■

Remark 1: Full-state measurement $u(x,t)$ is employed in the feedback law (36) but can be avoided by employing only the measurement of the outlet signal, $u(0,t)$, from which the full state $u(x,t)$ is observable, the observer

$$\begin{aligned}\breve{u}_t &= \breve{u}_x + \beta u(0) \quad (60)\\ \hat{u}(1) &= U \quad (61)\end{aligned}$$

and the observer-based controller

$$U = (\hat{k} * \breve{u})(1), \quad (62)$$

which can avoid solving the PDE (60), (61) online by employing its explicit solution as an arbitrary function $\breve{u}(x,t) = \breve{u}_0(x)$ for $t + x \in [0,1)$ and

$$\breve{u}(x,t) = U(t+x-1) + \int_{t+x-1}^{t} \beta(t+x-\tau)u(0,\tau)d\tau \quad (63)$$

for $t + x \geq 1$. A closed-loop stability result as in Theorem 2 can be established for this observer-based controller.

V. Simulations: Stabilization with NO-Approximated Gain Kernel $\beta \mapsto \mathcal{K}(\beta)$

Continuing Example 1, in Figure 3 we show that the system is open-loop unstable for both βs and we present tests with the learned kernels in closed-loop simulations up to $t = 2$. In both cases, the PDE settles (nearly perfectly) by $t = 1$, as expected from the target system with the perfect kernel k. The small ripple in the right simulation is due to the use of the approximated kernel $\hat{k}$. The simulations confirm the theoretical guarantee that an NO-approximated kernel can successfully emulate a backstepping kernel while maintaining stability.

The NO architecture in $\hat{\mathcal{K}}$ consists of about 680 thousand parameters with a training time of 1 minute (using an Nvidia RTX 3090Ti GPU) on a dataset of 900 different β defined as the Chebyshev polynomials $\beta = 6\cos(\gamma\cos^{-1}(x))$ where $\gamma \sim$ uniform(2, 10). We choose β of this form due to the rich set of PDEs and kernel functions constructed by varying only a single parameter. The resulting training relative L_2 error $4e-3$ and the testing relative L_2 loss on 100 instances sampled from the same distribution was $5e-3$. If a wider distribution of γ is chosen, the mapping can be learned but requires both a larger network and more data for the same accuracy.

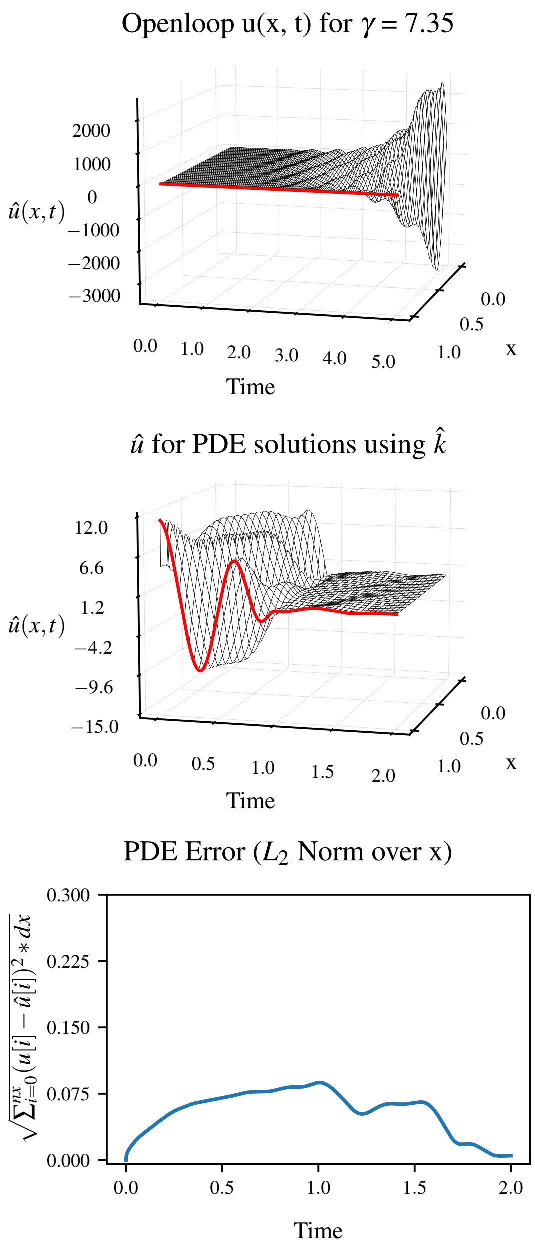

Fig. 3. Top row showcases open-loop instability for the recirculation function β that are the same as in Fig. 2, with $\gamma = 7.35$. Additionally, the bottom two rows highlight examples of PDE closed-loop state response and errors between the response with "perfect gain" k and "approximate gain" $\hat{k}$. β corresponds to the same values in Figure 2.

VI. Conclusion

We introduce a novel framework for approximating solution maps for the feedback gain functions k (9) in *control of PDEs*. We provide the guarantees that (i) any desired level of accuracy of NO approximation of the backstepping gain kernel is achieved for any β that satisfies $\|\beta\|_\infty \leq B$ for arbitrarily large given $B > 0$, and (ii) the PDE is stabilized with an NO-approximated gain kernel for any $\|\beta\|_\infty \leq B$.

For a given $B > 0$ and any chosen positive $\epsilon < \epsilon^*(B)$, the determination of the NO approximate operator $\hat{\mathcal{K}}(\cdot)$ is done offline, once only, and such a $\hat{\mathcal{K}}(\cdot)$, which depends on B and ϵ, is usable "forever," so to speak, for any recirculation kernel that does not violate $\|\beta\|_\infty \leq B$.

Our achievement of global exponential stability (not "practical"/approximate, but with an actual convergence of the state to zero) relies crucially—in each of the lemmas and theorems that we state—on the theoretical steps from the PDE backstepping toolkit (backstepping transform, target system, integral equation for kernel, successive infinite-series approximation, Lyapunov analysis). It is only by assigning the NO a service role in an otherwise model-based design that stability is assured wheras in other approaches such as RL [16], assurance is absent.

References

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

[2] B. Deng, Y. Shin, L. Lu, Z. Zhang, and G. E. Karniadakis. Approximation rates of deeponets for learning operators arising from advection–diffusion equations. *Neural Networks*, 153:411–426, 2022.

[3] 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.

[4] N. Kovachki, S. Lanthaler, and S. Mishra. On universal approximation and error bounds for fourier neural operators. *The Journal of Machine Learning Research*, 22(1):13237–13312, 2021.

[5] 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.

[6] 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.

[7] 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.

[8] 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.

[9] Z. Li, M. Liu-Schiaffini, N. Kovachki, B. Liu, K. Azizzadenesheli, K. Bhattacharya, A. Stuart, and A. Anandkumar. Learning dissipative dynamics in chaotic systems, 2021.

[10] 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.

[11] 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.

[12] E. Pickering, S. Guth, G. E. Karniadakis, and T. P. Sapsis. Discovering and forecasting extreme events via active learning in neural operators. *Nature Computational Science*, 2(12):823–833, Dec 2022.

[13] 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.

[14] 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.

[15] S. Wang, H. Wang, and P. Perdikaris. Learning the solution operator of parametric partial differential equations with physics-informed deeponets. *Science Advances*, 7(40):eabi8605, 2021.

[16] H. Yu, S. Park, A. Bayen, S. Moura, and M. Krstic. Reinforcement learning versus pde backstepping and pi control for congested freeway traffic. *IEEE Trans. Control Systems Technology*, 30:1595–1611, 2022.