Neural operators for PDE backstepping control of first-order hyperbolic PIDE with recycle and delay

Jie Qi\textsuperscript{a,b,∗}, Jing Zhang\textsuperscript{c}, Miroslav Krstic\textsuperscript{c}

\textsuperscript{a} College of Information Science and Technology, Donghua University, Shanghai, 201620, China
\textsuperscript{b} Engineering Research Center of Digitized Textile and Fashion Technology Ministry of Education, Donghua University, Shanghai, 201620, China
\textsuperscript{c} Department of Mechanical and Aerospace Engineering, University of California San Diego, La Jolla, CA 92093-0411, USA

\section*{A R T I C L E I N F O}

Dataset link: https://github.com/JingZhang-JZ/NO_hyperbolic_delay.git

Keywords:
First-order hyperbolic partial integral differential equation
PDE backstepping
DeepONet
Delays
Learning-based control

\section*{A B S T R A C T}

The recently introduced DeepONet operator-learning framework for PDE control is extended from the results for basic hyperbolic and parabolic PDEs to an advanced hyperbolic class that involves delays on both the state and the system output or input. The PDE backstepping design produces gain functions that are outputs of a nonlinear operator, mapping functions on a spatial domain into functions on a spatial domain, and where this gain-generating operator's inputs are the PDE's coefficients. The operator is approximated with a DeepONet neural network to a degree of accuracy that is provably arbitrarily tight. Once we produce this approximation-theoretic result in infinite dimension, with it we establish stability in closed loop under feedback that employs approximate gains. In addition to supplying such results under full-state feedback, we also develop DeepONet-approximated observers and output-feedback laws and prove their own stabilizing properties under neural operator approximations. With numerical simulations we illustrate the theoretical results and quantify the numerical effort savings, which are of two orders of magnitude, thanks to replacing the numerical PDE solving with the DeepONet.

\section*{1. Introduction}

A neural operator-based method has been developed for the pre-training the backstepping methodology, offline and once and for all, for certain entire classes of PDEs so that the implementation of the controller to any specific PDE within the class is nothing more than a function evaluation of a neural network that produces the controller gains based on the specific plant coefficients of the PDE being controlled. The method was initially introduced in [1,2], laying the groundwork for combination of neural network-based method and classical control theory, thus enabling more extensive developments in this field. In this paper, we extend this method to a broader and more advanced class of hyperbolic partial integro-differential systems, which involve delays on the state and the output or input.

\subsection*{1.1. The broader context of learning-based and data-driven control}

Recently, learning-based control approaches have attracted great attention due to their leveraging of capabilities of deep neural networks. Some of these approaches learn control strategies from data without explicit knowledge of system dynamics, and some are able to deal with uncertainties and disturbances. Stability and robustness can be proven with some of these control methods [3], which builds trust for their use in practice. Progress has taken place with learning-based model predictive control (MPC) for uncertain models [4,5], Lyapunov functional based control design [6,7], reinforcement learning (RL) based linear quadratic regulator [8,9], and other methods. RL has also been applied to PID tuning [10,11], with a notable use in [12], where a deep RL-based PID tuning method is proposed and experimented on the physical two-tank system without prior pre-training. For the risks that might arise during the RL based control process, recently, safe reinforcement learning has emerged as a new research focus, see e.g., [13–15].

Learning-based control in unmanned systems is pursued in [16–19]. For example in control of the unmanned aerial vehicles (UAVs), a dual-stream Actor–Critic network structure is applied to extract environmental features, enabling UAVs to safely navigate in environments with multiple obstacles [20]. Data-driven control methods extract the hidden patterns from a large amount of data, which improves control performance in uncertain environment. In [21], a deep network
learning-based trajectory tracking controller, called Neural-Fly, is proposed for drones’ agile flight in rapidly changing strong winds. Transfer learning also used to leverage control strategies and models that have already been learned to accelerate the learning and adaptation process for new tasks, e.g., [18,22,23].

1.2. Learning-enhanced PDE control

Many engineering problems are spatio-temporal processes, often modeled by partial differential equations (PDEs) instead of ordinary differential equations (ODEs), such as plug flow reactor [24], traffic flow [25,26], hydraulics and river dynamics [27], pipeline networks [28,29], melt spinning processes [30], flexible robots [31], flexible satellite [32], tokamaks [33] and so on.

PDE backstepping has been particularly effective in the stabilization of PDEs. Since this paper is focused on a hyperbolic partial integro-differential equation (PIDE) class, we mention only a few designs for hyperbolic systems here. A design for a single hyperbolic PDE was introduced in [34]. A pair of coupled hyperbolic PDEs was stabilized next, with a single boundary input in [35]. An extension to \( n + 1 \) hyperbolic PDEs with a single input was introduced in [36], an extension to cascades with ODEs in [37], an extension to “sandwiched” ODE–PDE–ODE systems in [38,39], and redesigns robust to delays in [40,41].

Since the dynamics of the PDE systems are defined in infinite-dimensional function spaces, the gains in the PDE control systems (feedback controllers, observers, identifiers) are not vectors or matrices but functions of spatial arguments. When the coefficients of the system are spatially-varying, the equations governing the control gain kernels usually cannot be solved explicitly, as they are complex PDEs and need to be solved numerically, e.g. [36,42–44]. When any coefficient changes, the control gain PDEs need to be re-solved, which is burdensome even if performed offline and once, let alone if it needs to be performed repeatedly in real time, in the context of adaptive control or gain scheduling.

Operator learning refers to the learning of an infinite-dimensional mapping operator by means of deep neural network. It is of interest to find a neural network (NN) which learns control gain operators from a large set of previously offline-solved control design problems for a sample set of PDEs in a certain class. For example, [45] utilized the Fourier Neural Operator to address the optimal Dirichlet boundary control problem in nonlinear optics. A robust framework employing an operator learning technique for such problems with PDEs constraints is provided in [46]. Furthermore, [47] demonstrates the application of DeepONet in learning the relationship between liquid pressure and bubble generation, thereby validating the efficacy and precision of neural operators (NO) in predicting the dynamics of multi-rate bubble growth.

The DeepONet framework [1,2,48] is an efficient method for PDE control, because it not only speeds up computation, e.g., on the order of magnitude of \( 10^3 \) time [2], as compared to solving for the control gains numerically, but also provides a methodology for stability analysis. The DeepONet [49] consists of two sub-networks, i.e., branch net and trunk net. The branch net encodes the discrete input function space and the trunk encodes the domain of the output functions. The combination of branch and trunk nets improves the generalization and efficiency of operators learning of the DeepONet, so that it realizes the reduction of infinite-dimensional functions from a relatively small number of datasets [50], which brings new insights for the learning based control. Furthermore, the universal approximation theorem [49–51], which states that a nonlinear continuous operator can be approximated by an appropriate DeepONet with any given approximation error, provides the basis for the rigorous stability analysis of the closed-loop system under the neural operator controller.

This offline learning PDE control design framework was pioneered in [1]. Among the PDE control design approaches, PDE backstepping was used, due to its non-reliance on model reduction and its avoidance of numerically daunting operator Riccati equations. Among the neural operator methods, the DeepONet [49,50] approach was employed, due to its availability of universal approximation theorem in infinite dimension. Closed-loop stability is guaranteed under the offline trained NN-approximation of the feedback gains. Paper [2] extends this framework from first-order hyperbolic PDEs to a more complex class of parabolic PDEs whose kernels are governed by second-order PDEs, raising the difficulty for solving such PDEs and for proving the sufficient smoothness of their solutions, so that the NOapproximations have guarantee of sufficient accuracy. Furthermore, an operator learning framework for accelerating nonlinear adaptive control is proposed in [52], where three operators are trained, namely parameter identifier operator, controller gain operator, and control operator.

1.3. Results, contributions, and organization of the paper

In this paper, we employ the DeepONet framework to learn the control kernel functions and the observer gains for the output feedback of a delayed first-order hyperbolic partial integro-differential equation (PIDE) system. Due to the system incorporating state and measurement or actuation delays, two transport PDEs are introduced to represent the delayed states, thus forming a hyperbolic PDEs cascade system. Applying the backstepping transformation, we derive a set of coupled PDEs that three backstepping kernels should satisfy, the solution of which can only be obtained numerically. Hence, three DeepONets are trained to approximate the three kernel functions from the numerical solutions. Once the neural operators are trained from data, the kernel equations do not need to be solved numerically again for new functional coefficients and new delays.

We use the universal approximation theorem to prove the existence of DeepONet approximations, with an arbitrary accuracy, of the exact continuous operators mapping the delay and the system coefficient functions into kernel functions. Based on the approximation result, we provide a state-feedback stability guarantee under neural operator kernels by using a Lyapunov functional.

We incorporate a “dead-time” into our PIDE model. Dead-time can represent either actuation or sensing delay—when the full state is unmeasured, the delay can be shifted between the input and the output. Without loss of generality, we locate the delay at the output/measurement. In such an architecture, control requires the design of an observer for the unmeasured state. Due to the delayed measurement, the backstepping transformation for the observer design and analysis contains four kernels, which determine two observer gains. We use two DeepONets to learn the observer gains directly, instead of the four kernel functions.

The observer with gains produced by neural operators is proved to converge to the actual states. Moreover, we prove the stability of the output feedback system under the neural gains through constructing a new Lyapunov functional. Within the proof, we combine the system under prescribed stabilizing controller with the feedback of the estimated states and the observer error system to establish the exponentially stability, thus verifying the separation principle. We demonstrate the theoretical results with numerical tests and the corresponding code is available on github.

The paper’s main contribution is the following:

- Unlike the two inaugural papers [1,2], this paper considers a PDE control problem subject to delays. The delayed system brings new challenges of dealing with multi-kernel coupled PDEs, for which we train three NNs to approximate three control kernels and two NNs to approximate two observer gains.

The paper’s additional contributions are:

- We combine the observer error system with the closed-loop system under the estimated state feedback to establish the exponentially stability, which verify the separation principle under the DeepONet learned output feedback controller.
We train two DeepONets to approximate the two observer gains instead of four kernel functions, which cuts the offline training computation cost in half.

The paper is organized as follows. In Section 2, we summarize the key steps and related conclusions for designing state feedback, observer design, and output feedback controllers using backstepping methods. Corresponding to the Backstepping theoretical results, Section 3 gives the DeepONet-based design and stability analysis for state feedback, observer and output feedback controller. In particular, we prove the operators are Lipschitz continuous and provide the existence of DeepONet approximations of the operators to any given accuracy in Section 3.1. We illustrate the theoretical results with numerical examples in Section 4.

Notation: Throughout the paper, we adopt the following notations for functions’ domain.

\[ T_1 = \{(s, q) : 0 \leq s \leq 1, 0 \leq q \leq 1\} , \]
\[ T_2 = \{(s, r) : 0 \leq s, r \leq 1\} , \]
\[ T_3 = \{(s, q) : 0 \leq s \leq 1\} , \]
\[ \mathbb{C} = \{c \in C^1[0, 1] : c(1) = 0\} . \]

For \( f(s) \in L^p[0, 1] \) and \( g(s, q) \in L^p(T) \), where \( T \in \mathbb{R}^2 \), with \( p = 2 \) or \( \infty \), we define the following norms:

\[ \| f \|_2 := \| f \|_{L^2} = \sup_{s \in [0, 1]} | f(s) | , \]
\[ \| g \|_2 := \| g \|_{L^2} = \sup_{s, q \in T} | g(s, q) | , \]
\[ \| f \|^2_{L^2} := \int_0^1 f^2(s) ds , \]
\[ \| g \|^2_{L^2} := \int_0^1 g^2(s, q) dsdq . \]

2. Backstepping design for a PIDE with output and state delays

We consider the following PIDE system with state and sensor delay

\[ x_s(t) = -x_s(t) + c(s)x(1, t - r) + \int_s^1 f(s, q)x(q, t) dq , \]
\[ x(0,t) = U(t) , \]
\[ y(t) = x(1, t - h) . \]

for all \((s, q, t) \in [0, 1] \times \mathbb{R}^+ \) with the initial condition \( x(s, 0) = x_0(s) \), and \( y(t) \) representing the output that can be measured. There are two types of delays in the system: recycle delay \( r \) due to transport, and measurement delay \( h \). The delay \( h \) can be alternatively thought of as input delay—the modeler is free to treat “dead time” as acting at either the sensor or the actuator. We treat the dead time as acting at the sensor.

Assumption 1. Denote the upper bound of the delay by \( \bar{r} \), namely, \( r \leq \bar{r} \). Usually, the transportation delay \( r \) is longer than the dead time \( h \), so we assume \( h := \bar{r} - h > 0 \), and thus \( 0 < h, q < \bar{r} \leq \bar{r} \).

Assumption 2. \( c \in C^1([0, 1]) \) with \( c(1) = 0 \), \( f \in C^1(T_1) \), and let the following symbols denote their bounds: \( \tilde{c} := \| c \| , \tilde{f} := \| f \| . \)

We introduce transport PDEs to represent the delayed state and delayed measurement, rewriting (9)–(11) as:

\[ x_s(t) = -x_s(t) + c(s)x(1, t - r) + \int_s^1 f(s, q)x(q, t) dq , \]
\[ x(0,t) = U(t) , \]
\[ h(x_s(t)) = v_1(x, t) , \]
\[ v(t) = z(1, t) , \]
\[ \eta u_s(t) = u_2(t) , \]
\[ u_1(t) = v(0, t) . \]

for \((s, t) \in (0, 1) \times \mathbb{R}^+ \), with \( v_0(s) \) and \( u_0(s) \) denoting the initial conditions for \( v \) and \( u \), respectively. We will sketch the backstepping design with state feedback for system (12)–(17) in the following two subsections.

2.1. Backstepping design for delay compensation with state-feedback

First, we employ the following backstepping transformation:

\[ x(s, t) = \Gamma_{K, L, J}(x, v, u)(s, t) := x(s, t) - \int_s^1 K(s, q)x(q, t) dq \]
\[ - h \int_s^1 L(s + h\sigma(r, t))dr - \eta \int_s^1 J(s + \sigma r) u(r, t) dr , \]

and its associated inverse transformation

\[ x(s, t) = \Gamma_{K, L, J}^{-1}(z, v, u)(s, t) := z(s, t) + \int_s^1 B(s, q)z(q, t) dq \]
\[ + \int_0^s D(s, r) v(r, t) dr + \int_0^s E(s, r) u(r, t) dr , \]

where kernels \( K, L \) are defined on \( T_1, L \) on \([0, 1 + h] \) by treating the function \( s + h r \) as a single variable, \( J \) on \([0, 1] \) by treating the function \( s + \sigma r \) as a single variable, and \( D, E \) on \( T_2 \). The task of the transformation (18) is to produce the following stable target system:

\[ z_s(t) = - z_s(t) , \quad \forall (s, t) \in (0, 1) \times \mathbb{R}^+ , \]
\[ z(0, t) = 0 , \]
\[ h(z_s(t)) = v_1(z, t) , \]
\[ v(t) = z(1, t) , \]
\[ \eta u(z_s(t)) = u_2(t) , \]
\[ u_1(t) = v(0, t) . \]

To map (12)–(17) into (18)–(23), the kernels need to satisfy:

\[ K_s(s, q) = - K_0(s, q) + f(s, q) - \int_s^q K(s, r)f(r, q) dr . \]
\[ K_0(s, 1) = L(s + h) . \]
\[ L(\phi) = \begin{cases} J(\phi + \eta) , & \phi < 1 \\ 0 , & \phi \geq 1 \end{cases} , \]
\[ J(\sigma) = \begin{cases} \int_0^1 K(\sigma, q)c(q) dq - c(\sigma) , & \sigma < 1 \\ 0 , & \sigma \geq 1 \end{cases} . \]

As \( c(1) = 0 \) is assumed in Assumption 2, \( J \) is continuous at \( \sigma = 1 \). Substituting (27) and (26) into (25), one gets

\[ K_s(s, q) + K_0(s, q) = f(s, q) - \int_0^q K(s, r)f(r, q) dr . \]
\[ K_0(s, 1) = \int_{s + \tau}^{s + \tau} K(s + \tau, q)c(q) dq - c(s + \tau) , \quad s + \tau < 1 . \]
\[ K_0(s, 1) = \int_{s + \tau}^{s + \tau} K(s + \tau, q)c(q) dq - c(s + \tau) , \quad s + \tau \geq 1 \]

It is worth noticing that \( K(s, 1) = 0 \) when \( r \geq 1 \), which implies that only one-case situation is needed. Using the method of characteristics, we get the integral form

\[ K(s, q) = \begin{cases} \Phi(f)(s, q) + \Psi(c)(s, q) , & s + r < q \\ + (\Phi(f, K) + \Psi(c, K))(s, q) , & s + r \geq q \end{cases} . \]
where \( \Phi_0 \) and \( \Psi_0 \) are depend on \( f \) and \( c \), respectively,
\[
\Phi_0(f)(s, q) := - \int_s^{s+1-q} f(\theta, \theta - s + q) d\theta,
\]
and \( \Phi \) and \( \Psi \) are functionals acting on \( K \),
\[
\Phi(f, K) := \int_s^{s+1-q} \int_0^{\theta-s+q} f(r, \theta - s + q) K(\theta, r) dr d\theta,
\]
and \( \Psi(c, K) = \int_{s+q+1} c(\theta) K(s - q - 1 + r, \theta) dr d\theta \).

Based on (30), we can derive \( L, J \) from (26) and (27).

**Theorem 1.** For every \( f, c \in C^1(T_1) \times C_2 \), the kernel \( K \in C^2(T_1) \) and \( L, J \in C^2[0, 1 + h] \) have bounds
\[
|K(s, q)| \leq \mathcal{K} := \mathcal{K}(s + f) e^{(s+\hat{f})},
\]
and \( L(s) \leq \mathcal{L} := e^{(s+\hat{f})} \).

The proof can be found in [53]. From the boundary conditions (13) and (19), the controller is
\[
U(t) = \int_0^1 K(0, q) x(q, t) dq + h \int_0^1 L(r) u(r, t) dr
\]
\[
+ \eta \int_0^1 J(q) u(r, t) dr.
\]

### 2.2. Backstepping design for the observer and the output-feedback

In this subsection, we will briefly introduce the design of the observer and the output-feedback controller using the backstepping method, and the detailed design process can be found in [53]. The proposed observer is a copy of (12)–(17) with the measurement error:
\[
\hat{x}(s, t) = -\hat{x}_0(s, t) + \int_s^1 f(s, q) \hat{x}(q, t) dq + c(s) \hat{u}(0, t) + Q_1(s)(c(0, t) - \hat{0}(0, t)),
\]
\[
\hat{x}(0, t) = U(t),
\]
\[
h \hat{u}_0(s, t) = \hat{u}_0(s, t) + Q_2(s)(c(0, t) - \hat{0}(0, t)),
\]
\[
\hat{0}(1, t) = \hat{x}(1, t),
\]
\[
\eta \hat{u}_0(s, t) - \hat{u}_0(s, t),
\]
\[
\hat{u}(1, t) = x(1, t) - h,
\]
where observer gains \( Q_1(s), Q_2(s) \in L^2(0, 1) \) are to be determined later and the initial conditions are denoted by \( \hat{x}_0, \hat{u}_0, \hat{u}_0 \) in \( L^2(0, 1) \). Define the error states:
\[
\hat{x} = x - \hat{x}, \quad \hat{v} = v - \hat{v}, \quad \hat{u} = u - \hat{u},
\]
which gives
\[
\hat{x}(s, t) = -\hat{x}_0(s, t) + \int_s^1 f(s, q) \hat{x}(q, t) dq + c(s) \hat{u}(0, t) - Q_1(s) \hat{0}(0, t),
\]
\[
\hat{x}(0, t) = 0,
\]
\[
h \hat{u}_0(s, t) = \hat{u}_0(s, t) - Q_2(s) \hat{0}(0, t),
\]
\[
\hat{0}(1, t) = \hat{x}(1, t),
\]
\[
\eta \hat{u}_0(s, t) - \hat{u}_0(s, t),
\]
\[
\hat{u}(1, t) = x(1, t) - h,
\]
with the initial conditions \( \hat{x}_0 = x_0 - \hat{x}_0, \hat{v}_0 = v_0 - \hat{v}_0, \hat{u}_0 = u_0 - \hat{u}_0 \). We employ the following backstepping transformations,
\[
\hat{x}(s, t) = F_s M \hat{F}_s \hat{x}(s, t)
\]
\[
\hat{x}(0, t) = 0,
\]
\[
h \hat{u}_0(s, t) = \hat{u}_0(s, t) - Q_2(s) \hat{0}(0, t),
\]
\[
\hat{0}(1, t) = \hat{x}(1, t),
\]
\[
\eta \hat{u}_0(s, t) - \hat{u}_0(s, t),
\]
\[
\hat{u}(1, t) = x(1, t) - h,
\]
and \( \hat{v}(s, t) = R \hat{F}_s \hat{v}(s, t) \).

Theorem 2. For every \( (h, f) \in \mathbb{R}^+ \times C^1(T_1) \), the kernel Eq. (62)–(67) admits a unique solution \( F, P \in C^2(T_1) \) and \( R \in C^0[0, 1] \) with the bound
\[
|F(s, q)| \leq f e^{(q-s)} \leq F := f e^{\hat{f}},
\]
and their associated inverse transformations
\[
\hat{x}(s, t) = F^{-1}[\hat{F}_s \hat{R}(\hat{F}_s \hat{v}(s, t)),
\]
\[
\hat{u}(s, t) = R^{-1}[\hat{R}(\hat{u}(s, t)) := \hat{u}(s, t) + \int_0^1 \hat{R}(s - q) \hat{u}(q, t) dq,
\]
where observer kernels \( M \) defined in \( T_2 \), \( F, \hat{F} \), \( P \) defined in \( T_1 \), while \( P \) defined in \( [0, 1 + h] \) by treating the function \( s + hq \) of \( (s, r) \) as a single variable, and \( R, \hat{R} \) defined in \( [0, 1] \) by treating the function \( s - q \) of \( (s, r) \) as a single variable. The transformations (51) and (52) admit the following observer error target system:
\[
\hat{x}(s, t) = -\hat{x}_0(s, t) + S(s) \hat{u}(0, t),
\]
\[
\hat{v}(0, t) = 0,
\]
\[
h \hat{u}_0(s, t) = \hat{u}_0(s, t),
\]
\[
\hat{0}(1, t) = \hat{x}(1, t),
\]
\[
\eta \hat{u}_0(s, t) - \hat{u}_0(s, t),
\]
\[
\hat{u}(1, t) = x(1, t) - h,
\]
where
\[
S(s) = c(s) + \int_0^q F(s, q) S(q) dq.
\]
and we have \( S := ||S|| \geq e^{\hat{f}} \). To convert the error system to the target system, the observer kernels need to satisfy
\[
F_s(s, q) = -\hat{F}_s(s, q) + \int_q^1 f(s, r) F(r, q) dr - f(s, q),
\]
\[
h M_s(s, q) = \hat{M}_s(s, q) + h \int_q^1 f(s, r) M(r, q) dr,
\]
\[
h P_s(s, q) = \hat{P}_s(s, q) + h \int_q^1 f(s, r) M(r, q) dr + h \int_q^1 f(s, r) P(r, q) dr,
\]
\[
F(0, q) = 0, \quad M(s, q) = P(s, q),
\]
\[
P(0, q) = 0, \quad P(s, 1) = h F(s, 1),
\]
\[
R(\xi) = M(1, 1 - \xi),
\]
with the observer gains are given
\[
Q_1(s) = -\frac{1}{h} M(s, 0),
\]
\[
Q_2(s) = -M(s, 1) - s.
\]
To realize the inverse transformation, the inverse kernels satisfy
\[
\hat{F}_s(s, q) = -F_s(s, q) - \int_q^1 f(s, r) \hat{F}_s(r, q) dr - f(s, q),
\]
\[
\hat{F}(0, q) = 0,
\]
\[
\hat{P}(\xi) = \hat{P}(1 + h(1 - \xi)),
\]
\[
\hat{R}(\xi) = \hat{R}(1 + h(1 - \xi)),
\]
Theorem 3 (Deepenet Universal Approximation Theorem [50], Theorem 2.1). Let $X \subseteq \mathbb{R}^{d_x}$ and $Y \subseteq \mathbb{R}^{d_y}$ be compact sets of vectors $x$ in $X$ and $y$ in $Y$, respectively. Let $U : X \rightarrow U \subseteq \mathbb{R}^{d_u}$ and $V : Y \rightarrow V \subseteq \mathbb{R}^{d_v}$ be sets of continuous functions $u(x)$ and $v(y)$, respectively. Let $U$ and $V$ be compact. Assume the operator $G : U \rightarrow Y$ 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 $\theta^{(k)}$, $\theta^{(k)}$ for neural networks $\mathcal{N}^c(\cdot; \theta^{(k)}), \mathcal{N}^c(\cdot; \theta^{(k)})$, $k = 1, \ldots, p$ and $x_i \in X$, $j = 1, \ldots, m$, with corresponding $u_m = (u(x_1), u(x_2), \ldots, u(x_m))^T$, such that

$$
|G(u)(y) - G_0(u_m)(y)| \leq \epsilon,
$$

(80)

where

$$
G_0(u_m)(y) = \sum_{k=1}^{p} \theta^{(k)} \mathcal{N}^c(y; \theta^{(k)}),
$$

(81)

for all functions $u \in U$ and for all values $y \in G(u) \subseteq Y$.

The theorem provides the theoretical underpinning for the utilization of DeepONet-based controllers, enabling the approximation of control kernel operators using neural networks if the they are continuous. In this section, we will utilize three DeepONet to approximate the three state-feedback control kernel operators and two DeepONets to approximate two observer gain operators, instead of four observer kernel operators. These operators are defined as follows:

Definition 1. Kernel operator $K : \mathbb{R}^* \times C^1(T_1) \times C \rightarrow C^1(T_1)$, $L : \mathbb{R}^* \times \mathbb{R}^* \times C^1(T_1) \times C \rightarrow C^0([0,1+h])$ and $J : \mathbb{R}^* \times C^1(T_1) \times C \rightarrow C^0([0,1+h])$ are defined by

$$
K(s,q) := K(r, f, c),
$$

(82)

$$
L(\phi) := L(r, \eta, f, c),
$$

(83)

$$
J(\phi) := J(f, c).
$$

(84)

For each constant $r$, $h \in \mathbb{R}^+$ and function $f \in C^1(T_1)$, $c \in C$, the operators $K$, $L$, and $J$ can generate the kernel functions $K(s,q), L(\phi)$ and $J(\phi)$, respectively, which satisfy Eqs. (24)–(27).

It is worth noting that the operator $L$ is independent of $h$ because $h$ solely affects the domain of $\phi$, as shown in Eq. (26) that $L(\phi) = 0$ if $\phi \geq 1$. Similarly, the operator $J$ is independent of $h$.

Definition 2. Observer gain $Q_1, Q_2 : \mathbb{R}^\ast \times C^1(T_1) \rightarrow C^0([0,1])$ are defined by

$$
Q_1(s) := Q_1(h, f), \quad i, j = 1, 2
$$

(85)

For each constant $h \in \mathbb{R}^+$ and function $f \in C^1(T_1)$, the operators $Q_1$ can generate the observer gains $Q_1(s)$, which satisfy Eqs. (68) and (69).

It is noteworthy that we employ directly NNs to train the operators for the observer gains. This choice is driven by both the considerable number of observer kernels (four in total) and the fact that only the gains play a role in the observer's functioning.

3.1. Accuracy of approximation of backstepping operator with DeepONet

Lemma 1 (Lipschitzness of Backstepping Kernel Operators). The kernel operators $K : \mathbb{R}^* \times C^1(T_1) \times C \rightarrow C^1(T_1), L : \mathbb{R}^* \times \mathbb{R}^* \times C^1(T_1) \times C \rightarrow C^0([0,1+h])$ and $J : \mathbb{R}^* \times C^1(T_1) \times C \rightarrow C^0([0,1+h])$ are locally Lipschitz and, specifically, for any $r, f, c$, the operators satisfy

$$
\|K(t_1, f_1, c_1) - K(t_2, f_2, c_2)\| \\
\leq L_1 \max \{\|t_1 - t_2\|, \|f_1 - f_2\|, \|c_1 - c_2\|\},
$$

(86)

$$
\|L(t_1, \eta_1, f_1, c_1) - L(t_2, \eta_2, f_2, c_2)\| \\
\leq L_2 \max \{\|t_1 - t_2\|, \|\eta_1 - \eta_2\|, \|f_1 - f_2\|, \|c_1 - c_2\|\},
$$

(87)

$$
\|J(t_1, f_1, c_1) - J(t_2, f_2, c_2)\| \\
\leq L_3 \max \{\|t_1 - t_2\|, \|f_1 - f_2\|, \|c_1 - c_2\|\},
$$

(88)

with the Lipschitz constant $L_K, L_L, L_J > 0$.

Proof. We begin with the Lipschitz continuity of operator $K$, rewriting left hand side of (86) as

$$
\|K(t_1, f_1, c_1) - K(t_2, f_2, c_2)\| \\
\leq L_1 \max \{\|t_1 - t_2\|, \|f_1 - f_2\|, \|c_1 - c_2\|\}.
$$

(89)

where

$$
K(t_1, f_1, c_1) - K(t_2, f_2, c_2) = K(t_1, f_1, c_1) - K(t_2, f_2, c_2),
$$

(90)

$$
K(t_1, f_1, c_1) - K(t_2, f_2, c_2) = K(t_1, f_2, c_1) - K(t_2, f_2, c_1),
$$

(91)

$$
K(t_1, f_2, c_1) - K(t_2, f_2, c_1) = K(t_1, f_2, c_1) - K(t_2, f_2, c_1),
$$

(92)

We first consider the continuity of $K$ w.r.t. $r$, as $r$ is a scalar parameter. The integration form (30), can be further rewritten in the term of operator with two branches:

$$
\mathcal{K}_1(r) = \Phi_0(f) + \Phi_0(0, r) + \Phi_{12}(r, f, \mathcal{K}_2(r)) + \Phi_{12}(r, f, \mathcal{K}_2(r)) + \Phi_{12}(r, c, \mathcal{K}_2(r)), \quad s + r > q,
$$

(93)

$$
\mathcal{K}_2 = \Phi_0(f) + \Phi(f, \mathcal{K}_2), \quad s + r \leq q.
$$

(94)

where

$$
\Phi_{12}(r, f, \mathcal{K}_2(r)) = \int_{r}^{r+q} f(\theta, s+q-r) \mathcal{K}_2(s+q-r) dt, \quad \left(0 < s < r < q, \mathcal{K}_2(s+q-r) = 0 \right),
$$

(95)

$$
\mathcal{K}_2(r, c, \mathcal{K}_2) = \int_{r}^{r+q} f(\theta, s+q-r) \mathcal{K}_2(s+q-r) dt, \quad \left(0 < s < r < q, \mathcal{K}_2(s+q-r) = 0 \right),
$$

(96)

$$
\mathcal{K}_2(r, c, \mathcal{K}_2) = \int_{r}^{r+q} f(\theta, s+q-r) \mathcal{K}_2(s+q-r) dt, \quad \left(0 < s < r < q, \mathcal{K}_2(s+q-r) = 0 \right),
$$

(97)

$$
\mathcal{K}_2(r, c, \mathcal{K}_2) = \int_{r}^{r+q} f(\theta, s+q-r) \mathcal{K}_2(s+q-r) dt, \quad \left(0 < s < r < q, \mathcal{K}_2(s+q-r) = 0 \right),
$$

(98)

with

$$
\psi(r, s, q) = \min \{1, s - q + 1 + 2r\}.
$$

(99)

Take the derivative of the operators of $r$

$$
\partial_r \mathcal{K}_1(r) = \int_{r}^{r+q} f(\theta, s+q-r) \mathcal{K}_2(s+q-r) dt, \quad \left(0 < s < r < q, \mathcal{K}_2(s+q-r) = 0 \right).
$$

(100)
\[
\delta_{t_j}K_2 = 0, \quad s + r \leq q. \tag{101}
\]

where
\[
\Gamma(s, q) = c' (s - q + 1 + r) + \int_{s}^{s+q-1} f(\theta + r, \theta - s + q)(K_2 - K_1)(\theta, \theta + r)d\theta
\]
\[
+ 2c(\psi(s, q))(K_2 - K_1)(s - q + 1 + r, \psi(s, q)) - \epsilon(s - q + 1 + r)K_2(s - q + 1 + r) - \epsilon s - q + 1 + r)K_2(s - q + 1 + r) + \Psi_{11}(r, c, \delta_j(K_2(s, \theta))). \tag{102}
\]

Notice that \(\delta_t(K(s, \theta) = K(s, \theta).\) It is straightforward to demonstrate in a similar way the proof of Theorem 1 that \(K(s, q)\) is bounded by
\[
|K(s, q)| \leq \Gamma_0 e^{(q-s)}, \tag{103}
\]
with a constant \(\Gamma_0 > 0.\) Addition to the fact \(K\) is bounded, \(\Gamma\) is also bounded and the bounds is denoted by \(\bar{\Gamma} = \|\Gamma(s, q)\|\.\) Applying the successive approximation approach, we reach the boundedness of \(\delta_tK_1(s)\) as follows
\[
|\delta_tK_1(s)| \leq \epsilon \int_{s}^{s+q-1} \frac{(q-s)^n}{n!} \epsilon e^{c(s+q)}. \tag{104}
\]

Consequently, we infer that operator \(K\) is Lipschitz continuous with \(r\) with Lipschitz constant \(\Gamma_0 e^{(q-s)}\). Second, we investigate the boundedness of \(K_1 - K_2\). From (30), we have
\[
Kf_1 - Kf_2 = \Phi_0(f_1 - f_2) + \Phi(f_1 - f_2, Kf_2) + \Phi(f_1 - f_2, Kf_2). \tag{105}
\]

Introduce the iteration
\[
\delta_t K^{n+1} = \Phi(f_1 - f_2) + \Phi(f_1 - f_2, Kf_2). \tag{106}
\]

which verifies
\[
Kf_1 - Kf_2 = \sum_{n=0}^{\infty} \delta_t K^n. \tag{108}
\]

Recalling K is bounded and combining the definition of \(\Phi_0\) and \(\Phi\) in (31) and (33), respectively, we get
\[
\|\delta_t K^n\| = (1 + \|K\|)\|f_1 - f_2\|. \tag{109}
\]

By induction,
\[
\delta_t K^n \leq (1 + \|K\|)^n\|f_1 - f_2\|. \tag{110}
\]

Therefore it follows that for all \((s, q) \in T,\)
\[
|Kf_1 - Kf_2| \leq (1 + \|K\|)^q\|f_1 - f_2\|. \tag{111}
\]

Third, we consider the boundedness of (92). From (30), it derives
\[
K_{c_1} - K_{c_2} = \left\{ \begin{array}{ll} c_1 - c_2 + \Psi_1(c_1 K(c_1) - c_2 K(c_2)) & , \quad s + r < q \\ 0 & , \quad s + r \geq q \end{array} \right. \tag{112}
\]

In a similar way to get the bound of \(Kf_1 - Kf_2\), it arrives at
\[
|K_{c_1} - K_{c_2}| \leq (1 + \|K\|)^{q} |c_1 - c_2| \leq (1 + \|K\|)^q |c_1 - c_2|. \tag{113}
\]

Consequently, we get the Lipschitz constant
\[
L_K = 3 \max \left\{ \frac{1}{\epsilon} \left( \frac{c + f}{\epsilon} \right)(1 + \epsilon f) \right\} e^{(q+\epsilon)}. \tag{114}
\]

So far we have shown that operator \(K\) exhibits local Lipschitz continuity with respect to inputs \(r, f\) and \(c\).

Next, we will prove that operators \(L\) and \(J\) are local Lipschitz continuous using the similar approach due to they dependent on \(K\) as shown in (26) and (27). Subsequently, we just present the differences from the above proof. Denote
\[
\Theta(c, K)(\sigma) = \int_{s}^{q} K(r, f, c)(\sigma, q)c(q)dq. \tag{115}
\]

and thus
\[
\|J(r_1, f_1, c_1) - J(r_2, f_2, c_2)\| = \|\Theta(c_1 - c_2, K_1) + \Theta(c_2, K_2) - \Theta(c_1 - c_2, K_2)\|
\]
\[
\leq \epsilon L_K /3|\tau_1 - \tau_2| + ||f_1 - f_2|| + ||c_1 - c_2|| + (1 + \|K\||c_1 - c_2||
\]
\[
\leq L_J /3|\tau_1 - \tau_2| + ||f_1 - f_2|| + ||c_1 - c_2||, \tag{115}
\]

where
\[
L_J = \epsilon L_K + 3\|K\| + 3. \tag{116}
\]

Since \(L\) is a shift of \(J,\) we have
\[
\|L_{r_1, d_1, f_1, c_1} - L_{r_2, d_2, f_2, c_2}\|
\]
\[
= \|J(r_1, f_1, c_1)(\phi + \eta_1) - J(r_2, f_2, c_2)(\phi + \eta_2)\|
\]
\[
= \int_{\phi + \eta_1}^{\phi + \eta_2} \|K_{\phi + \eta_1}(\phi + \eta_2) - K_{\phi + \eta_2}(\phi + \eta_1)\|c_1(q)dq
\]
\[
+ \int_{\phi + \eta_1}^{\phi + \eta_2} K_{\phi + \eta_1}(\phi + \eta_2)c_1(q)dq - c_1(\phi + \eta_1) + c_1(\phi + \eta_2) + \Theta(c_1 - c_2, K_1)(\phi + \eta_2) + \Theta(c_2, K_2)(\phi + \eta_2) - c_1(\phi + \eta_2) + c_2(\phi + \eta_2). \tag{117}
\]

Recalling \(K(s, q) - K(s_2, q)\) \leq \Gamma_0 e^q\) due to (103), the left hand side of (87) becomes
\[
\|L_{r_1, d_1, f_1, c_1} - L_{r_2, d_2, f_2, c_2}\|
\]
\[
\leq \epsilon L\|\eta_1 - \eta_2\| + \epsilon L\|\eta_1 - \eta_2\| + L_\epsilon\|\eta_1 - \eta_2\|
\]
\[
+ \epsilon L_K /3|\tau_1 - \tau_2| + ||f_1 - f_2|| + ||c_1 - c_2|| + (1 + \|K\||c_1 - c_2||
\]
\[
\leq L_\epsilon /4|\tau_1 - \tau_2| + \eta_1 - \eta_2| + ||f_1 - f_2|| + ||c_1 - c_2||, \tag{118}
\]

where
\[
L_\epsilon = 4 \max \{\epsilon L_K /3, 1 + \|K\| + \epsilon L_K /3, \epsilon L_\Gamma e^q + \epsilon \|K\| + L_\epsilon\}. \tag{118}
\]

with \(L_\epsilon\) is the Lipschitz constant for function \(\epsilon \in C^1.\)

**Lemma 2 (Lipschitzness of Observer Gain Operators).** The observer gain operators \(Q_i : \mathbb{R}^n \times C^0([\tau_i]) \rightarrow C^0([0,1])\) for \(i = 1, 2,\) are locally Lipschitz and, specifically, the operators satisfy
\[
\|Q_i(h_1, f_1) - Q_i(h_2, f_2)\| \leq L_{Q_i} \max \|h_1 - h_2\| ||f_1 - f_2||. \tag{119}
\]

with the Lipschitz constants \(L_{Q_i} > 0.\)

The proof of this lemma is similar to that of Lemma 1, so we omit it due to space constraints.

Based on Theorems 1, 3, Lemma 1 and the Theorem 3.3 in paper [54], we get the following result for the approximation of the kernels by DeepONets.
Theorem 4. For any \((r, f, c) \in \mathbb{R}^4 \times C^1(T) \times C\) with \(\epsilon, \kappa > 0\), there exist positive integers \(p^*(\epsilon), m^*(\epsilon)\), such that for any \(p > p^*\) and \(m > m^*\), there are neural networks \(f_y^N(\cdot; \theta_y^k), g_y^N(\cdot; \theta_y^k)\), \(i = 1, 2, 3, k = 1, \ldots, p\), and \(s, q \in T_i\), \(j = 1, \ldots, m\), such that
\[
\|K(r, f, c) - \hat{K}(r, f, c)\|_m(s, q) \leq \epsilon.
\]
\[
\hat{K} = \sum_{k=1}^{p} g_y^N((r, f, c)m; \theta_y^k) f_y^N(s; \theta_y^k).
\]
\[
\|\mathcal{L}(r, n, f, c)\|_m \leq \epsilon.
\]
\[
\hat{L} = \sum_{k=1}^{p} g_y^N((r, f, c)_m; \theta_y^k) f_y^N(\phi; \theta_y^k).
\]
\[
\|\mathcal{J}(r, f, c)\| \leq \epsilon.
\]
\[
\hat{J} = \sum_{k=1}^{p} g_y^N((r, f, c)_m; \theta_y^k) f_y^N(\sigma; \theta_y^k).
\]

Remark 1. It is worth noting that the parameter \(m\) determines the number of grids used for discretizing the function. For instance, a two-dimensional function \(f\) should be discretized on a grid on \(T_i\) with \(m\) grid points.

Theorem 5. For any \((h, f) \in \mathbb{R}^4 \times C^1(T)\) and \(\epsilon > 0\), there exist positive integers \(p^{**}(\epsilon), m^{**}(\epsilon)\), such that for any \(p > p^{**}\) and \(m > m^{**}\), there are neural networks \(f_y^N(\cdot; \theta_y^k), g_y^N(\cdot; \theta_y^k)\), \(i = 1, 2, \ldots, p\), and \(s, q\) \in \(T_i\), \(j = 1, \ldots, m\), such that
\[
\|\mathcal{Q}(h, f) - \hat{Q}(h, f)\|_m(s, q) \leq \epsilon.
\]
\[
\hat{Q} = \sum_{k=1}^{p} g_y^N((h, f)_m; \theta_y^k) f_y^N(s; \theta_y^k).
\]

Remark 1. It is worth noting that the parameter \(m\) determines the number of grids used for discretizing the function. For instance, a two-dimensional function \(f\) should be discretized on a grid on \(T_i\) with \(m\) grid points.

3.2. State-Feedback stabilization under DeepONet gain

Let \(\hat{K} = \hat{K}(r, f, c)\), \(\hat{L} = \hat{L}(r, h, f, c)\) be approximate operators, and their image functions, with accuracy \(\epsilon\) relative to the exact backstepping kernel \(K = K(r, f, c)\), \(L = L(r, h, f, c)\) and \(J = J(r, h, f, c)\), respectively. The following theorem establishes the properties of the feedback system.

Theorem 6. For any \((r, h, f, c) \in \mathbb{R}^4 \times C^1(T) \times C\), there exist a sufficiently small \(\epsilon^* > 0\), such that the feedback control law
\[
U(t) = \int_{0}^{1} \hat{K}(0, q)x(q, t) dq + \int_{0}^{1} \hat{L}(hr)v(r, t) dr
\]
with \(\eta > 0\) such that the closed-loop system satisfies the exponential stability bound, for all \(\epsilon > 0\)
\[
\|x\|_{L_2}^2 + \|\hat{v}\|_{L_2}^2 + \|u\|_{L_2}^2
\]
\[
\leq W_0 e^{-\eta \|q_0\|_{L_2}^2} + \|u_i\|_{L_2}^2 + \|u_j\|_{L_2}^2.
\]

Proof. Before proceeding, let \(\hat{K} = K - \hat{K}, \hat{L} = L - \hat{L}\) and \(\hat{J} = J - \hat{J}\) denote the difference between the kernels and their approximations.

The proof includes three steps. First, we take the same transformation as (18), while with the controller (128), we have the following target system:
\[
z_i(s, t) = -z_i(s, t),
\]
\[
z(0, t) = -\int_{0}^{1} \hat{K}(0, q)x(q, t) dq - \int_{0}^{1} \hat{L}(hr)v(r, t) dr
\]
\[
- \int_{0}^{1} J(\eta)u(r, t) dr.
\]

Second, we substitute the inverse transformation of (18) into (122) and get a boundary condition exclusively containing states \((z, v, u)\)
\[
z(0, t) = -\int_{0}^{1} \hat{K}(0, q)\Gamma^{-1}B,L,D,E(z, v, u)(q, t) dq
\]
\[
- \int_{0}^{1} \hat{L}(hr)v(r, t) dr - \eta \int_{0}^{1} J(\eta)u(r, t) dr
\]
\[
= -\int_{0}^{1} z(q, t)\hat{K}(0, q) + \int_{0}^{1} \hat{K}(0, r)B(r, q) dq
\]
\[
- \int_{0}^{1} v(q, t)\hat{L}(hr) + \int_{0}^{1} \hat{L}(0, r)D(r, q) dq
\]
\[
- \int_{0}^{1} u(q, t)\hat{J}(\eta) + \int_{0}^{1} \hat{J}(0, r)E(r, q) dq.
\]

Substituting (18) into (18), we get the relationship between the direct and inverse backstepping kernels:
\[
B(s, q) = \hat{K}(s, q) + \int_{d}^{s} \hat{K}(s, a)\hat{K}(a, q) da
\]
\[
D(s, r) = \hat{L}(s + hr) + \int_{d}^{s} \hat{L}(s, a)D(a, s) da
\]
\[
E(s, r) = \eta J(s + \eta r) + \int_{d}^{s} \hat{J}(s, a)E(a, s) da.
\]

Hence, the inverse kernel satisfies the following bounds:
\[
\|B\| \leq \hat{B} = \|K\|e^{\|K\|} = \hat{K}\hat{E},
\]
\[
\|D\| \leq \hat{D} = \|L\|e^{\|L\|} = \hat{L}\hat{E},
\]
\[
\|E\| \leq \hat{E} = \|J\|e^{\|J\|} = \hat{J}\hat{E}.
\]

Third, we carry out the Lyapunov stability analysis. Define the following Lyapunov functionals:
\[
V_1 = \|z\|^2_{L_2}, V_2 = \int_{0}^{1} e^{-\eta t} z^2(s, t) ds,
\]
Taking the time derivative of $V$ which yields

$$\dot{V} = \frac{1}{h}e^{-\beta h}V_4 - \frac{1}{\eta}e^{-\beta h}V_6 \leq \frac{1}{\eta} \dot{V}_6,$$

(149)

with $\beta_i > 0$, $i = 1, 2, 3$. Note that the following Lyapunov functional pairs satisfy norm-equivalence relationships: $V_i$ and $\dot{V}_i$; $V_4$ and $\dot{V}_4$; $V_5$ and $\dot{V}_5$, namely,

$$\dot{V}_4 \leq V_4 \leq e^h V_4,$$

(147)

$$\dot{V}_5 \leq V_5 \leq \frac{1}{\eta} \dot{V}_5,$$

(148)

with $\beta_i > 0$, $i = 1, 2, 3$. Note that the following Lyapunov functional pairs satisfy norm-equivalence relationships: $V_i$ and $\dot{V}_i$; $V_4$ and $\dot{V}_4$; $V_5$ and $\dot{V}_5$, namely,

$$\dot{V}_4 \leq V_4 \leq e^h V_4,$$

(147)

$$\dot{V}_5 \leq V_5 \leq \frac{1}{\eta} \dot{V}_5,$$

(148)

Recalling the boundary condition (137) and $|\dot{\mathcal{K}}_i|, |\dot{L}_i|, |\dot{J}| < \varepsilon$ given in Theorem 4, we know

$$z^2(0, t) \leq 6e^2 \mathcal{K}_V + 6e^2 h^2 \mathcal{L}_V + 6e^2 \eta^2 \mathcal{L}_V,$$

(150)

where

$$\mathcal{K}_V = 1 + \mathcal{K}^2 e^2, \quad \mathcal{L}_V = 1 + L^2 e^2.$$

(151)

In addition to the norm inequalities (147)–(149), we reach

$$V \leq -(\beta_i e^{-\beta_i h} - \beta_i e^{-\beta_i h^2})z^2(1) - (\beta_i - e^{-\beta_i h^2})u^2(0) - u^2(0)$$

$$- (\beta_i - 6e^2 \beta_i h \mathcal{K}_V) - (\beta_i - 6e^2 \beta_i h \mathcal{L}_V) - (\beta_i - 6e^2 \beta_i \eta \mathcal{L}_V).$$

(152)

Letting $\beta_i \leq \beta_2$, $\beta_i e^{\beta_i h^2} \leq \beta_i e^{\beta_i h^2}$ and

$$(\varepsilon^*)^2 = \min \left\{ \frac{b_1}{h e^2}, \frac{b_2}{6e^2 \beta_i h \mathcal{L}_V}, \frac{b_3}{6e^2 \beta_i \eta \mathcal{L}_V} \right\}.$$

(153)

To maximize the value of $\varepsilon^*$, we choose $\beta_i = \beta_i e^{\beta_i h^2}$ and $\beta_2 = e^{\beta_i h^2}$, which yields

$$(\varepsilon^*)^2 = \min \left\{ \frac{b_1}{6e^2 \beta_i h \mathcal{L}_V}, \frac{b_2}{6e^2 \beta_i h \mathcal{L}_V}, \frac{b_3}{6e^2 \beta_i \eta \mathcal{L}_V} \right\}.$$

(154)

If we select $\varepsilon < \varepsilon^*$, there exists a $a_0(\varepsilon) > 0$, such that

$$V \leq -a_0 V,$$

where

$$a_0 = \min \left\{ \frac{b_1}{h e^2}, \frac{b_2}{6e^2 \beta_i h \mathcal{L}_V}, \frac{b_3}{6e^2 \beta_i h \mathcal{L}_V}, \frac{b_4}{\eta e^2 \beta_i h \mathcal{L}_V} \right\}$$

which yields $V \leq V(0)e^{-a_0 t}$. It is derived from (144)–(146),

$$m_1(V_1 + V_3 + V_5) \leq V \leq m_2(V_1 + V_3 + V_5),$$

(155)

with

$$m_1 = \min\{\beta_i e^{-\beta_i h}, h \beta_i, \eta \}, \quad m_2 = \max\{\beta_i, h \beta_i e^{\beta_i h^2}, \eta e^{\beta_i h^2}\}.$$n

Therefore

$$(V_1 + V_3 + V_5) \leq m_2/V_1(0) + V_3(0) + V_5(0)e^{-a_0 t}.$$n

(146)

Also, we get the $L^2$ norm relationship between the states of (12)–(17) and those of (131)–(136),

$$\frac{1}{m_4} V_0 \leq V_1 + V_3 + V_5 \leq m_3 V_0,$$

(156)

where $V_0 = \|x\|_{L^2}^2 + \|e\|_{L^2}^2 + \|u\|_{L^2}^2$, with

$$m_1 = \max\{4(1 + \mathcal{K}^2), 1 + 4h^2 L^2, 1 + 4\eta^2\mathcal{L}^2\},$$

(157)

$$m_2 = \max\{4(1 + B^2), 1 + 4D^2, 1 + 4E^2\}.$$n

Hence, we arrive at the stability bound (129) with

$$W_0 = \frac{m_2 m_4 m_1}{m_4}.$$n

3.3. Stabilization of the observer error system under DeepONet observer gain

Theorem 7. For any $(h, f) \in \mathbb{R}^+ \times C^1(T_1)$, there exist a sufficiently small $\varepsilon^* > 0$, such that observer

$$\dot{x}_i(s, t) = \frac{1}{f_0} \left( f(s, q) \dot{x}_i(q, t) d\sigma q + v_i(t) \right),$$

$$\dot{v}_i(t) = 0,$$

(156)

where $\mathcal{O}_i := \mathcal{O}_i(h, f)$, defined in (85), $i = 1, 2$, of approximation accuracy $\varepsilon \in (0, \varepsilon^*)$ ensures the observer error system, for all initial conditions $x_0, \hat{x}_0, v_0, \hat{v}_0, u_0 \in L^2[0, 1]$, satisfies the exponential stability bound

$$\|x - \hat{x}\|_{L^2}^2 + \|u - \hat{u}\|_{L^2}^2 \leq W_1 e^{\alpha_1 t} \left( \|x_0 - \hat{x}_0\|_{L^2}^2 + \|v_0 - \hat{v}_0\|_{L^2}^2 \right),$$

(162)

with $W_1 > 0$ and $a_1 > 0$.

Proof. Before proceeding, let $\mathcal{O}_i = \mathcal{O}_i - \mathcal{O}_i, i = 1, 2$ denote the difference between the exact observer gain and the neural operators. Similar to the proof of Theorem 6, the proof contains two steps. First, we employ the transformation (51) and (52) to convert the error system (45)–(50), where the gains $Q$, are replaced with the NO observer gain $\mathcal{O}_i, i = 1, 2$, to the following target system

$$\dot{z}_i(s, t) = -z_i(s, t) + S_i(s) \dot{s}(0, t) + \delta_i(s) \dot{z}(0, t),$$

(163)

$$\dot{z}(0, t) = 0,$$

(164)

$$\mathcal{Q}_i(s, t) = \mathcal{Q}_i(s, t) + \delta_i(s) \dot{s}(0, t),$$

(165)

$$\dot{\mathcal{Q}}_i(t) = \mathcal{Q}_i(t),$$

(166)

$$\eta_\mathcal{Q}_i(s, t) = \dot{\mathcal{Q}}_i(s, t),$$

(167)

$$\dot{\mathcal{Q}}_i(t) = 0,$$

(168)

where

$$\delta_i(s, q) = \int_s^1 F(s, q) \delta_i(q) d\sigma q + \frac{1}{h} \int_0^1 M(s, q) \delta_i(q) d\sigma q,$$

(169)

$$\dot{\delta}_i(s) = \int_0^1 R(s - q) \delta_i(q) d\sigma q + \dot{Q}_i(s),$$

(170)

and $S$ is defined in (61). With (125) in Theorem 5, it is obvious that

$$\|\delta_i\| \leq \delta_i := \varepsilon \varepsilon, \quad \|\delta_i\| \leq \delta_i := \varepsilon \varepsilon,$$

(171)

(172)

where $\gamma_0 = \varepsilon \varepsilon \left( \frac{1}{2M_{\delta_1}} + 1 \right).$

Second, we introduce the Lyapunov functional

$$V_10 = \beta_1 V_2 + V_3 + \beta_1 V_4,$$

(173)
with
\[ V_2 = \int_0^1 e^{-\beta_2 t} z^2(s,t) ds, \]  
(174)
\[ V_h = h \int_0^1 e^{\beta_1 t} \tilde{z}^2(s,t) ds \]
(175)
and \( \beta_1, \beta_2 \) are positive constants. Taking the time derivative, we get
\[ V_{10} = \beta_1 \int_0^1 e^{-\beta_2 t} \tilde{z}^2(s,t) ds + \int_0^1 e^{\beta_1 t} \tilde{z}^2(s,t) ds \]
\[ + 2 \beta_2 \int_0^1 e^{-\beta_2 t} \tilde{z}(s,t) d\tilde{s}(0) + 2 \beta_1 \int_0^1 e^{\beta_1 t} \tilde{z}(s,t) d\tilde{s}(0) \]
\[ + 2 \int_0^1 e^{\beta_1 t} \tilde{u}(s,t) d\tilde{s}(0) + \beta_1 \int_0^1 e^{\beta_1 t} \tilde{u}^2(s,t) ds, \]
(176)
where we have used \( \tilde{v}(0) = \tilde{w}(0) \) from (52).

\[ V_{10} \leq -\beta_1 e^{-\beta_1 t} \tilde{z}^2(1) - \beta_2 \beta_3 V_h + e^{\beta_1 t} \tilde{u}^2(0) - \frac{b_3}{h} V_h \]
\[ - \beta_2 \tilde{d}_1 V_1 + \beta_2 \beta_3 \tilde{d}_2 \tilde{u}^2(0) + \frac{h}{\beta_3} V_h + \beta_3 \tilde{d}_1 \tilde{u}^2(0) \]
\[ \leq - (\beta_1 - \beta_2 \beta_3) \tilde{z}^2(1) - (1 - \beta_2 \beta_3) \tilde{d}_1 \tilde{u}^2(0) \]
\[ - (\beta_1 - \beta_2 \beta_3) \tilde{d}_2 \tilde{u}^2(0) - \frac{1}{\eta} \beta_2 b_3 V_h \]
\[ - \frac{1}{h} (b_2 - \beta_2) V_1 - \frac{1}{\eta} \beta_2 b_2 V_h. \]
(177)
Therefore, there exists a \( a_1(\epsilon) > 0 \) for \( \epsilon < \epsilon^* \), such that \( V_{10} \leq -a_1 V_{10} \).

\[ V_{10} = \int_0^1 \tilde{X}(0) \tilde{x}(q,t) dq + h \int_0^1 \tilde{Z}(hr) \tilde{y}(r,t) dr \]
\[ + \eta \int_0^1 \tilde{J}(hr) \tilde{y}(r,t) dr, \]
(182)
to stabilize the system (12)–(17). Fig. 1 illustrates the framework of the neural operator based output feedback for the delayed PDE system. As shown in Fig. 1, we apply three neural operators to learn the operators \( K, J, \) and \( Q \) defined in (82)–(84), then to derive the gain functions which are used in the controller. For the observer, we apply two neural operators to learn the operator \( Q_1, Q_2 \) defined in (85), which are used in the observer. We use the estimated system states for feedback with the learned neural gain functions in the control law. The control kernel and the observer gain functions can be learned once. The trained DeepONets are ready to produce the control kernel and observer gain functions for any new functional coefficients and any new delays.

The following theorem establishes the exponentially stability for the cascading system under the output-feedback control with the DeepONet gains.

**Theorem 8.** Consider the system (12)–(17), along with the observer (156)–(161) and the control (182), where the exact backstepping control kernels \( K, J, \) and observer gains \( Q_1, Q_2 \) are approximated by DeepONets \( \hat{K}, \hat{J}, \) and \( \hat{Q}_1, \hat{Q}_2 \), respectively with the accuracy \( \epsilon \in (0, \epsilon^*). \) For any \( (r, f, c) \in \mathbb{R}^n \times C^1(T_1) \times C \) corresponding to the control kernels \( K, J, r, f, c) \in \mathbb{R}^n \times \mathbb{R}^n \times C^1(T_1) \times C \) to \( L \), and \( (h, f) \in \mathbb{R}^n \times C^1(T_1), \) corresponding to observer gains \( Q_1, Q_2 \), there exist a sufficiently small \( \epsilon^* > 0 \), such that the observer-based control (182) ensures that the observer cascading closed-loop system satisfies the exponential stability bound, for all \( t > 0 \)
\[ \theta(t) \leq W_1 e^{-\alpha t} \theta(0), \]
(183)
where
\[ \theta(t) = ||x||^2_{L^2} + ||\tilde{y}||^2_{L^2} + ||u||^2_{L^2} + ||\tilde{y}||^2_{L^2} + ||\tilde{y}||^2_{L^2} + ||\tilde{z}||^2_{L^2}, \]
with \( W_1 > 0 \) and \( \alpha > 0 \).

**Proof.** We consider the observer error system
\[ \tilde{x}(s, t) = -\tilde{x}(s, t) + \int_s^1 f(s, q) \tilde{x}(q, t) dq \]
\[ + \epsilon(s) \tilde{y}(0, t) - \hat{Q}_1(s) \tilde{y}(0, t), \]
(184)
and the observer (156)–(161) with control (182), since they are equivalent to the cascading system (12)–(17) and observer (156)–(161) with control (182).

The proof contains two steps. First, we derive the target system of the state-feedback system, except for one extra term \( \dot{z}(s,t) = \mathcal{G}(s) \tilde{u}(0,t) \), by applying the backstepping transformation

\[
\dot{z}(s,t) = \mathcal{G}(s) \tilde{u}(0,t)
\]

(190)

Second, we introduce the following Lyapunov functional to prove the stability of the cascading target system (193)–(198) and (162)–(168). Since target system (193)–(198) has the same form as that of the target system of the state-feedback system, except for one extra term in (193) and in (195), respectively, and boundary condition (198), we redefine the Lyapunov functionals (144)–(146) as follows:

\[
V_1 = \|z\|^2_{L^2}, \quad V_2 = \int_0^t e^{-b_1 s} \|z(s,t)\|^2 ds,
\]

(185)

\[
V_3 = \|\tilde{z}\|^2_{L^2}, \quad V_4 = h \int_0^t e^{-b_2 s} \|\tilde{z}(s,t)\|^2 ds,
\]

(186)

\[
V_5 = \|\tilde{u}\|^2_{L^2}, \quad V_6 = \eta \int_0^t e^{-b_3 s} \|\tilde{u}(s,t)\|^2 ds,
\]

(187)

with \( b_i > 0 \), \( i = 1, 2, 3 \). Take time derivative of \( V := \beta_1 V_2 + \beta_2 V_4 + V_6 \) with \( \beta_i > 0 \), \( i = 1, 2, 3 \), we have

\[
\dot{V} = -\beta_1 e^{b_1 s} \tilde{z}(s,t) + \beta_2 e^{b_2 s} \tilde{z}(s,t) - \beta_2 e^{b_2 s} \tilde{z}(s,t) + 2\beta_1 \int_0^t e^{-b_1 s} \|z(s,t)\|^2 ds dt
\]

(188)

\[
\dot{V} = -\beta_1 e^{b_1 s} \tilde{z}(s,t) + \beta_2 e^{b_2 s} \tilde{z}(s,t) - \beta_2 e^{b_2 s} \tilde{z}(s,t) + 2\beta_1 \int_0^t e^{-b_1 s} \|z(s,t)\|^2 ds dt
\]

(189)

and combining (150), we get

\[
V \leq \left( -\beta_1 e^{b_1 s} \tilde{z}(s,t) + \beta_2 e^{b_2 s} \tilde{z}(s,t) - \beta_2 e^{b_2 s} \tilde{z}(s,t) + 2\beta_1 \int_0^t e^{-b_1 s} \|z(s,t)\|^2 ds dt \right)
\]

(203)

and combining (150), we get

\[
V \leq \left( \beta_1 e^{b_1 s} \tilde{z}(s,t) + \beta_2 e^{b_2 s} \tilde{z}(s,t) - \beta_2 e^{b_2 s} \tilde{z}(s,t) + 2\beta_1 \int_0^t e^{-b_1 s} \|z(s,t)\|^2 ds dt \right)
\]

(200)

\[
V \leq \left( -\beta_1 e^{b_1 s} \tilde{z}(s,t) + \beta_2 e^{b_2 s} \tilde{z}(s,t) - \beta_2 e^{b_2 s} \tilde{z}(s,t) + 2\beta_1 \int_0^t e^{-b_1 s} \|z(s,t)\|^2 ds dt \right)
\]

(201)

\[
V \leq \left( -\beta_1 e^{b_1 s} \tilde{z}(s,t) + \beta_2 e^{b_2 s} \tilde{z}(s,t) - \beta_2 e^{b_2 s} \tilde{z}(s,t) + 2\beta_1 \int_0^t e^{-b_1 s} \|z(s,t)\|^2 ds dt \right)
\]

(202)

and combining (150), we get

\[
V \leq \left( -\beta_1 e^{b_1 s} \tilde{z}(s,t) + \beta_2 e^{b_2 s} \tilde{z}(s,t) - \beta_2 e^{b_2 s} \tilde{z}(s,t) + 2\beta_1 \int_0^t e^{-b_1 s} \|z(s,t)\|^2 ds dt \right)
\]

(203)

and combining (150), we get

\[
V \leq \left( -\beta_1 e^{b_1 s} \tilde{z}(s,t) + \beta_2 e^{b_2 s} \tilde{z}(s,t) - \beta_2 e^{b_2 s} \tilde{z}(s,t) + 2\beta_1 \int_0^t e^{-b_1 s} \|z(s,t)\|^2 ds dt \right)
\]

(200)

\[
V \leq \left( -\beta_1 e^{b_1 s} \tilde{z}(s,t) + \beta_2 e^{b_2 s} \tilde{z}(s,t) - \beta_2 e^{b_2 s} \tilde{z}(s,t) + 2\beta_1 \int_0^t e^{-b_1 s} \|z(s,t)\|^2 ds dt \right)
\]

(201)

\[
V \leq \left( -\beta_1 e^{b_1 s} \tilde{z}(s,t) + \beta_2 e^{b_2 s} \tilde{z}(s,t) - \beta_2 e^{b_2 s} \tilde{z}(s,t) + 2\beta_1 \int_0^t e^{-b_1 s} \|z(s,t)\|^2 ds dt \right)
\]

(202)

and combining (150), we get

\[
V \leq \left( -\beta_1 e^{b_1 s} \tilde{z}(s,t) + \beta_2 e^{b_2 s} \tilde{z}(s,t) - \beta_2 e^{b_2 s} \tilde{z}(s,t) + 2\beta_1 \int_0^t e^{-b_1 s} \|z(s,t)\|^2 ds dt \right)
\]

(203)
using simultaneously the networks for operators $\hat{\rho}$ numerical solutions with different parameters, and then train $L$ have four input variables, we first train $\hat{\rho}$ smooth $L$ pointed out, all the NNs in the following simulations are trained using $L$ we will apply the smooth $L$ the smooth $L$ the smooth $L$ using both loss functions is shown in Fig. 3, which illustrates that using $L$ compared to the $L$ find that using the smooth $L$ $L$ small. We also train the DeepONets using the $L$ the prediction and the true value is small, the gradient value is also $L$ loss near $L$ portion replaced with a quadratic function such that its slope is $L$. The smooth $L$ Loss($\hat{\rho} - \rho$) = $L$ if $|\hat{\rho} - \rho| < 1$, $L$ otherwise. The smooth $L$ loss can be seen as exactly $L$ loss, but with the $|\hat{\rho} - \rho|$ portion replaced with a quadratic function such that its slope is $L$ at $|\hat{\rho} - \rho| = 1$. The quadratic segment when $|\hat{\rho} - \rho| < 1$ smooths the $L$ loss near $|\hat{\rho} - \rho| = 0$, avoiding sharp changes in slope. The smooth $L$ combines the advantages of the $L$ and $L$ loss functions. When the difference between the prediction and the true value is large, the gradient value will not be too large; when the difference between the prediction and the true value is small, the $L$ loss has smaller errors and less fluctuations. Therefore, we will apply the smooth $L$ to train the networks. If not specifically pointed out, all the NNs in the following simulations are trained using the smooth $L$ loss function.

Since operator $K$'s input variables are three and the other operators have four input variables, we first train $\hat{K}(r, f, c)$ on a dataset of 8000 numerical solutions with different parameters, and then train simultaneously the networks for operators $\hat{L}(r, h, f, c)$ and $\hat{J}(r, h, f, c)$ using 10000 instances. The NN for operator $K$ achieves a training loss of $1.36E-5$ and a testing loss of $1.34E-5$ after 300 epochs in around 11 minutes, shown in Fig. 3(a). The two NNs for operator $L$ and $J$ achieve a training loss of $1.90E-5$ and a testing loss of $1.90E-5$ after 300 epochs in about 15 minutes, which are shown in Fig. 3(b). (The experimental code runs on Intel® Core™ i9-7900X CPU @ 3.30 GHz x 20 and GPU TITAN Xp/PCIe/SSE2.) In Fig. 4, we demonstrate the analytical kernels that are solved numerically, the learned DeepONet kernels and the errors between them, where the coefficients are chosen as $r = 1$, $h = 0.5$, $f(s, q)$ with $\mu_1 = \mu_2 = 5$ and $c(s)$ with $\mu_3 = 5$. Also, in Fig. 5, we show both the analytical control gains and learned control gains, respectively.

To test the performance of the neural operator based control, we apply the trained neural gains in controller (128). Here, we use the same parameter settings as Fig. 4 and let initial condition be $x(s, 0) = \sin(x)$. The upwind scheme with a time step size of $\Delta t = 0.001$ and a trapezoidal integration rule are used to numerically solve the PIDE under controller (128). Before proceeding, we show in Fig. 6 that the dynamical state $x(s, t)$ of the system under a nominal controller without delay compensation fails to converge. In Fig. 7, we demonstrate the dynamics of the closed-loop with the full state feedback, using the numerically solved control gains and the DeepONet learned control gains, respectively. The closed-loop system dynamics with NO kernels approximates the PIDE well with a peak error of less than 8% compared to the closed-loop system with analytical kernels.
Fig. 4. The first row shows the kernel functions $K(s, q)$, the learned kernel functions $\hat{K}(s, q)$ and the errors $K(s, q) - \hat{K}(s, q)$. The second row shows the kernel functions $L(\phi)$, the learned kernel functions $\hat{L}(\phi)$ and the errors $L(\phi) - \hat{L}(\phi)$. The last row shows the kernel functions $J(\sigma)$, the learned kernel functions $\hat{J}(\sigma)$ and the errors $J(\sigma) - \hat{J}(\sigma)$.

Fig. 5. The first row shows the analyzed control gains $K(0, q)$, $L(hr)$, $J(\eta r)$, and the learned control gains $\hat{K}(0, q)$, $\hat{L}(hr)$, $\hat{J}(\eta r)$. The last row shows the errors $K(0, q) - \hat{K}(0, q)$, $L(hr) - \hat{L}(hr)$, $J(\eta r) - \hat{J}(\eta r)$. 
4.2. Output feedback

We train two neural observer gains \( \hat{Q}_1(s) \) and \( \hat{Q}_2(s) \) instead of the four observer kernels, which reduces the computational cost in half. The same parameter settings as for the full-state feedback are applied in the NN training, and the sensor delay \( h \) is chosen from \( U(0.1, 0.6) \).

Similar to the DeepONets for learning control kernels, except that the input channel for the first layer CNN is 2. Two DeepONets are employed to learn the gain functions \( Q_1 \) and \( Q_2 \), respectively, each containing 6926913 parameters. The two observer networks are trained together on 1600 instances, which only takes around 4 min. Fig. 8 shows the analyzed observer gains, the learned DeepONet observer gains and the errors between them as \( h = 0.5s \). The network achieves a training loss of 5.02E−5 and a testing loss of 6.36E−5 after 300 epochs, which are shown in Fig. 3(c).

Fig. 9 demonstrates the convergence of the observation with the DeepONet learned gains to the system’s actual state. In Fig. 10, we test the closed-loop system under the output feedback (182) with three DeepONets approximating the control kernels and two DeepONets approximating the observer gains when the initial condition of the observer is set to the initial value of the system plus a random number that obeys \( U(-1, 1) \).

Table 1 presents a comparison of the time consumption for kernel functions when solved numerically versus that generated by the trained DeepONets, respectively. The term ‘average calculation time’ refers to the mean time taken over 100 runs. The duration required by numerical solvers grows significantly with the increase in discrete spatial step size, namely sampling precision. Conversely, the computation time for NOs shows only a marginal increase with larger spatial step sizes. However, the loss defined in (208) still maintains on the order of 10^{-5}. It is worth noting that the approximation accuracy \( \varepsilon \) defined in Theorems 4 and 5 corresponds to the Euclidean norm (or 2-norm) in vector space for the spatially discretized function (See, e.g. Theorem 2 in [49]). In the other words, the accuracy, in terms of \( \varepsilon \), is the square root of the loss (208). From the simulation results, we find that the accuracy of the control kernels is on the order of 10^{-3} and that of the observer gains on the order of 10^{-2}.

5. Conclusion

In this paper, we apply the DeepONet operators to learn the PDE backstepping control kernels and observer of a first-order hyperbolic PDE system with state and sensor delays. Three neural operators have been trained with a set of numerical solutions derived from backstepping kernel equations. These operators are capable of approximating three control kernel functions with an accuracy on the order of 10^{-3}. Leveraging the universal approximation theorem, we demonstrate the potential for neural operators to approximate analytical kernel operators with arbitrary precision. The stability of the closed-loop system, when utilizing state feedback augmented with neural operator learning gains, has been established. Additionally, we employ two neural operators to learn the observer gains and establish that the observer, when enhanced with neural gains, converges. Simulation results indicate that the accuracy of approximating observer gains can attain an order of 10^{-2}. Integrating these gains into both the observer-based control system and the observer error system, we demonstrate the stability of the output feedback system, which verifies the separation principle under the neural operator gains. Further research will concern the delay-adaptive control of PDEs whose delays are unknown and control of high-dimensional PDEs control whose kernel functions are defined in higher spatial dimension.

CRediT authorship contribution statement

Jie Qi: Conceptualization, Formal analysis, Methodology, Writing – original draft, Writing – review & editing. Jing Zhang: Data curation, Software, Validation, Visualization, Writing – review & editing. Miroslav Krstic: Conceptualization, Methodology, Supervision, Writing – original draft, Writing – review & editing.

Declaration of competing interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Data availability

The codes are shared on github with link: https://github.com/JingZhang-JZ/NO_hyperbolic_delay.git.

Appendix. Derivation of the Kernel equations

Take the time derivative of the transformation (18), which yields

\[
\begin{align*}
z_j(s,t) &= x_j + c(s)u(0,t) + \int_s^1 f(s,q)x(q,t)dq \\
&+ K(s,1)x(1) - K(s,s)x(s) - \int_s^1 K_q(s,q)x(q)dq \\
&- \int_s^1 K(s,q)c(q)dq(0,t) - \int_s^1 K(s,q)x(q,t)dq \\
&- \int_s^1 \int_q^1 K(s,q)f(q,r)x(r)drdq \\
&- L(s + h)c(1) + L(s)c(0) + h \int_0^1 L'(s + hr)c(r,t)dr \\
&- J(s + \eta)c(1) + J(s)c(0) + \eta \int_0^1 J'(s + \eta r)c(r,t)dr.
\end{align*}
\]

And taking the derivative w.r.t. \( s \), one can get

\[
\begin{align*}
z_j(s,t) &= x_j(s,t) + K(s,s)x(s,t) - \int_s^1 K_q(s,q)x(q,t)dq \\
&- h \int_0^1 L'(s + hr)c(r,t)dr - \eta \int_0^1 J'(s + \eta r)c(r,t)dr.
\end{align*}
\]

Substituting the above two equations into (18)–(23), we can derive the kernel functions Eqs. (24)–(27) based on certain equivalence relations.
### Table 1
Summary of kernel function calculation time consumption.

<table>
<thead>
<tr>
<th>Model</th>
<th>Average calculation time (sec) (spatial step size (\Delta s = 0.02))</th>
<th>Average calculation time (sec) (spatial step size (\Delta s = 0.01))</th>
<th>Average calculation time (sec) (spatial step size (\Delta s = 0.005))</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Control kernels</strong></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>((K, L, J))</td>
<td>Numerical solver 0.025</td>
<td>0.654</td>
<td>1.692</td>
</tr>
<tr>
<td></td>
<td>Neural operators 0.011</td>
<td>0.0139</td>
<td>0.0265</td>
</tr>
<tr>
<td></td>
<td>Speedups 2.3x</td>
<td>47.1x</td>
<td>63.8x</td>
</tr>
<tr>
<td><strong>Observer gains</strong></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>((Q_1, Q_2))</td>
<td>Numerical solver 0.030</td>
<td>0.079</td>
<td>0.348</td>
</tr>
<tr>
<td></td>
<td>Neural operators 0.006</td>
<td>0.006</td>
<td>0.007</td>
</tr>
<tr>
<td></td>
<td>Speedups 5x</td>
<td>13.2x</td>
<td>49.7x</td>
</tr>
</tbody>
</table>

Fig. 7. The closed-loop evolution with full-state feedback (128). The left column in the first row shows state \(x(s, t)\) with the analyzed kernels \(K\), \(L\) and \(J\). The right column in the first row shows state \(x_{NO}(s, t)\) with the learned kernels \(\hat{K}\), \(\hat{L}\) and \(\hat{J}\). The last row shows the \(L_2\)-norm of state \(x\), \(x_{NO}\), and the error between them.

Fig. 8. The first row shows the analyzed observer gains \(Q_1(s)\), \(Q_2(s)\), and the learned observer gains \(\hat{Q}_1(s)\), \(\hat{Q}_2(s)\). The last row shows the error \(Q_1(s) - \hat{Q}_1(s)\), \(Q_2(s) - \hat{Q}_2(s)\).
Fig. 9. The system actual state $x(s, t)$ and the neural operator based observer $\hat{x}_{\text{NO}}(s, t)$ under controller $U(t) = 5 \sin(3\pi t) + 3 \cos(2\pi t)$. The left column showcases the actual state of the system state. The right column showcases the estimated system state $\hat{x}_{\text{NO}}(s, t)$ with the neural operator based observer. Note that the initial condition of the system is $x(s, 0) = \sin(2\pi s)$, while the initial condition of the neural operator observers is $\hat{x}_{\text{NO}}(s, 0) = 10$.

Fig. 10. The closed-loop evolution under output feedback. The left column in the first row showcases the evolution of state $x(s, t)$ with the analytical kernels $K, L, J$ and observer gains $Q_1, Q_2$. The right column in the first row showcases the evolution of state $\hat{x}_{\text{NO}}(s, t)$ with NO kernels $\hat{K}, \hat{L}, \hat{J}$ and NO observer gains $\hat{Q}_1, \hat{Q}_2$. The last row shows the $L_2$-norm of state $x, x_{\text{NO}}$, and the error between them.
References


