LEARNING THE RICCATI SOLUTION OPERATOR FOR TIME-VARYING LQR VIA DEEP OPERATOR NETWORKS

JUN CHEN*, UMBERTO BICCARI†, AND JUNMIN WANG*

Abstract. We propose a computational framework for replacing the repeated numerical solution of differential Riccati equations in finite-horizon Linear Quadratic Regulator (LQR) problems by a learned operator surrogate. Instead of solving a nonlinear matrix-valued differential equation for each new system instance, we construct offline an approximation of the associated solution operator mapping time-dependent system parameters to the Riccati trajectory. The resulting model enables fast online evaluation of approximate optimal feedback laws across a wide class of systems, thereby shifting the computational burden from repeated numerical integration to a one-time learning stage.

From a theoretical perspective, we establish control-theoretic guarantees for this operator-based approximation framework. In particular, we derive bounds quantifying how operator approximation errors propagate to feedback performance, trajectory accuracy, and cost suboptimality, and we prove that exponential stability of the closed-loop system is preserved under sufficiently accurate operator approximation. These results provide a systematic framework to assess the reliability of data-driven approximations in optimal control.

On the computational side, we design tailored DeepONet architectures for matrix-valued, time-dependent problems and introduce a progressive learning strategy to address scalability with respect to the system dimension. Numerical experiments on both time-invariant and time-varying LQR problems demonstrate that the proposed approach achieves high accuracy and strong generalization across a wide range of system configurations, while delivering substantial computational speedups compared to classical Riccati solvers. The method offers an effective and scalable alternative for parametric and real-time optimal control applications.

More broadly, the proposed approach illustrates how operator learning can be used to build reusable surrogates for nonlinear matrix differential equations arising in control and dynamical systems.

1. Introduction and motivations

Optimal control plays a central role in science and engineering, offering a systematic framework for designing control strategies that regulate dynamical systems while balancing performance and control effort [1, 2, 3, 4].

Among the most widely used formulations, the Linear Quadratic Regulator (LQR) provides a tractable and robust methodology, with applications including robotics, aerospace engineering, signal processing, and autonomous systems [5, 6, 7, 8, 9, 10]. Its formulation is characterized by the Riccati equation, which determines the optimal feedback controller.

Despite its classical nature, solving LQR problems remains computationally demanding in modern settings where system parameters vary or must be evaluated repeatedly. In the case of linear time-varying systems, the optimal feedback law is characterized by the Differential Riccati Equation

2020 *Mathematics Subject Classification.* 68T07,93B52,93D15,93D40.

Key words and phrases. Riccati equation, Operator Learning, DeepONets, Optimal control.

This project has received funding from the European Research Council (ERC) under the European Union's Horizon 2030 research and innovation programme (grant agreement NO: 101096251-CoDeFeL). UB was partially supported by the Grant PID2023-146872OB-I00-DyCMaMod of MICIU (Spain) and by the COST Actions ''CA24122 - multiscale Stochastics, Patterns, and Analysis of Combinatorial Environments'' and ''CA24136 - Interactions between Control Theory and Machine Learning''. CJ and JW were supported by the National Natural Science Foundation of China under Grant 92471108 and 12131008, and the Program of China Scholarship Council(Grant No.202506030033).

(DRE), a nonlinear matrix-valued differential equation that must be solved backward in time. Classical numerical methods for Riccati equations include direct and iterative schemes based on matrix factorizations, invariant subspace techniques, and backward integration methods, whose computational cost typically scales cubically with the state dimension [11, 12].

While this approach is effective for solving a single instance of an LQR problem, it becomes a major limitation in settings where one must evaluate optimal feedback laws for a large number of system configurations. Such situations arise naturally in parametric control, uncertainty quantification, and real-time applications like Model Predictive Control (MPC), where optimal control problems must be solved sequentially in real time for evolving system states and parameters. In these contexts, each new parameter instance requires solving a Riccati equation from scratch, leading to a computational cost that grows linearly with the number of instances and rapidly becomes prohibitive in high-dimensional settings.

This computational bottleneck motivates the search for alternative paradigms that can exploit the structure of the problem across different parameter configurations. In particular, the repeated solution of Riccati equations can be replaced by the evaluation of a surrogate model that directly maps system parameters – such as time-dependent matrices $(A(t), B(t), Q(t), R(t))$ – to the associated solution.

Under this perspective, in this work, we adopt an Operator Learning (OL) approach in which the Riccati equation is viewed as a nonlinear mapping between function spaces. Instead of solving the DRE for each new instance, we aim to approximate this solution operator once, and then reuse it to enable fast evaluation of approximate optimal feedback laws for previously unseen systems. This leads to an offline-online computational strategy, where the main computational effort is performed during a training stage, and the evaluation for new parameter configurations becomes essentially instantaneous.

To realize this operator-based framework, we employ Deep Operator Networks (DeepONets [13, 14]), which are specifically designed to learn mappings between infinite-dimensional function spaces. DeepONets combine a branch network that encodes the input functions with a trunk network that represents the output dependence, providing a flexible architecture for approximating operators arising in differential equations. This enables a single trained model to generalize across a broad class of systems with varying dynamics and cost structures. In contrast to existing learning-based approaches for solving differential equations and control problems –, such as Physics-Informed Neural Networks (PINNs) [15, 16], originally developed for PDE approximation, and Reinforcement Learning (RL) methods [17, 18] –, which generally require instance-wise training or optimization, our approach targets the full Riccati solution operator, allowing a single trained model to generalize across a broad class of systems.

More precisely, existing approaches differ fundamentally in how they treat parametric variability. PINNs are typically trained to approximate the solution of a single differential equation instance by enforcing the governing equations in the loss function. As a consequence, each new system configuration requires retraining or fine-tuning, which limits their applicability in settings involving large families of parameter-dependent problems.

RL methods, on the other hand, aim to learn control policies through interaction with the system, often without explicitly exploiting the underlying Riccati structure. While RL provides a flexible framework for control, it generally focuses on policy approximation rather than on learning the mapping from system parameters to optimal solutions, and may require extensive sampling or exploration for each new environment.

In contrast, the OL approach adopted in this work targets directly the parametric solution map associated with the Riccati equation. By learning this mapping offline, the resulting model can be reused across a wide range of system configurations without retraining, enabling an amortized computational strategy that fundamentally differs from instance-wise approaches.

While we focus on the Riccati equation arising in LQR problems, the proposed approach is representative of a broader computational paradigm: learning reusable solution operators for nonlinear matrix-valued differential equations in control and dynamical systems. In this perspective, the Riccati equation serves as a canonical testbed where both numerical efficiency and control-theoretic properties, such as stability and optimality, can be rigorously assessed.

A key challenge in this setting is to ensure that the learned approximation preserves the fundamental properties of the optimal solution, in particular stability and performance guarantees of the resulting closed-loop system. To address this issue, we establish a rigorous link between the approximation error of the learned Riccati operator and the behavior of the induced feedback law. More precisely, we show that errors at the operator level propagate in a controlled manner to the feedback gain, the closed-loop trajectories, and the associated cost functional. In particular, the performance loss of the learned controller can be explicitly bounded in terms of the operator approximation error, and exponential stability of the closed-loop system is preserved provided this error remains sufficiently small. This establishes a complete error propagation chain from operator approximation to control performance, providing a control-theoretic interpretation of OL in this setting.

The proposed framework is validated through numerical experiments demonstrating that the learned operator achieves high accuracy and near-perfect stabilization rates, while significantly reducing computational cost compared to classical solvers. Furthermore, we show that the approach scales to higher-dimensional systems and can exploit structural similarities across dimensions through a progressive learning strategy, whose empirical performance suggests the presence of low-dimensional structure in the underlying operator.

The main contributions of this work can be summarized as follows:

1. we introduce an operator-based computational framework for LQR, replacing repeated numerical solution of Riccati equations by a learned solution operator;
2. we design a DeepONet-based architecture tailored to matrix-valued, time-dependent control problems;
3. we establish a control-theoretic error propagation framework linking operator approximation error to feedback gain accuracy, trajectory deviation, cost suboptimality, and closed-loop stability;
4. we demonstrate the effectiveness and scalability of the approach through extensive numerical experiments.

The remainder of the paper is organized as follows. Section 2 introduces the finite-horizon LQR problem and formulates the associated Riccati equation from an OL perspective. Section 3 presents the DeepONet-based approximation framework, including the network architecture, data generation strategies, and a progressive learning approach to address scalability across dimensions. Section 4 provides theoretical guarantees, including approximation results for the Riccati operator and an analysis of error propagation and closed-loop stability. Section 5 reports numerical experiments illustrating the accuracy, stability, and computational efficiency of the proposed method in both time-invariant and time-varying settings. Finally, Section 6 gathers our conclusions and propose new future research directions.

2. Problem formulation and operator viewpoint

From an engineering perspective, LQR provides a systematic framework to design feedback controllers that achieve an optimal trade-off between system performance and control effort. More precisely, the objective of LQR is to regulate the state of a dynamical system toward a desired equilibrium (typically the origin) while minimizing a quadratic cost functional that penalizes both deviations of the state and the magnitude of the control input.

LQR has been extensively used across a wide range of engineering domains. In aerospace engineering, it is employed for attitude control and trajectory stabilization of aircraft and spacecraft, where maintaining stability under perturbations is critical [19, 20]. In robotics, LQR-based controllers are used for motion planning, balancing problems (e.g., inverted pendulum or humanoid robots) [10, 21], and trajectory tracking [22]. In electrical and mechanical systems, it appears in vibration suppression, power system regulation, and process control [23, 24, 25]. More recently, LQR also plays a central role in modern control architectures such as MPC, where it serves as a local optimal controller or terminal cost approximation [26, 27].

In this work, we consider the finite-horizon LQR problem for continuous-time linear time-varying systems. Let $T > 0$ be a fixed time horizon. The system dynamics are given by

$$\begin{cases} \dot{x}(t) = A(t)x(t) + B(t)u(t), \quad t \in [0,T], \\ x(0) = x_0, \end{cases} \tag{2.1}$$

where $x(t) \in \mathbb{R}^n$ denotes the state and $u(t) \in \mathbb{R}^m$ with $m \leq n$ the control input. The matrices $A(\cdot) \in \mathbb{R}^{n\times n}$ and $B(\cdot) \in \mathbb{R}^{n\times m}$ are assumed to be continuous in time.

In this context of time-varying systems, the finite-horizon LQR problem is particularly relevant for transient control tasks, where system parameters evolve over time or where control objectives are defined over a finite time interval. Examples include trajectory optimization [28], time-dependent resource allocation [29], and adaptive control in uncertain environments [30].

The LQR controller is designed by minimizing the quadratic cost functional

$$J(u) = \int_0^T \Big(x(t)^\top Q(t)x(t) + u(t)^\top R(t)u(t) \Big)\, dt + x(T)^\top P_T x(T), \tag{2.2}$$

where $Q(\cdot) \in \mathbb{R}^{n\times n}$ and $R(\cdot) \in \mathbb{R}^{m\times m}$ are continuous and symmetric weighting matrices, with $Q(\cdot)$ positive semi-definite and $R(\cdot)$ positive definite, and $P_T \in \mathbb{R}^{n\times n}$ is a positive definite and time-invariant terminal cost matrix.

The structure of this cost functional (2.2) reflects two competing design requirements. On the one hand, the term $x^\top Qx$ enforces state regulation, promoting accuracy, stability, and tracking performance. On the other hand, the term $u^\top Ru$ penalizes control effort, preventing excessive actuator usage, energy consumption, or aggressive control actions that may be infeasible in practice. The choice of the weighting matrices Q, R, and P_T therefore encodes engineering specifications such as precision, robustness, and resource constraints.

One of the key advantages of the LQR framework is that it yields a closed-loop linear state feedback law that is optimal and stabilizing under mild assumptions. This makes it particularly attractive in real-time applications, where computational simplicity and robustness are essential. In more detail, it is well known [2] that, under standard stabilizability and detectability assumptions (see Section 4), the optimal control is given in feedback form

$$u^*(t) = -K^*(t)x(t), \qquad \text{with } K^*(t) = R^{-1}(t)B^\top(t)P^*(t),$$

where $P^*(t)$ is the unique symmetric positive definite solution of the Differential Riccati Equation (DRE)

$$\begin{cases} -\dot{P}(t) = A(t)^\top P(t) + P(t)A(t) - P(t)B(t)R^{-1}(t)B(t)^\top P(t) + Q(t), \ t \in [0,T] \\ P(T) = P_T. \end{cases} \tag{2.3}$$

Despite its conceptual simplicity, the practical deployment of LQR in modern applications is often hindered by computational challenges. As discussed in the introduction, solving the Riccati equation (2.3) repeatedly for varying system parameters can become prohibitively expensive in

high-dimensional or real-time settings. To overcome this bottleneck, in this work, we adopt an operator viewpoint. We denote by

$$\Theta = (A(\cdot), B(\cdot), Q(\cdot), R(\cdot), P_T)$$

the collection of system parameters, and we define the mapping

$$\mathcal{F} : \Theta \mapsto P(\cdot),$$

which associates to each parameter instance the corresponding solution of the Riccati equation.

From a computational viewpoint, this operator encapsulates the full dependence of the optimal control law on the system parameters. Classical approaches evaluate this mapping by solving the Riccati equation (2.3) numerically for each input Θ. In contrast, our objective is to approximate this operator once and reuse it, enabling efficient evaluation across multiple problem instances.

3. DeepONet-based approximation of the Riccati operator

We present a DeepONet-based framework to approximate the Riccati solution operator introduced in Section 2. The goal is to construct a neural network that maps system parameters directly to the corresponding solution of the Riccati equation, enabling fast evaluation for new instances.

3.1. Operator Learning formulation: the DeepONet architecture.

Recall that we aim to approximate the mapping

$$\mathcal{F} : \Theta \mapsto P(\cdot),$$

which associates to each system configuration $\Theta = (A(\cdot), B(\cdot), Q(\cdot), R(\cdot), P_T)$ the solution of the corresponding Riccati equation.

In practice, both the input functions and the output trajectory must be represented in finite-dimensional form. To this end, as we will see, the time-dependent matrices are sampled or parametrized using suitable discretizations or basis expansions, leading to a finite-dimensional representation of the input. Similarly, the solution $P(t)$ is evaluated at a set of time points or represented through a functional basis.

Under this representation, the OL problem is reduced to a supervised learning task: given a dataset of input–output pairs (Θ_i, P_i), where P_i is the solution of the Riccati equation corresponding to Θ_i, the objective is to construct a model that approximates the mapping $\mathcal{F}$ and generalizes to unseen parameter instances.

To approximate $\mathcal{F}$, we employ a DeepONet, which is designed to learn mappings between function spaces. The architecture consists of two main components: a branch network and a trunk network. The branch network processes the input representation of the system parameters Θ, encoding them into a finite-dimensional feature vector. The trunk network takes as input the time variable t and outputs a set of basis functions that capture the temporal structure of the solution.

The final output is obtained by combining the outputs of the branch and trunk networks. More precisely, the predicted solution is represented as

$$\mathcal{G}(\Theta)(t) = \sum_{k=1}^{p} \beta_k(\Theta)\tau_k(t) \sim P(t),$$

where $\beta_k(\Theta)$ are coefficients produced by the branch network and $\tau_k(t)$ are basis functions produced by the trunk network.

In order to handle matrix-valued outputs, the network is designed so that the final layer produces a vector representation that can be reshaped into a matrix of appropriate dimension. The architecture can be adapted to different problem sizes by adjusting the number of basis functions and network widths.

The DeepONet architecture provides a flexible framework for approximating nonlinear mappings between function spaces, such as the Riccati solution map $\mathcal{F}$. Its theoretical foundation relies on universal approximation results for operators [31, 14], which ensure that sufficiently regular mappings can be approximated with arbitrary accuracy on compact subsets.

It is worth emphasizing that this OL formulation differs conceptually from classical neural PDE solvers and control learning approaches. In particular, rather than approximating a single solution (as in PINNs) or directly learning a control policy (as in RL), the objective here is to approximate the full mapping from system parameters to Riccati solutions. This distinction is crucial in applications where the same model must be evaluated repeatedly for different system configurations, as it enables a clear separation between an offline training phase and a fast online evaluation stage.

In the present setting, the mapping $\mathcal{F}$, which associates system parameters to the corresponding solution of the Riccati equation, is well-defined and depends continuously on the input data. Moreover, under standard assumptions on stabilizability and positive definiteness, this dependence is locally Lipschitz, as follows from classical sensitivity results for Riccati equations [32, 11]. In this context, we have by [14, Theorem 1] that, for any $\varepsilon > 0$, there exists a DeepONet $\mathcal{G}$ such that for all Θ and $t \in [0, T]$,

$$\left\|\mathcal{F}(\Theta)(t) - \mathcal{G}(\Theta)(t)\right\| < \varepsilon.$$

3.2. **Data generation and training strategy.** The performance of the proposed OL approach, in particular the level of approximation ε that DeepONet can achieve, depends critically on the construction of representative training data [13]. In this section, we describe how datasets are generated for both time-invariant and time-dependent systems, as well as the training procedure used to fit the model.

3.2.1. *Data generation for time-invariant parameters.* For time-invariant parameters (A, B, Q, R), the Riccati equation (2.3) reduces to the Algebraic Riccati Equation (ARE)

$$A^\top P + PA - PBR^{-1}B^\top P + Q = 0. \tag{3.1}$$

In this case, our task reduces to learn the map $(A, B, Q, R) \mapsto P$. Notice that this is no longer an operator but rather a matrix-valued function. Nevertheless, the DeepONet methodology can still be applied.

To generate training samples, we shall consider structured families of linear systems (A, B) and design matrices (Q, R) that ensure controllability and cover a broad range of dynamical behaviors.

For what concerns Q and R, we will restrict them to be diagonal matrices with non-negative and positive diagonal entries, respectively, to preserve the semi-positive and positive nature required to build LQR controllers. This assumption simplifies the data generation procedure and allows for a controlled exploration of the influence of state and control weights. Notice that, while symmetric matrices are diagonalizable, restricting to diagonal structures does not cover the full generality of possible couplings between state components, and should therefore be interpreted as a modeling choice rather than a general representation.

For the dynamics matrices A and B, instead, we adopt parametrizations based on the Brunovsky canonical form, which provides a complete characterization of controllable linear systems up to similarity transformations. This is based on a fundamental property of control systems [33, 2]: for any controllable pair (A, B), there exists an invertible change of coordinates under which the system can be decomposed into a block-diagonal structure consisting of independent controllable subsystems. More precisely, we can find a matrix $T \in \mathbb{R}^{n\times n}$ such that

$$(T^{-1}AT,\, T^{-1}B) = (A_c, B_c)$$

with

$$A_c = \mathrm{diag}(A_{c,1}, \dots, A_{c,r}), \quad B_c = \mathrm{diag}(B_{c,1}, \dots, B_{c,r}), \tag{3.2}$$

and where each $(A_{c,i}, B_{c,i})$ is a single-input controllable canonical block of size n_i, with

$$A_{c,i} = \begin{bmatrix} 0 & 1 & & 0 \\ & \ddots & \ddots & \\ -a_{i,0} & \cdots & -a_{i,n_i-2} & -a_{i,n_i-1} \end{bmatrix}, \quad B_{c,i} = \begin{bmatrix} 0 \\ \vdots \\ 1 \end{bmatrix}, \quad n = \sum_{i=1}^{r} n_i. \tag{3.3}$$

This representation offers two key advantages for dataset generation. First, it ensures that all sampled systems are controllable by construction, which is a fundamental requirement for the well-posedness of the LQR problem and the existence of a unique positive definite solution to the Riccati equation. Second, it provides a systematic parametrization of the space of linear dynamics through the coefficients of the companion matrices, enabling controlled exploration of a wide range of dynamical behaviors.

From an approximation perspective, this parametrization induces a low-dimensional yet expressive representation of the space of admissible systems. Since similarity transformations do not affect the qualitative properties of the Riccati solution (e.g., stabilizability and optimal cost structure), learning the operator in Brunovsky coordinates captures the essential features of the mapping $(A, B, Q, R) \mapsto P$ without redundancy. This allows the network to focus on intrinsic dynamical features rather than superficial coordinate variations.

Overall, the use of Brunovsky canonical forms leads to a dataset that is both structurally rich and mathematically controlled, ensuring coverage of representative system behaviors while maintaining consistency with the theoretical assumptions underlying the Riccati equation.

3.2.2. *Data generation for time-dependent parameters.* For time-dependent parameters, the previous approach based on the Brunovsky decomposition is no longer applicable. We shall then define a different strategy to generate the matrices $A(t)$, $B(t)$, $Q(t)$ and $R(t)$.

As in the time-independent case before, the dataset must be representative enough so to ensure good generalization performances for DeepONet once trained. To adhere to this principle, the dataset is constructed using a truncated trigonometric expansion of the system matrices. In particular, we will consider matrices of the form

$$\begin{aligned} A(t) &= A_0 + \sum_{i=1}^{r_{\text{base}}} \Big(C_{1i}\phi_i(t) + C_{2i}\psi_i(t) \Big), \\ B(t) &= B_0 + \sum_{i=1}^{r_{\text{base}}} \Big(D_{1i}\phi_i(t) + D_{2i}\psi_i(t) \Big), \\ Q(t) &= M^\top(t) M(t) \quad \text{with} \quad M(t) = M_0 + \sum_{i=1}^{r_{\text{base}}} \Big(E_{1i}\phi_i(t) + E_{2i}\psi_i(t) \Big), \end{aligned} \tag{3.4}$$

with randomly sampled coefficients

$$\begin{matrix} A_0 \in \mathbb{R}^{n\times n}, & B_0 \in \mathbb{R}^{n\times m}, & M_0 \in \mathbb{R}^{n\times n} \\ C_{ki} \in \mathbb{R}^{n\times n}, & D_{ki} \in \mathbb{R}^{n\times m}, & E_{ki} \in \mathbb{R}^{n\times n} \end{matrix}, \quad i = 1, \dots, r_{\text{base}},\ k = 1, 2,$$

and where we have denoted

$$\phi_i(t) = \cos(i\pi t) \quad \text{and} \quad \psi_i(t) = \sin(i\pi t).$$

The control weighting matrix $R(t)$, instead, will be assumed to be the identity, i.e., $R(t) = R = I_m$.

This construction of the data can be interpreted as sampling from a structured low-frequency manifold in function space induced by truncated Fourier expansions. In this perspective, the matrices $A(\cdot)$, $B(\cdot)$ and $Q(\cdot)$ are not arbitrary time-dependent functions, but belong to a finite-dimensional

subspace spanned by smooth basis functions. This aligns with the OL framework, where the objective is to approximate mappings between compact subsets of infinite-dimensional spaces.

From an approximation-theoretic viewpoint, Fourier-type bases provide dense representations of smooth functions, ensuring that the sampled trajectories capture a wide range of temporal behaviors while remaining controlled in regularity. The truncation level r_{base} thus defines the intrinsic dimension of the function class and acts as a complexity parameter for the input space.

From a control perspective, this construction ensures that the generated systems exhibit bounded variation and regular dynamics, which guarantees well-posedness of the differential Riccati equation and stability of the numerical solvers. As a consequence, the dataset can be viewed as a compact, low-frequency subset of admissible system trajectories, on which the Riccati operator is learned.

Finally, we emphasize that this choice is not restrictive: by increasing the number of basis functions, one can approximate increasingly complex time dependencies. Therefore, the proposed dataset construction provides a systematic and scalable framework for generating representative training samples for time-varying LQR problems. In this sense, the trigonometric basis acts as a bridge between infinite-dimensional function spaces and their finite-dimensional representations used in the learning framework.

3.2.3. *Training procedure.* The model is trained using supervised learning, where the objective is to minimize the discrepancy between the predicted and true Riccati solutions over the training dataset. The loss function we employed is Mean Square Error (MSE), defined as

$$\mathcal{L} = \frac{1}{N}\sum_{i=1}^{N} \|P_i^{\text{pred}} - P_i^{\text{true}}\|_F^2, \qquad \text{time-independent parameters}$$

$$\mathcal{L} = \frac{1}{NM}\sum_{i=1}^{N}\sum_{m=1}^{M} \|P_i^{\text{pred}}(t_m) - P_i^{\text{true}}(t_m)\|_F^2, \qquad \text{time-dependent parameters}$$

where P_i^{true} denotes the reference solution, P_i^{pred} the network prediction, and $\{t_m\}_{m=1}^N$ are the discretization points of the time interval $[0, T]$.

Training is performed using standard optimization algorithms (e.g., Adam), with a suitable choice of learning rate, batch size, and number of epochs. The dataset is split into training and testing subsets to evaluate generalization performance. Complete implementation details will be given in Section 5.

3.3. **Progressive Operator Learning across dimensions.** A central challenge in OL is scalability with respect to the system dimension. In our case, as the state dimension increases, both the inputs (A, B, Q, R) and the output P grow quadratically, making direct learning of the Riccati operator increasingly expensive in terms of data, model size, and training time.

To address this issue, we propose a **Progressive Operator Learning** strategy, aimed at transferring knowledge learned in low-dimensional settings to higher-dimensional problems. The key idea is that the Riccati solution operator may exhibit structural similarities across dimensions, so that part of its complexity can be captured in a lower-dimensional representation and reused when moving to larger systems.

This intuition is partly supported by structural considerations in the time-invariant setting. In this case, controllable linear systems admit a canonical representation via the Brunovsky canonical form, under which the dynamics decompose into chains of integrators. The associated Riccati solutions inherit a hierarchical structure reflecting these subsystems. From this perspective, increasing the system dimension can often be interpreted as extending an existing structure rather than introducing entirely new dynamics, which provides a rationale for transferring a learned operator across dimensions.

However, this argument relies on the availability of such canonical decompositions and therefore applies primarily to the time-invariant case. In the time-dependent setting, no analogous global normal form is available, and the system matrices may exhibit more complex temporal variations. As a consequence, the assumption that the Riccati operator admits a dimension-robust representation cannot be justified theoretically in the same way.

For this reason, the proposed progressive strategy should be understood as an empirically motivated approach aimed at exploiting potential low-dimensional structure in the Riccati operator across system dimensions. We do not claim the existence of a general dimension-independent representation of the Riccati operator. Instead, the construction can be viewed as a form of latent operator factorization, where the core mapping is captured in a reduced representation and complemented by dimension-specific transformations. The validity of this interpretation is ultimately assessed through numerical experiments presented in Section 5, which indicate that such a reduced representation can capture a significant portion of the operator complexity in the considered regimes.

Let $\mathcal{F}_n$ denote the operator mapping system parameters to the solution of the Riccati equation in dimension n. Our objective is to approximate $\mathcal{F}_{n_1}$ for $n_1 > n$ by leveraging a pre-trained approximation $\widehat{\mathcal{F}}_n$ of $\mathcal{F}_n$. This is achieved by constructing a composite architecture of the form

$$\widehat{\mathcal{F}}_{n_1} \sim \mathcal{E}_{n_1} \circ \widehat{\mathcal{F}}_n \circ \mathcal{I}_{n_1},$$

where

- $\mathcal{I}_{n_1}$ is an embedding operator mapping high-dimensional inputs into a lower-dimensional latent space;
- $\mathcal{E}_{n_1}$ is a lifting operator reconstructing the high-dimensional output.

This decomposition separates the learning task into a shared operator representation (captured by $\widehat{\mathcal{F}}_n$) and dimension-specific adaptations (captured by $\mathcal{I}_{n_1}$ and $\mathcal{E}_{n_1}$), allowing one to significantly reduce the number of trainable parameters and the associated computational cost.

From this viewpoint, the progressive architecture can be interpreted as approximating the Riccati operator through a structured low-dimensional latent representation, with embedding and lifting layers acting as dimension-dependent encoders and decoders.

3.3.1. *Architecture and error decomposition.* In practice, we will implement the progressive strategy by combining a pre-trained DeepONet with two additional neural components:

Embedding layer. A fully connected network $\mathcal{I}_{n_1}$ that maps the flattened high-dimensional input (namely, concatenated entries of (A, B, Q, R)) into a feature vector compatible with the input space of the pre-trained low-dimensional branch network.

Extension (lifting) layer. A fully connected network $\mathcal{E}_{n_1}$ that maps the low-dimensional output of the pre-trained model to the higher-dimensional Riccati solution.

The parameters of the pre-trained DeepONet are kept fixed, while only the embedding and extension layers are trained on high-dimensional data. This significantly reduces the number of trainable parameters and the associated computational cost.

Moreover, this progressive construction induces a natural decomposition of the approximation error. Denoting by $\mathcal{F}_{n_1}$ the true operator and by $\widehat{\mathcal{F}}_{n_1}$ its progressive approximation, we can formally write

$$\|\mathcal{F}_{n_1}(\Theta) - \widehat{\mathcal{F}}_{n_1}(\Theta)\| \leq \|\mathcal{F}_{n_1}(\Theta) - \mathcal{E}_{n_1} \circ \mathcal{F}_n \circ \mathcal{I}_{n_1}(\Theta)\| + \|\mathcal{F}_n - \widehat{\mathcal{F}}_n\| + \varepsilon_{\text{adapt}},$$

where the three terms correspond respectively to:

1. a **representation error**, measuring how well the high-dimensional operator can be expressed through the low-dimensional one;
2. the **approximation error** of the pre-trained DeepONet;
3. an **adaptation error**, arising from the training of the embedding and lifting networks.

This decomposition highlights that the effectiveness of the progressive strategy depends on the existence of a suitable low-dimensional representation and on the quality of the learned transformations.

4. Theoretical guarantees for the learned Riccati operator

The goal of this section is to characterize how approximation errors at the level of the Riccati solution operator affect the resulting control law and closed-loop dynamics. In particular, we aim to quantify how perturbations in the operator propagate through the feedback gain to the state trajectories and the associated cost functional. This provides a systematic framework to assess the reliability of learned operators in control applications.

4.1. **Notation.** Denote by $\mathbb{S}^n$ the set of $n \times n$ real symmetric matrices. Let $\mathbb{S}^n_+ \subset \mathbb{S}^n$ and $\mathbb{S}^n_{++} \subset \mathbb{S}^n$ denote the sets of positive semidefinite and positive definite matrices, respectively.

For $A, B \in \mathbb{S}^n$, we write $A \succeq B$ if $A - B \in \mathbb{S}^n_+$, and $A \succ B$ if $A - B \in \mathbb{S}^n_{++}$. In particular, $A \succeq 0$ (resp. $A \succ 0$) means that A is symmetric positive semidefinite (resp. symmetric positive definite), i.e., $x^\top A x \geq 0$ (resp. > 0) for all $x \in \mathbb{R}^n \setminus \{0\}$. Define

$$\Sigma^m_{++} := \Big\{ R : [0,T] \to \mathbb{S}^m_{++} \;\Big|\; \exists \rho > 0, \text{ s.t. } R(t) \succeq \rho I_m, \text{ for all } t \in [0,T] \Big\},$$

where I_m denotes the $m \times m$ identity. Finally, for a matrix $A \in \mathbb{R}^{n\times m}$ we denote by $\|A\|_F$ the Frobenius norm, and for a vector $x \in \mathbb{R}^n$ we denote by $\|x\|_2$ the Euclidean norm.

4.2. **Approximation of the Riccati operator.** The approximation of the Riccati operator by DeepONets is justified by combining regularity properties of the Riccati map with universal approximation results for OL. Under the assumptions in (4.1), the mapping $\mathcal{F} : \mathcal{X} \to \mathcal{Y}$ that associates the system parameters Θ with the solution $P(\cdot)$ of the Riccati equation is well-defined, continuous, and locally Lipschitz with respect to the input functions. This follows from standard sensitivity results for RDEs, which ensure continuous dependence of the solution on the coefficients. On the other hand, DeepONets are known to be universal approximators of nonlinear operators between Banach spaces. Therefore, the regularity of the Riccati operator together with the universal approximation property of DeepONets ensures that the solution operator $\mathcal{F}$ can be approximated arbitrarily well on compact subsets of $\mathcal{X}$.

Let $\Theta = (A(\cdot), B(\cdot), Q(\cdot), R(\cdot), P_T)$, and define the admissible input space

$$\mathcal{X} = \left\{ \Theta \text{ s. t. } \begin{array}{lll} A \in C([0,T];\mathbb{R}^{n\times n}) & B \in C([0,T];\mathbb{R}^{n\times m}) & \\ Q \in C([0,T];\mathbb{S}^n_+) & R \in C([0,T];\Sigma^m_{++}) & P_T \in \mathbb{S}^n_{++} \\ (A(t),B(t)) \text{ is uniformly stabilizable} & & \\ (A(t),Q(t)^{1/2}) \text{ is uniformly detectable} & & \end{array} \right\}. \tag{4.1}$$

Let us recall that (A, B) is stabilizable and $(A, Q^{1/2})$ is detectable if every eigenvalue $\lambda \in \mathbb{C}$ of A with non-negative real part satisfies

$$\operatorname{rank}\Big(\lambda I - A \,|\, B\Big) = n \quad \text{and} \quad \operatorname{rank}\begin{pmatrix} \lambda I - A \\ Q^{1/2} \end{pmatrix} = n.$$

For all $\Theta \in \mathcal{X}$, the Riccati equation (2.3) admits a unique solution $P \in C([0,T];\mathbb{S}^n_{++})$ [3, 34]. Accordingly, we define the output space

$$\mathcal{Y} := \Big\{ P \in C([0,T];\mathbb{S}^n_{++}) \;\Big|\; P \text{ solves (2.3) for some } \Theta \in \mathcal{X} \Big\}. \tag{4.2}$$

This choice of spaces allows us to view the Riccati map as an operator

$$\mathcal{F} : \mathcal{X} \mapsto \mathcal{Y}, \quad \mathcal{F}(\Theta) = P(\cdot). \tag{4.3}$$

which is continuous and locally Lipschitz with respect to the system parameters.

4.3. **Error propagation and stability analysis of the learned controller.** Consider system (2.1) with quadratic cost (2.2). For $\Theta = (A, B, Q, R, P_T) \in \mathcal{X}$, with $\mathcal{X}$ as in (4.1), let $P^* \in C([0,T]; \mathbb{S}^n_{++})$ be the unique solution of the DRE (2.3), and denote $\widehat{P}$ be an approximate solution of (2.3) obtained by DeepONet. Define:

$$K^*(t) = R^{-1}(t)B^\top(t)P^*(t), \qquad \widehat{K}(t) = R^{-1}(t)B^\top(t)\widehat{P}(t) \qquad \text{optimal feedback gain} \tag{4.4a}$$

$$x^*(t), \qquad \widehat{x}(t) \qquad \text{optimal state} \tag{4.4b}$$

$$u^*(t) = -K^*(t)x^*(t), \qquad \widehat{u}(t) = -\widehat{K}(t)\widehat{x}(t) \qquad \text{optimal control} \tag{4.4c}$$

Denoting by $\mathcal{F}$ the exact Riccati operator and by $\widehat{\mathcal{F}}$ its approximation, the learned control law is obtained by composing this operator with the feedback mapping. As a consequence, an approximation error at the operator level propagates through the following chain:

$$\widehat{\mathcal{F}} \sim \mathcal{F} \;\longrightarrow\; \widehat{K} \sim K^* \;\longrightarrow\; \widehat{x} \sim x^* \;\longrightarrow\; J(\widehat{u}) \sim J(u^*).$$

The following result provides a rigorous quantification of this propagation mechanism, including explicit bounds on the trajectory deviation, cost suboptimality, and stability of the resulting closed-loop system.

Theorem 4.1. *Given $T > 0$ and $\Theta = (A, B, Q, R, P_T) \in \mathcal{X}$, with $\mathcal{X}$ as in (4.1), let $P^* \in C([0,T]; \mathbb{S}^n_{++})$ be the unique solution of the DRE (2.3). Let $\widehat{P}$ be an approximate solution of (2.3) obtained by DeepONet, and define the error $E(t) := \widehat{P}(t) - P^*(t)$. Assume there exists $\varepsilon > 0$ such that*

$$\sup_{t\in[0,T]} \|E(t)\| < \varepsilon. \tag{4.5}$$

Then, for any initial datum $x_0 \in \mathbb{R}^n$, we have the following

1. ***Finite-horizon performance error.*** *There exist constants $C_1, C_2, C_3, C_4 > 0$, depending on Θ and T, such that the optimal states and controls given by (4.4b) and (4.4c) satisfy*

$$\sup_{t\in[0,T]} \|\widehat{x}(t) - x^*(t)\|_2 \le C_1\|x_0\|_2\Big(e^{C_2\varepsilon T} - 1\Big), \tag{4.6}$$

$$J(\widehat{u}) - J(u^*) \le C_3\|x_0\|_2^2\varepsilon^2 T e^{C_4\varepsilon T}. \tag{4.7}$$

2. ***Stability of the learned closed-loop system.*** *There exists $\varepsilon^* = \varepsilon^*(\Theta, T) > 0$ such that if $\varepsilon < \varepsilon^*$ the approximate optimal state $\widehat{x}(t)$ is uniformly stable on $[0,T]$. More precisely, there exist constants $M(\Theta,T), \mu(\Theta,T) > 0$ such that*

$$\|\widehat{x}(t)\|_2 \le Me^{-\mu t}\|x_0\|_2, \quad \text{for all } t \in [0,T].$$

Proof sketch. We provide here the main ideas for the proof of Theorem 4.1. Complete details can be found in Appendix A.

The argument is based on the fact that the feedback gain depends linearly on the Riccati matrix. Hence the approximation error on the learned Riccati operator propagates directly to the closed-loop dynamics. First, from the definition (4.4a) of the feedback gains, we can estimate

$$\|\widehat{K}(t) - K^*(t)\|_F \le C\|E(t)\|_F.$$

So controlling the Riccati error uniformly in time also controls the feedback perturbation.

For the trajectory estimate, let

$$e(t) := \widehat{x}(t) - x^*(t), \quad A^*_{cl}(t) := A(t) - B(t)K^*(t).$$

Since x^* solves the optimal closed-loop system and $\widehat{x}$ solves the perturbed one, subtracting the two equations yields an error dynamics of the form

$$\dot{e}(t) = A_{cl}^*(t)e(t) - B(t)R^{-1}(t)B^\top(t)E(t)\widehat{x}(t), \quad e(0) = 0.$$

Next, introduce the Lyapunov functional $V_e(t) \coloneqq e(t)^\top P^*(t)e(t)$. Because $P^*(t) \in \mathbb{S}^n_{++}$ uniformly on $[0,T]$, $V_e(t)$ is equivalent to $\|e(t)\|_2^2$. Differentiating $V_e(t)$ along the error dynamics, and using the Riccati equation satisfied by $P^*(t)$, the nominal terms cancel while the remaining terms are proportional to the perturbation $E(t)$. This gives a differential inequality of the form

$$\dot{V}_e(t) \le \alpha\|E(t)\|_F\sqrt{V_e(t)} + \beta\|E(t)\|_F V_e(t),$$

for suitable constants $\alpha, \beta > 0$. An application of Grönwall's lemma, together with $e(0) = 0$, then yields (4.6).

For the cost estimate, we write $J(\widehat{u}) - J(u^*)$ and expand the quadratic terms around the optimal pair (x^*, u^*). Since the LQR cost is quadratic and u^* is optimal, the first-order terms cancel, while the remaining terms are controlled by

$$\|\widehat{x} - x^*\|^2_{L^\infty(0,T)} \quad \text{and} \quad \|\widehat{u} - u^*\|^2_{L^\infty(0,T)}.$$

Using

$$\widehat{u} - u^* = -(\widehat{K} - K^*)\widehat{x} - K^*(\widehat{x} - x^*),$$

together with the previous gain and trajectory bounds and a Lyapunov analysis, one obtains (4.7).

Finally, for the stability part, the nominal optimal closed-loop system

$$\dot{x}^*(t) = \big(A(t) - B(t)K^*(t)\big)x^*(t)$$

is uniformly exponentially stable under the standing stabilizability and detectability assumptions used to define the admissible class $\mathcal{X}$. Since

$$\widehat{A}_{cl}(t) - A_{cl}^*(t) = -B(t)\big(\widehat{K}(t) - K^*(t)\big),$$

the learned closed-loop matrix is a small perturbation of the nominal exponentially stable one whenever the DeepONet approximation error is sufficiently small. Standard robustness of exponential stability for linear time-varying systems then yields the existence of $\varepsilon^* > 0$ such that, if (4.5) holds with $\varepsilon < \varepsilon^*$, the learned closed-loop system remains uniformly exponentially stable. □

Theorem 4.1 shows that the learned controller inherits the key qualitative properties of the optimal solution, provided the approximation error of the Riccati operator is sufficiently small. In particular, the result establishes robustness of exponential stability with respect to operator perturbations, together with quantitative bounds on the induced performance degradation. This goes beyond standard approximation guarantees by providing a control-theoretic interpretation of the learned operator, ensuring that its use within a feedback loop remains reliable.

5. Numerical experiments

In this section, we present numerical simulations to demonstrate the effectiveness of DeepONets in solving LQR problems. Following our discussion in Section 3, both time-independent and time-dependent parameters will be considered.

All simulations in this paper were conducted on a laptop running Windows 11 Home (version 24H2, OS build 26100.7840), equipped with an Intel® Core™ Ultra 7 255HX CPU (20 cores and 20 threads) and 32 GB of RAM. The system model is an HP OMEN MAX Gaming Laptop 16-ah0xxx, which includes an NVIDIA GeForce RTX 5060 Laptop GPU with 4 GB of dedicated memory. The implementation was developed in Python 3.13.9 using PyTorch 2.8.0+cu129.

5.1. **Time-independent parameters.** We shall focus here on the case of time-independent matrices (A, B, Q, R), for which the DRE (2.3) reduces to the ARE (3.1). Our objective is to learn the solution operator $\mathcal{F} : (A, B, Q, R) \mapsto P$.

5.1.1. *3-dimensional case.* In a first round of experiments, we have focused on a simple 3-dimensional case. As we commented before, to learn properly $\mathcal{F}$ and achieve good generalization performances, the quality and coverage of the training dataset are crucial.

The LQR weighting matrices Q and R are chosen diagonal to penalize states and control inputs independently. For what concerns the dynamics matrices A and B, instead, to construct representative training samples we exploit the Brunovsky canonical form (3.2)-(3.3).

In dimension $n = 3$, this reduces to the following three cases.

Case1. Single-input controllable form:

$$A = \begin{bmatrix} 0 & 1 & 0 \\ 0 & 0 & 1 \\ -a_0 & -a_1 & -a_2 \end{bmatrix}, \quad B = \begin{bmatrix} 0 \\ 0 \\ 1 \end{bmatrix}, \quad Q = \begin{bmatrix} q_1 & 0 & 0 \\ 0 & q_2 & 0 \\ 0 & 0 & q_3 \end{bmatrix}, \quad R = \begin{bmatrix} r_1 \end{bmatrix}.$$

Case 2. Decoupled mixed-order form:

$$A = \begin{bmatrix} 0 & 1 & 0 \\ -a_0 & -a_1 & 0 \\ 0 & 0 & -b_0 \end{bmatrix}, \quad B = \begin{bmatrix} 0 & 0 \\ 1 & 0 \\ 0 & 1 \end{bmatrix}, \quad Q = \begin{bmatrix} q_1 & 0 & 0 \\ 0 & q_2 & 0 \\ 0 & 0 & q_3 \end{bmatrix}, \quad R = \begin{bmatrix} r_1 & 0 \\ 0 & r_2 \end{bmatrix}.$$

Case 3. Fully decoupled first-order form:

$$A = \begin{bmatrix} -a_0 & 0 & 0 \\ 0 & -b_0 & 0 \\ 0 & 0 & -c_0 \end{bmatrix}, \quad B = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}, \quad Q = \begin{bmatrix} q_1 & 0 & 0 \\ 0 & q_2 & 0 \\ 0 & 0 & q_3 \end{bmatrix}, \quad R = \begin{bmatrix} r_1 & 0 & 0 \\ 0 & r_2 & 0 \\ 0 & 0 & r_3 \end{bmatrix}.$$

To ensure a comprehensive and representative dataset, we generated sufficient random samples for each of the above structural types. In particular, for each configuration, system parameters are randomly sampled from uniform distributions:

$$a_i \sim \mathcal{U}(-2, 2), \quad q_j, r_j \sim \mathcal{U}(0.1, 1.1).$$

Moreover, the coefficients a_i are sampled to cover the following spectral characterization of A:

1. Fully stable: all eigenvalues have negative real parts.
2. Fully unstable: all eigenvalues have positive real parts.
3. Partially stable: some eigenvalues have negative real parts and others positive.

A total of 15000 samples are generated per trial, with an 80%–20% train-test split. The ground-truth solutions of the ARE are computed using a standard CARE solver. The DeepONet model we employed consists of

- a branch network with 5 layers of hidden dimensions 36, 512, 256, 128, and 256, respectively;
- a trunk network with 4 layers of hidden dimensions 2, 128, 256, and 256, respectively.

Training has been performed for 1500 epochs using the Adam optimizer.

Performance evaluation. We assess the quality of the learned operator through two complementary metrics, reflecting both control performance and approximation accuracy:

1. **closed-loop stability**, evaluating whether the feedback law induced by the learned operator preserves stability of the controlled system, which is the key property from a control-theoretic perspective;
2. **test loss**, measuring the discrepancy between the predicted and true Riccati solutions, and providing a quantitative indicator of operator approximation accuracy.

This dual evaluation allows us to simultaneously quantify how well the operator is approximated and how reliably the resulting controller behaves when deployed in closed loop.

Table 1 summarizes the prediction performance over 10 independent trials. The results demonstrate a high level of robustness of the learned controller: in 7 out of 10 trials, all test samples are successfully stabilized, while in the reaming 3 trials only a single unstable case is observed. This corresponds to a stability rate exceeding 99.99%, indicating that the learned operator preserves the qualitative behavior of the optimal feedback law across a wide range of systems. The test loss remains consistently low, confirming that this robustness is achieved together with accurate operator approximation.

Trial No.	**Stable samples**	**Unstable samples**	**Stable eigenvalues**	**Unstable eigenvalues**	**Test loss**
1	3000	0	9000	0	4.93×10^{-3}
2	3000	0	9000	0	3.62×10^{-3}
3	2999	1	8999	1	6.21×10^{-3}
4	3000	0	9000	0	4.95×10^{-3}
5	3000	0	9000	0	4.74×10^{-3}
6	3000	0	9000	0	5.70×10^{-3}
7	2999	1	8999	1	5.11×10^{-3}
8	3000	0	9000	0	4.37×10^{-3}
9	2999	1	8999	1	5.60×10^{-3}
10	3000	0	9000	0	9.50×10^{-3}
Mean					5.47×10^{-3}
Std					1.58×10^{-3}

TABLE 1. DeepONet prediction results over 10 independent trials in dimension $n = 3$ (test set: 3000 samples per trial).

These results show that the learned operator yields a feedback law that is robust to approximation errors, preserving closed-loop stability in virtually all cases while maintaining good quantitative accuracy.

To further investigate the failure mechanism, we focus on Trial 3, which contains the only unstable sample. Figure 1 presents the eigenvalue distributions of the true and predicted closed-loop systems for this trial.

Figure 1 show the eigenvalue distributions of A^{cl} and $\widehat{A}^{\mathrm{cl}}$, respectively. Under the true control law, all eigenvalues lie strictly in the left half-plane, confirming closed-loop stability. For the predicted control law, the eigenvalues closely match the true distribution and remain in the left half-plane for almost all samples. However, one eigenvalue is observed to shift slightly into the right half-plane, resulting in the only unstable case.

A closer examination reveals that, for this sample, the approximation error $\|E\|$ exceeds the threshold ε^* identified in Theorem 4.1. This leads to the observed deviation in the eigenvalue location and explains the loss of stability. Importantly, this behavior provides an empirical validation of the theoretical prediction: instability occurs precisely when the conditions ensuring stability preservation are violated. In this sense, the failure case is not anomalous but rather confirms the sharpness and relevance of the theoretical bound.

Overall, the stable-sample prediction accuracy exceeds 99.99%, while the rare failure case is fully explained by the theoretical error threshold. This confirms that the learned operator behaves in accordance with the control-theoretic guarantees established in Section 4.

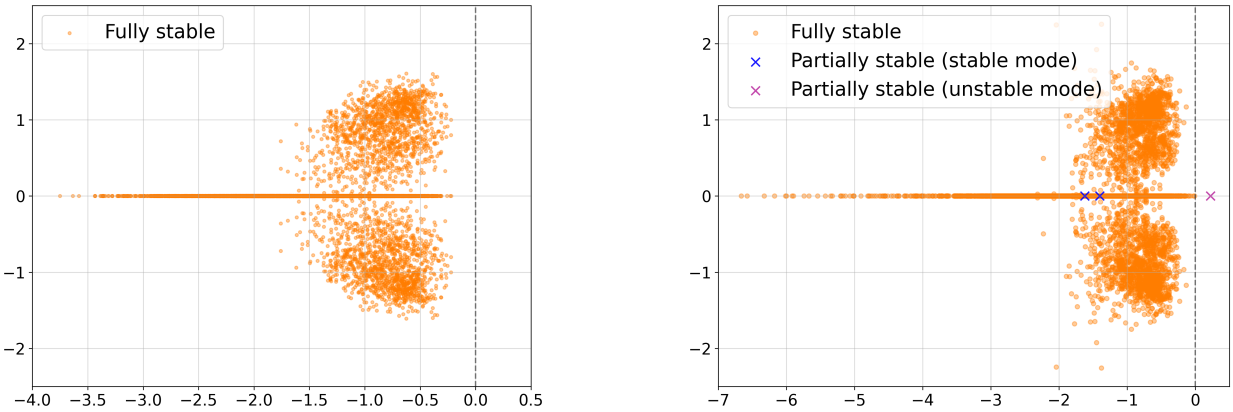

FIGURE 1. Eigenvalue distribution for 3-d LQR problems. On the left we have the true closed-loop matrix A^{cl} and on the right the DeepONet approximated one $\widehat{A}^{\mathrm{cl}}$.

5.1.2. *Progressive DeepONet architecture: 4-dimensional case.* As mentioned in Section 3.3, a central challenge in OL is scalability with respect to the system dimension. To address this issue, we have proposed a Progressive OL strategy allowing to transfer knowledge learned in low-dimensional settings to higher-dimensional problems.

In this section, we assess the performance of the progressive DeepONet by transferring our model trained in dimension $n = 3$ to the case $n = 4$.

Following the same data generation procedure as in the 3-d case, we consider five Brunovsky types ($[4]$, $[3, 1]$, $[2, 2]$, $[2, 1, 1]$, and $[1, 1, 1, 1]$) and generate samples covering stable, unstable, and mixed dynamics. A total of 19000 training samples and 1000 test samples are used. We compared two models under identical settings:

1. the proposed **progressive DeepONet**, which only trains the embedding and extension layers;
2. a **standard 4-d DeepONet** trained from scratch with architecture $64 \to 512 \to 256 \to 128 \to 256$ (branch) and $2 \to 128 \to 256 \to 256$ (trunk).

Both models are trained for 3000 epochs using Adam with learning rate 10^{-4}.

As before, we evaluated the performances using closed-loop stability and prediction accuracy. The standard 4-d DeepONet achieves a stable-sample accuracy of 97.80%, while the progressive DeepONet achieves 97.50%. At the eigenvalue level, the accuracy is 99.42% and 99.38%, respectively.

The marginal performance gap indicates that a substantial portion of the complexity of the Riccati operator can be captured in a low-dimensional representation learned in the 3-d setting. In this sense, the progressive strategy provides empirical evidence that the operator admits a structured representation in which its essential features can be encoded in a reduced latent space and transferred across dimensions. From a control perspective, these results further confirm that the learned operator retains robustness properties when transferred across dimensions. Despite the dimensional increase and the reduced number of trainable parameters, the progressive architecture preserves closed-loop stability with high reliability, indicating that the essential stability-inducing structure of the Riccati operator is captured in the learned representation.

At the same time, Table 2 shows that the progressive DeepONet significantly reduces computational cost, with a reduction of approximately 95% in the number of trainable parameters and about 77% in training time. These gains, obtained with only a marginal loss in stability accuracy, further support the interpretation that the Riccati operator may exhibit an intrinsic low-dimensional structure in the considered regimes, that can be effectively exploited through a latent representation.

Aspect	4-d DeepONet	Progressive DeepONet	Reduction
Parameter count	494.8k	22.4k	95.47%
Training time	271.81s	61.81s	77.25%
Stable sample accuracy	97.80%	97.50%	
Stable eigenvalue accuracy	99.42%	99.38%	
Test loss	4.44×10^{-2}	8.74×10^{-2}	

TABLE 2. Performance and computational resource comparison for 4-dimensional systems.

These results show that the progressive architecture achieves accuracy comparable to a standard DeepONet trained from scratch, while using significantly fewer trainable parameters and requiring substantially less training time. More precisely, the progressive model retains a high level of closed-loop stability and prediction accuracy, with only a marginal degradation compared to the baseline model. At the same time, it reduces the number of trainable parameters by an order of magnitude and accelerates convergence, demonstrating a clear advantage in terms of computational efficiency.

Overall, the proposed progressive DeepONet provides an efficient and scalable framework for approximating high-dimensional Riccati solution operators. Beyond computational efficiency, the results suggest that the operator may exhibit structural regularities across dimensions that can be captured through a reduced latent representation, even though a rigorous theoretical justification of this phenomenon remains an open question.

5.1.3. *10-dimensional case.* To further evaluate the scalability of the proposed OL framework, we extend our experiments to 10-d systems.

Let us stress that, when $n = 10$, there are 42 possible Brunovsky structural configurations of the matrix A. Generating and training a dataset covering all such cases is computationally prohibitive and unnecessary for our purposes. Instead, we chose to focus on the following particular Brunovsky canonical form

$$A = \begin{bmatrix} 0 & 1 & 0 & \cdots & 0 \\ 0 & 0 & 1 & \cdots & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & 0 & \cdots & 1 \\ -a_0 & -a_1 & -a_2 & \cdots & -a_9 \end{bmatrix}, \quad B = \begin{bmatrix} 0 \\ 0 \\ \vdots \\ 0 \\ 1 \end{bmatrix}, \quad Q = \begin{bmatrix} q_1 & 0 & 0 & \cdots & 0 \\ 0 & q_2 & 0 & \cdots & 0 \\ \vdots & & 0 & \cdots & \vdots \\ 0 & 0 & \cdots & q_9 & 0 \\ 0 & 0 & 0 & \cdots & q_{10} \end{bmatrix}, \quad R = [r_1],$$

with $q_i \geq 0$ and $r_1 > 0$. The coefficients a_i are once again randomly sampled to generate systems with stable, unstable, and mixed dynamics.

Due to the increased computational cost in high dimensions, we construct a moderately sized dataset consisting of 6000 training samples (3000 stable and 3000 unstable) and 200 test samples (100 stable and 100 unstable). A compact DeepONet architecture is adopted, with branch network ($211 \to 256 \to 64$ and trunk network $2 \to 128 \to 64$, trained for 1000 epochs using Adam with learning rate 10^{-4}.

We evaluate performance using the same stability-based criteria as in the 3-d case. Figure 2 presents the eigenvalue distributions and Table 3 summarizes the results on the 10-d test set. The model achieves a stable-sample accuracy of 98.50%, while the eigenvalue-level accuracy reaches 99.70%.

Notably, all unstable predictions correspond to systems that do not satisfy the assumptions underlying Theorem 4.1, in particular the conditions ensuring stability of the nominal closed-loop system. This observation is consistent with the theoretical framework, as stability preservation is

guaranteed only within the admissible class $\mathcal{X}$. Outside this regime, no such guarantee is expected, and the observed behavior aligns with this limitation.

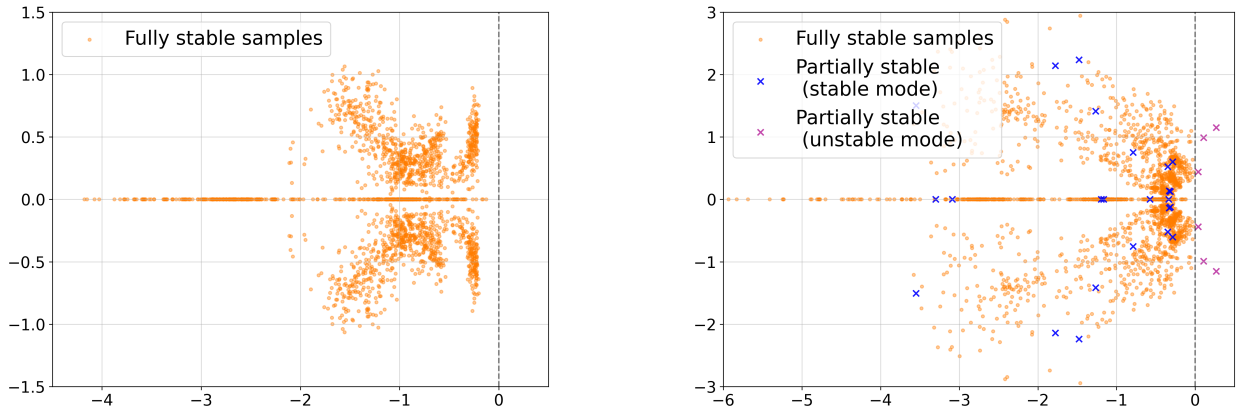

FIGURE 2. Eigenvalue distribution for 10-d LQR problems. On the left we have the true closed-loop matrix A^{cl} and on the right the DeepONet approximated one $\widehat{A}^{\mathrm{cl}}$.

Stable samples	Unstable samples	Stable eigenvalues	Unstable eigenvalues	Test loss
197	3	1994	6	8.77×10^{-3}

TABLE 3. Summary of DeepONet prediction results for 10-d (test set: 200 samples).

Overall, our numerical results consistently support the theoretical framework developed in Section 4. Across all tested regimes, the learned operator preserves the stability properties of the optimal feedback law with very high probability, even in the presence of non-negligible approximation errors. Moreover, the rare failure cases occur precisely when the theoretical assumptions are violated or when the approximation error exceeds the predicted threshold. This provides strong empirical evidence that the proposed approach not only achieves accurate operator approximation, but also ensures reliable control performance in a robust and theoretically predictable manner.

5.2. **Time-dependent parameters.** In this section, we evaluate the proposed DeepONet framework for learning the solution operator of time-varying DREs:

$$\mathcal{F} : \big(A(\cdot), B(\cdot), Q(\cdot), R(\cdot), P_T\big) \mapsto P(\cdot),$$

where $P(t)$ denotes the unique symmetric positive definite solution of (2.3). As for the time-independent case before, we have performed simulations for 3- and 10-dimensional models.

5.2.1. *Data generation, DeepONet architecture and experimental settings.* For a system with state dimension n and control dimension m, the interval $[0, 1]$ is discretized into N_t uniform time steps. The time-varying matrices $A(t)$, $B(t)$, and $Q(t)$ are parameterized via a finite trigonometric basis as in (3.4).

All coefficients $A_0, B_0, M_0, C_{ki}, D_{ki}, E_{ki}$ with $k = 1, 2$ and $i = 1, \ldots, r_{\mathrm{base}}$ are sampled from $\mathcal{N}(0, 0.05^2)$. Here we have set $r_{\mathrm{base}} = 2$. Moreover, to ensure that our dataset fulfills the necessary conditions for the existence of a LQR control, we applied a filtering procedure and retain only those matrices that satisfy stabilizability of $(A(t), B(t))$ and detectability of $(A(t), Q^{1/2}(t))$ at all discrete

time steps. The control weighting matrix R, instead, is assumed to be constant: $R(t) = R = I_m$ for all $t \in [0, 1]$.

For the 3-d case, a dataset of 1000 valid trajectories is obtained with the above procedure. For the 10-d case, instead, our dataset is constituted by 3000 valid trajectories.

The terminal condition is fixed as $P_T = I_n$. Each trajectory is numerically solved via a backward fourth-order Runge-Kutta scheme (RK4).

As for the architecture, for the 3-d case we considered a DeepONet with branch and trunk networks having inner dimensions $[256, 256, 128]$ and $[128, 128, 128]$, respectively, both with GELU activation. The model is trained for 60 epochs using Adam with learning rate 1×10^{-3}.

For the 10-d case, instead, we considered a DeepONet with input dimension $\overline{m} = 1375$. The branch and trunk networks have inner dimensions $[1024, 1024, 512, 512]$ and $[256, 512, 512]$, respectively, both with GELU activation. The model is trained for 150 epochs using Adam with learning rate 10^{-4}.

Finally, to quantify prediction accuracy for the closed-loop system, we define the following error metrics.

Riccati solution errors

$$e_P = \frac{1}{K}\sum_{k=1}^{K}\left(\int_0^T \|P^{(k)}(t) - \widehat{P}^{(k)}(t)\|_F^2\, dt\right)^{\frac{1}{2}} \qquad \text{absolute error}$$

$$e_P^r = \frac{1}{K}\sum_{k=1}^{K}\left(\int_0^T \|P^{(k)}(t)\|_F^2\, dt\right)^{-\frac{1}{2}}\left(\int_0^T \|P^{(k)}(t) - \widehat{P}^{(k)}(t)\|_F^2\, dt\right)^{\frac{1}{2}} \qquad \text{relative error}$$

State errors

$$e_x = \frac{1}{K}\sum_{k=1}^{K}\left(\int_0^T \|x^{(k)}(t) - \widehat{x}^{(k)}(t)\|_2^2\, dt\right)^{\frac{1}{2}} \qquad \text{absolute error}$$

$$e_x^r = \frac{1}{K}\sum_{k=1}^{K}\left(\int_0^T \|x^{(k)}(t)\|_2^2\, dt\right)^{-\frac{1}{2}}\left(\int_0^T \|x^{(k)}(t) - \widehat{x}^{(k)}(t)\|_2^2\, dt\right)^{\frac{1}{2}} \qquad \text{relative error}$$

Cost functional errors

$$e_J = \frac{1}{K}\sum_{k=1}^{K}\left|J^{(k)} - \widehat{J}^{(k)}\right| \qquad \text{absolute error}$$

$$e_J^r = \frac{1}{K}\sum_{k=1}^{K}\frac{\left|J^{(k)} - \widehat{J}^{(k)}\right|}{\left|J^{(k)}\right|} \qquad \text{relative error}$$

Here, K denotes the number of test trajectories. For each trajectory indexed by $k = 1, \ldots, K$, $P^{(k)}(t)$, $x^{(k)}(t)$, and $J^{(k)}$ represent the ground-truth Riccati solution, state trajectory, and cost functional, respectively, while $\widehat{P}^{(k)}(t)$, $\widehat{x}^{(k)}(t)$, and $\widehat{J}^{(k)}$ denote their corresponding DeepONets predictions.

5.2.2. *Results.* To evaluate the generalization capability of the trained model, we generate $K = 100$ test trajectory from a random instance not seen during training. The results in Table 4 show that DeepONet accurately predicts $P(t)$, $x(t)$, and J, with errors slightly increasing from 3-d to 10-d but remaining reasonably small, confirming good generalization across dimensions.

Error metric	$d = 3$	$d = 10$
Test loss	4.58×10^{-6}	2.310×10^{-5}
e_P	2.92×10^{-2}	2.72×10^{-1}
e_P^r	1.66×10^{-2}	8.19×10^{-2}
e_x	1.59×10^{-4}	6.76×10^{-3}
e_x^r	1.06×10^{-4}	2.22×10^{-3}
e_J	9.26×10^{-6}	2.67×10^{-3}
e_J^r	2.90×10^{-6}	2.48×10^{-4}

TABLE 4. Average absolute and relative errors for $P(t)$, state $x(t)$, and cost J in different dimensions.

Moreover, Figure 3 illustrates the state trajectories $x(t)$ and corresponding control inputs $u(t)$ for a representative 3-d test system under both the true Riccati and DeepONet-based controllers. The results demonstrate that the learned operator accurately reproduces the system behavior and feedback control in both settings, confirming its effectiveness and generalization capability.

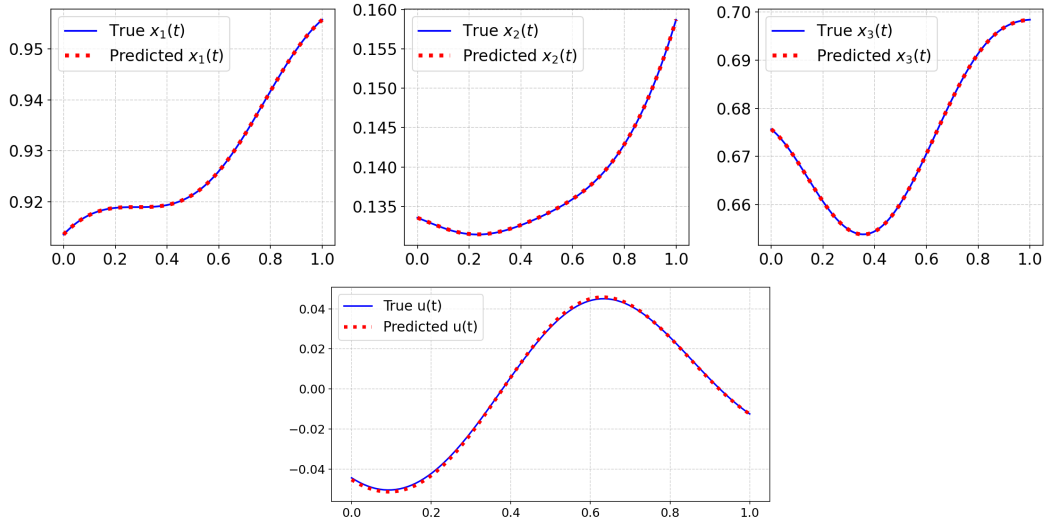

FIGURE 3. Comparison of predicted and true trajectories (top) and predicted and tru controls (bottom) for a 3-d test system.

5.3. **Computational complexity and efficiency.** We conclude the numerical analysis by providing a unified assessment of the computational cost of the proposed approach, integrating the results obtained for both time-invariant (ARE) and time-dependent (DRE) settings. The objective is to quantify the cost of training the operator, the efficiency of its evaluation, and the resulting computational gain with respect to classical Riccati solvers.

Classical numerical methods for solving Riccati equations, such as CARE solvers in the time-invariant case and backward integration schemes for the differential Riccati equation, rely on dense linear algebra operations, including matrix factorizations and multiplications. As a consequence, their computational cost scales cubically with the state dimension, that is, of order $\mathcal{O}(n^3)$ per instance. When optimal control problems must be solved repeatedly for varying system parameters, the total

Learned solution	Dimension	Training time (s)
ARE	$n = 3$	42.63
	$n = 4$	271.81
	$n = 4$ progressive	61.81
	$n = 10$	382.82
DRE	$n = 3$	804.59
	$n = 10$	29951.34

TABLE 5. Offline training time for all the DeepONet models considered in our experiments.

Dimension	Method	Time (ms)	Speedup
$n = 3$	CARE	0.51632	1×
	DeepONet	0.00137	376.86×
$n = 4$	CARE	0.56849	1×
	DeepONet	0.00197	287.72×
	DeepONet-Progressive	0.00157	360.58×
$n = 10$	CARE	0.87370	1×
	DeepONet	0.00150	582.47×

TABLE 6. Comparison of computational efficiency for solving ARE using CARE and DeepONet

cost therefore grows linearly with the number of instances, which becomes a major limitation in high-dimensional or real-time applications.

In contrast, the DeepONet-based framework follows an offline–online computational strategy. The offline phase consists of generating training data by solving multiple Riccati equations and optimizing the network parameters. This phase incurs a non-negligible one-time cost. Once the model is trained, however, the evaluation of the learned operator for a new parameter instance only requires a forward pass through the neural network. This inference step is computationally inexpensive and largely independent of the complexity of the underlying Riccati equation.

The magnitude of the offline training cost depends on both the problem dimension and the nature of the system parameters. The training time for all the configurations considered in our experiments a re collected in Table 5.

While in the case of the ARE the training time is relatively small, for the DRE the cost increases significantly due to the time-dependent nature of the problem, reaching a pick of approximately 8.3 hours in dimension $n = 10$. These results illustrate the higher computational burden associated with learning time-dependent operators, while remaining tractable within an offline setting.

Once trained, the computational advantage of the approach becomes evident at inference time. As reported in Tables 6 and 7, the evaluation of the learned operator is up to several hundred times faster than solving the Riccati equation using classical CARE or RK4 routines.

This speedup is consistent across dimensions and reflects the fundamental difference between the two approaches: while classical solvers recompute the solution from scratch for each instance, the DeepONet model directly evaluates a learned mapping.

Overall, these results show that the proposed OL framework provides a substantial computational advantage in scenarios involving multiple problem instances. By shifting the main computational

Dimension	Method	Time (ms)	Speedup
$n = 3$	RK4	11.1564	1×
	DeepONet	0.0065	1723×
$n = 10$	RK4	52.2292	1×
	DeepONet	0.2484	210×

TABLE 7. Comparison of computational efficiency for solving DRE using RK4 and DeepONet

effort to an offline training phase and enabling fast evaluation at inference, the method allows for scalable and efficient approximation of Riccati solution operators across a wide range of system configurations.

6. Conclusions

In this work, we introduced an OL framework for the efficient approximation of the solution map associated with the differential Riccati equation arising in finite-horizon LQR problems. By recasting the Riccati equation as a nonlinear operator mapping time-dependent system parameters to matrix-valued trajectories, we leveraged DeepONets to learn this mapping in a data-driven yet structure-aware manner.

From a theoretical standpoint, we established a rigorous link between the approximation error of the learned operator and the performance of the induced feedback controller. In particular, we derived quantitative bounds on the suboptimality gap and proved that exponential stability of the closed-loop system is preserved under sufficiently accurate operator approximation. These results provide a principled framework to assess the reliability of neural operator approaches in optimal control.

On the computational side, we designed tailored DeepONet architectures capable of handling matrix-valued, time-dependent outputs, and introduced a progressive learning strategy to address scalability across dimensions. Numerical experiments demonstrated that the proposed approach achieves high accuracy, excellent stability preservation, and substantial computational speedups compared to classical Riccati solvers, especially in high-dimensional or repeated-query regimes.

Overall, our results highlight the potential of OL as a viable paradigm for amortized optimal control, enabling real-time evaluation of feedback laws in complex and parametric settings. Beyond the specific LQR setting, this work supports a broader perspective in which OL provides a systematic way to construct reusable surrogates for nonlinear differential equations arising in control and dynamical systems. This approach has the potential to significantly reduce computational cost in applications requiring repeated evaluation of parametrized models.

The results presented in this work open several avenues for further investigation at the interface of OL and optimal control.

1. A first fundamental question concerns the generalization properties of neural operators in control settings. While our analysis provides a posteriori bounds linking the approximation error of the learned Riccati operator to the performance of the induced controller, a deeper understanding of generalization remains largely open. In particular, deriving a priori estimates on the sample complexity required to achieve a prescribed control accuracy, and characterizing how such requirements scale with the system dimension, the time horizon, and the regularity of the system coefficients, constitute important directions for future research.
2. Another relevant challenge is the design of structure-preserving learning architectures. The solution of the Riccati equation enjoys intrinsic properties such as symmetry and positive

definiteness, which play a crucial role in ensuring stability of the closed-loop system. In the present work, these properties are only approximately satisfied by the learned operator, and violations may occur when the approximation error exceeds the threshold identified in Theorem 4.1. Developing neural operator architectures that enforce such structural constraints by construction, for instance through suitable parameterizations of symmetric positive definite matrices or geometry-aware network designs, is a key open problem.

3. Extending the proposed framework beyond the linear-quadratic setting also represents a natural and challenging direction. In particular, applying OL techniques to nonlinear optimal control problems would require approximating the solution operators of Hamilton–Jacobi–Bellman equations, which are fully nonlinear and may lack regularity. Similarly, incorporating state and control constraints, as in model predictive control, would significantly increase the complexity of the associated operators and raise new theoretical and computational issues.
4. From the perspective of scalability, the progressive OL strategy introduced in this work provides an effective empirical approach to transfer knowledge across dimensions. However, its theoretical justification remains limited. An important open question is whether the Riccati solution operator admits a low-dimensional or dimension-independent representation, possibly in terms of reduced-order structures or compact embeddings of the parameter space. Establishing such properties would provide a rigorous foundation for scalable OL in high-dimensional control problems.

Appendix A. Proof of Theorem 4.1

Proof. We prove the claims separately.

Trajectory error. Define the optimal closed-loop matrix

$$A_{cl}^*(t) \coloneqq A(t) - B(t)K^*(t). \tag{A.1}$$

Using that

$$\begin{aligned}\widehat{K}(t) &= R^{-1}(t)B^\top(t)\widehat{P}(t) = R^{-1}(t)B^\top(t)P^*(t) + R^{-1}(t)B^\top(t)\big(\widehat{P}(t) - P^*(t)\big)\\ &= K^*(t) + R^{-1}(t)B^\top(t)E(t),\end{aligned}$$

we can readily check that the error $e(t) \coloneqq \widehat{x}(t) - x^*(t)$ follows the dynamics

$$\begin{cases}\dot{e}(t) = A_{cl}^*(t)e(t) - B(t)R^{-1}(t)B^\top(t)E(t)\widehat{x}(t), \quad t \in [0,T]\\ e(0) = 0.\end{cases}$$

Consider the Lyapunov function $V_e(t) \coloneqq e(t)^\top P^*(t)e(t)$. Differentiating along the solutions of (2.3) gives

$$\begin{aligned}\dot{V}_e(t) &= \dot{e}^\top(t)P^*(t)e(t) + e^\top(t)\dot{P}^*(t)e(t) + e^\top(t)P^*(t)\dot{e}(t)\\ &= e^\top(t)\dot{P}^*(t)e(t) + 2e^\top(t)P^*(t)\dot{e}(t)\\ &= e^\top(t)\dot{P}^*(t)e(t) + 2e^\top(t)P^*(t)\Big(A_{cl}^*(t)e(t) - B(t)R^{-1}(t)B^\top(t)E(t)\widehat{x}(t)\Big)\\ &= e^\top(t)\dot{P}^*(t)e(t) + 2e^\top(t)P^*(t)A_{cl}^*(t)e(t) - 2e^\top(t)P^*(t)B(t)R^{-1}(t)B^\top(t)E(t)\widehat{x}(t).\end{aligned}$$

Moreover, (2.3) and (A.1) yield

$$\begin{aligned}e^\top(t)\dot{P}^*(t)e(t) &= e^\top(t)\Big(-A^\top(t)P^*(t) - P^*(t)A(t) + P^*(t)B(t)R^{-1}(t)B^\top(t)P^*(t) - Q(t)\Big)e(t),\\ 2e^\top(t)P^*(t)A_{cl}^*(t)e(t) &= 2e^\top(t)P^*(t)\Big(A(t) - B(t)R^{-1}(t)B^\top(t)P^*(t)\Big)e(t),\end{aligned}$$

so that we obtain

$$\dot{V}_e(t) = -e^\top(t)\Big(Q(t) + (K^*)^\top(t)R(t)K^*(t)\Big)e(t) - 2e^\top(t)P^*(t)B(t)R^{-1}(t)B^\top(t)E(t)\widehat{x}(t).$$

Since $Q + (K^*)^\top RK^* \succeq 0$ uniformly on $[0,T]$, it follows that

$$\dot{V}_e(t) \le 2\|P^*(t)\|_F\|B(t)\|_F^2\|R^{-1}(t)\|_F\|E(t)\|_F\|e(t)\|_2\|\widehat{x}(t)\|_2. \tag{A.2}$$

Now, by continuity of P^* on $[0,T]$, there exist two positive constants $0 < \gamma_{\min} < \gamma_{\max}$ such that

$$\gamma_{\min} I_n \preceq P^*(t) \preceq \gamma_{\max} I_n.$$

Hence $V_e \ge \gamma_{\min}\|e\|_2^2$, which implies

$$\|e\|_2 \le \sqrt{\frac{V_e}{\gamma_{\min}}}. \tag{A.3}$$

Moreover, since x^* is uniformly bounded on $[0,T]$, i.e., there exist constants $M_0 > 0$ such that

$$\|x^*(t)\|_2 \le M_0\|x_0\|_2.$$

From $\widehat{x}(t) = x^*(t) + e(t)$ we therefore have

$$\|\widehat{x}(t)\|_2 \le \|x^*(t)\|_2 + \|e(t)\|_2 \le M_0\|x_0\|_2 + \|e(t)\|_2.$$

In view of this, we get from (A.2) that

$$\dot{V}_e(t) \le c_1\|E(t)\|_F\|e(t)\|_2\Big(M_0\|x_0\|_2 + \|e(t)\|_2\Big),$$

where we have defined

$$c_1 := 2\sup_{t\in[0,T]}\Big(\|B(t)\|_F^2\|R^{-1}(t)\|_F\|P^*(t)\|_F\Big). \tag{A.4}$$

Then, we obtain from (A.3) that

$$\dot{V}_e(t) \le c_1\|E(t)\|_F\sqrt{\frac{V_e(t)}{\gamma_{\min}}}\left(M_0\|x_0\|_2 + \sqrt{\frac{V_e(t)}{\gamma_{\min}}}\right) \le \alpha\|E(t)\|_F\sqrt{V_e(t)} + \beta\|E(t)\|_F V_e(t),$$

where we have set

$$\alpha = \frac{c_1 M_0\|x_0\|_2}{\sqrt{\gamma_{\min}}} \quad \text{and} \quad \beta = \frac{c_1}{\gamma_{\min}}. \tag{A.5}$$

This yields

$$\begin{aligned}\frac{d}{dt}\sqrt{V_e(t)} = \frac{\dot{V}_e(t)}{2\sqrt{V_e(t)}} &\le \frac{1}{2\sqrt{V_e(t)}}\Big(\alpha\|E(t)\|_F\sqrt{V_e(t)} + \beta\|E(t)\|_F V_e(t)\Big)\\ &= \frac{\alpha}{2}\|E(t)\|_F + \frac{\beta}{2}\|E(t)\|_F\sqrt{V_e(t)}.\end{aligned}$$

Setting $y(t) = \sqrt{V_e(t)}$, the above estimate becomes

$$\dot{y}(t) \le \frac{\alpha}{2}\|E(t)\|_F + \frac{\beta}{2}\|E(t)\|_F y(t).$$

Using the Grönwall's lemma, combined with $y(0) = 0$, we then obtain

$$\begin{aligned}y(t) &\le \frac{\alpha}{2}\int_0^t \exp\left(\frac{\beta}{2}\int_s^t \|E(\tau)\|_F\,d\tau\right)\|E(s)\|_F\,ds \le \frac{\alpha\varepsilon}{2}\int_0^t e^{\frac{\beta}{2}\varepsilon(t-s)}\,ds\\ &= \frac{\alpha}{\beta}\left(e^{\frac{\beta\varepsilon t}{2}} - 1\right) = M_0\|x_0\|_2\sqrt{\gamma_{\min}}\left(e^{\frac{\beta\varepsilon t}{2}} - 1\right),\end{aligned}$$

with ε given by (4.5). Then we get from (A.3) and the definition (A.5) of β

$$\|e(t)\|_2 \le \sqrt{\frac{V_e(t)}{\gamma_{\min}}} = \frac{y(t)}{\sqrt{\gamma_{\min}}} \le M_0\|x_0\|_2\left(e^{\frac{\beta\varepsilon t}{2}} - 1\right) = M_0\|x_0\|_2\left(e^{\frac{c_1\varepsilon t}{2\gamma_{\min}}} - 1\right),$$

that is,

$$\sup_{t\in[0,T]} \|\widehat{x}(t) - x^*(t)\|_2 \le M_0\|x_0\|_2\left(e^{\frac{c_1\varepsilon T}{2\gamma_{\min}}} - 1\right) \tag{A.6}$$

with c_1 given by (A.4).

Cost error. Consider now the Lyapunov function $V(t) := \widehat{x}(t)^\top P^*(t)\widehat{x}(t)$. Differentiating along the trajectory $\widehat{x}(t)$, and since $P^*(t)$ is symmetric, we get

$$\begin{aligned}\dot{V}(t) &= \dot{\widehat{x}}(t)^\top P^*(t)\widehat{x}(t) + \widehat{x}(t)^\top \dot{P}(t)^*\widehat{x}(t) + \widehat{x}(t)^\top P^*(t)\dot{\widehat{x}}(t)\\ &= \widehat{x}(t)^\top \dot{P}(t)^*\widehat{x}(t) + 2\widehat{x}(t)^\top P^*(t)(A(t)\widehat{x}(t) + B(t)\widehat{u}(t)).\end{aligned}$$

The Riccati equation (2.3) then yields

$$\begin{aligned}\dot{V}(t) =& \widehat{x}(t)^\top\Big(- A(t)^\top P^*(t) - P^*(t)A(t) + P^*(t)B(t)R^{-1}(t)B(t)^\top P^*(t) - Q(t)\Big)\widehat{x}(t)\\ &+ 2\widehat{x}(t)^\top P^*(t)A(t)\widehat{x}(t) + 2\widehat{x}(t)^\top P^*(t)B(t)\widehat{u}(t).\end{aligned}$$

Since

$$\begin{aligned}&-\widehat{x}(t)^\top A(t)^\top P^*(t)\widehat{x}(t) - \widehat{x}(t)^\top P^*(t)A(t)\widehat{x}(t) + 2\widehat{x}(t)^\top P^*(t)A(t)\widehat{x}(t)\\ &= -\Big(\widehat{x}(t)^\top P^*(t)A(t)\widehat{x}(t)\Big)^\top - \widehat{x}(t)^\top P^*(t)A(t)\widehat{x}(t) + 2\widehat{x}(t)^\top P^*(t)A(t)\widehat{x}(t)\\ &= -2\widehat{x}(t)^\top P^*(t)A(t)\widehat{x}(t) + 2\widehat{x}(t)^\top P^*(t)A(t)\widehat{x}(t) = 0,\end{aligned}$$

we obtain from the previous identity that

$$\dot{V}(t) = -\widehat{x}(t)^\top Q(t)\widehat{x}(t) + \widehat{x}(t)^\top P^*(t)B(t)R^{-1}(t)B(t)^\top P^*(t)\widehat{x}(t) + 2\widehat{x}(t)^\top P^*(t)B(t)\widehat{u}(t).$$

Next, recalling that $K^*(t) = R^{-1}(t)B(t)^\top P^*(t)$, we have

$$\begin{aligned}(K^*(t))^\top R(t)K^*(t) &= P^*(t)B(t)R^{-1}(t)B(t)^\top P^*(t)\\ \widehat{u}(t)^\top R(t)K^*(t)\widehat{x}(t) &= \widehat{x}(t)^\top P^*(t)B(t)\widehat{u}(t).\end{aligned}$$

Therefore

$$\begin{aligned}&\Big(\widehat{u}(t) + K^*(t)\widehat{x}(t)\Big)^\top R(t)\Big(\widehat{u}(t) + K^*(t)\widehat{x}(t)\Big)\\ &\quad= \widehat{u}(t)^\top R(t)\widehat{u}(t) + 2\widehat{u}(t)^\top R(t)K^*(t)\widehat{x}(t) + \widehat{x}(t)^\top (K^*(t))^\top R(t)K^*(t)\widehat{x}(t)\\ &\quad= \widehat{u}(t)^\top R(t)\widehat{u}(t) + 2\widehat{x}(t)^\top P^*(t)B(t)\widehat{u}(t) + \widehat{x}(t)^\top P^*(t)B(t)R^{-1}(t)B(t)^\top P^*(t)\widehat{x}(t).\end{aligned}$$

Hence,

$$\begin{aligned}&\widehat{x}(t)^\top P^*(t)B(t)R^{-1}(t)B(t)^\top P^*(t)\widehat{x}(t) + 2\widehat{x}(t)^\top P^*(t)B(t)\widehat{u}(t)\\ &\qquad= \Big(\widehat{u}(t) + K^*(t)\widehat{x}(t)\Big)^\top R(t)\Big(\widehat{u}(t) + K^*(t)\widehat{x}(t)\Big) - \widehat{u}(t)^\top R(t)\widehat{u}(t),\end{aligned}$$

and finally

$$\dot{V}(t) = -\widehat{x}(t)^\top Q(t)\widehat{x}(t) - \widehat{u}(t)^\top R(t)\widehat{u}(t) + \Big(\widehat{u}(t) + K^*(t)\widehat{x}(t)\Big)^\top R(t)\Big(\widehat{u}(t) + K^*(t)\widehat{x}(t)\Big).$$

Equivalently,

$$\widehat{x}(t)^\top Q(t)\widehat{x}(t) + \widehat{u}(t)^\top R(t)\widehat{u}(t) = -\dot{V}(t) + \Big(\widehat{u}(t) + K^*(t)\widehat{x}(t)\Big)^\top R(t)\Big(\widehat{u}(t) + K^*(t)\widehat{x}(t)\Big)$$

and integrating over $[0,T]$ yields

$$\begin{aligned}\int_0^T \Big(\widehat{x}(t)^\top Q(t)\widehat{x}(t) + \widehat{u}(t)^\top R(t)\widehat{u}(t)\Big)\,dt \\ = V(0) - V(T) + \int_0^T \Big(\widehat{u}(t) + K^*(t)\widehat{x}(t)\Big)^\top R(t)\Big(\widehat{u}(t) + K^*(t)\widehat{x}(t)\Big)\,dt.\end{aligned}$$

Since $P^*(T) = P_T$, we then obtain

$$\begin{aligned}J(\widehat{u}) &= \widehat{x}(T)^\top P_T\widehat{x}(T) + \int_0^T \Big(\widehat{x}(t)^\top Q(t)\widehat{x}(t) + \widehat{u}(t)^\top R(t)\widehat{u}(t)\Big)\,dt \\ &= \widehat{x}(T)^\top P_T\widehat{x}(T) + V(0) - V(T) + \int_0^T \Big(\widehat{u}(t) + K^*(t)\widehat{x}(t)\Big)^\top R(t)\Big(\widehat{u}(t) + K^*(t)\widehat{x}(t)\Big)\,dt \\ &= x_0^\top P^*(0)x_0 + \int_0^T \Big(\widehat{u}(t) + K^*(t)\widehat{x}(t)\Big)^\top R(t)\Big(\widehat{u}(t) + K^*(t)\widehat{x}(t)\Big)\,dt.\end{aligned}$$

On the other hand, for the optimal control $u^*(t) = -K^*(t)x^*(t)$, we have

$$J(u^*) = x_0^\top P^*(0)x_0.$$

Therefore,

$$J(\widehat{u}) - J(u^*) = \int_0^T \Big(\widehat{u}(t) + K^*(t)\widehat{x}(t)\Big)^\top R(t)\Big(\widehat{u}(t) + K^*(t)\widehat{x}(t)\Big)\,dt.$$

But since $\widehat{u}(t) = -\widehat{K}(t)\widehat{x}(t)$, it follows that

$$\widehat{u}(t) + K^*(t)\widehat{x}(t) = \Big(K^*(t) - \widehat{K}(t)\Big)\widehat{x}(t) = -R^{-1}(t)B(t)^\top E(t)\widehat{x}(t),$$

and thus

$$\begin{aligned}J(\widehat{u}) - J(u^*) &= \int_0^T \widehat{x}(t)^\top E(t)^\top B(t)R^{-1}(t)B(t)^\top E(t)\widehat{x}(t)\,dt \\ &\le \int_0^T \|\widehat{x}(t)\|_2^2\|E(t)\|_F^2\|B(t)\|_F^2\|R^{-1}(t)\|_F\,dt.\end{aligned}$$

Moreover, from (A.6) we have that

$$\begin{aligned}\|\widehat{x}(t)\|_2 &\le \|x^*(t)\|_2 + \|\widehat{x}(t) - x^*(t)\|_2 \\ &\le M_0\|x_0\|_2 + M_0\|x_0\|_2\left(e^{\frac{c_1\varepsilon T}{2\gamma_{\min}}} - 1\right) = M_0\|x_0\|_2 e^{\frac{c_1\varepsilon T}{2\gamma_{\min}}}.\end{aligned}$$

Hence, using (4.5), we can conclude that

$$J(\widehat{u}) - J(u^*) \le C_3\varepsilon^2 T e^{C_4\varepsilon T}\|x_0\|_2^2,$$

with

$$C_3 \coloneqq M_0^2 \sup_{t\in[0,T]}\Big(\|B(t)\|_F^2\|R^{-1}(t)\|_F\Big) \quad \text{and} \quad C_4 \coloneqq \frac{c_1}{\gamma_{\min}}$$

Stability of the closed-loop system. Define the optimal and approximated closed-loop matrices

$$A_{cl}^*(t) = A(t) - B(t)K^*(t) \quad \text{and} \quad \widehat{A}_{cl}(t) = A(t) - B(t)\widehat{K}(t),$$

and denote their difference

$$\Delta A(t) \coloneqq \widehat{A}_{cl}(t) - A^*_{cl}(t) = -B(t)R^{-1}(t)B(t)^\top E(t).$$

Because $B \in C([0,T];\mathbb{R}^{n\times m})$ and $R \in C([0,T];\Sigma^m_{++})$, both B and R^{-1} are uniformly bounded on $[0,T]$. Therefore there exists a constant $C_0 > 0$, depending only on the problem data, such that

$$\|\Delta A(t)\|_F \le C_0 \|E(t)\|_F, \quad \text{for all } t \in [0,T],$$

and we have from (4.5)

$$\sup_{t\in[0,T]} \|\Delta A(t)\|_F \le C_0\varepsilon.$$

Moreover, since the nominal optimal closed-loop system $\dot{x}(t) = A^*_{cl}(t)x(t)$ is uniformly exponentially stable, there exist constants $M > 0$ and $\mu_0 > 0$, depending only on the problem data, such that its solution operator $\Phi_*(t,s)$ satisfies

$$\|\Phi_*(t,s)\|_2 \le Me^{-\mu_0(t-s)}, \quad \text{for all } 0 \le s \le t \le T. \tag{A.7}$$

Let now $\widehat{x}$ solve the learned closed-loop system

$$\dot{\widehat{x}}(t) = \widehat{A}_{cl}(t)\widehat{x}(t) = \Big(A^*_{cl}(t) + \Delta A(t)\Big)\widehat{x}(t), \quad \widehat{x}(0) = x_0.$$

The variation-of-constants formula gives

$$\widehat{x}(t) = \Phi_*(t,0)x_0 + \int_0^t \Phi_*(t,s)\Delta A(s)\widehat{x}(s)\,ds.$$

Using (A.7) we therefore have

$$\|\widehat{x}(t)\|_2 \le Me^{-\mu_0 t}\|x_0\|_2 + MC_0\varepsilon \int_0^t e^{-\mu_0(t-s)}\|\widehat{x}(s)\|_2\,ds.$$

Set

$$y(t) \coloneqq e^{\mu_0 t}\|\widehat{x}(t)\|_2.$$

Multiplying the previous inequality by $e^{\mu_0 t}$, we obtain

$$y(t) \le M\|x_0\|_2 + MC_0\varepsilon \int_0^t y(s)\,ds.$$

By Grönwall's inequality,

$$y(t) \le M\|x_0\|_2 e^{MC_0\varepsilon t}, \quad \text{for all } t \in [0,T],$$

that is,

$$\|\widehat{x}(t)\|_2 \le Me^{-(\mu_0 - MC_0\varepsilon)t}\|x_0\|_2, \quad \text{for all } t \in [0,T].$$

Finally, if we choose

$$\varepsilon^* \coloneqq \frac{\mu_*}{2MC_0},$$

then

$$\mu \coloneqq \mu_0 - MC_0\varepsilon \ge \frac{\mu}{2} > 0,$$

which immediately yields

$$\|\widehat{x}(t)\|_2 \le Me^{-\mu t}\|x_0\|_2, \quad \text{for all } t \in [0,T].$$

□

References

[1] D. E. Kirk, Optimal control theory: an introduction, Courier Corporation, 2004.

[2] H. Kwakernaak, R. Sivan, Linear optimal control systems, Vol. 1, Wiley-interscience New York, 1972.

[3] E. D. Sontag, Mathematical control theory: deterministic finite dimensional systems, Vol. 6, Springer Science & Business Media, 2013.

[4] E. Trélat, Contrôle optimal: théorie & applications, Vol. 36, Vuibert Paris, 2005.

[5] E. Alcala, V. Puig, J. Quevedo, T. Escobet, R. Comasolivas, Autonomous vehicle control using a kinematic Lyapunov-based technique with LQR-LMI tuning, Control Eng. Pract. 73 (2018) 1--12.

[6] D. Dhingra, K. Kaheman, S. B. Fuller, Modeling and LQR control of insect sized flapping wing robot, npj Robotics 3 (1) (2025) 6.

[7] A. S. Elkhatem, S. N. Engin, Robust LQR and LQR-PI control strategies based on adaptive weighting matrix selection for a UAV position and attitude tracking control, Alex. Eng. J. 61 (8) (2022) 6275--6292.

[8] G. Gu, X.-R. Cao, H. Badr, Generalized LQR control and Kalman filtering with relations to computations of inner-outer and spectral factorizations, IEEE Trans. Automat. Control 51 (4) (2006) 595--605.

[9] X. Peng, J. Yin, J. Yu, J. Song, K. Liu, Z. Li, T. Wei, Linear quadratic regulator-based coordinated optimization for harmonic compensation in multifunctional grid-connected inverters, Int. J. Electr. Power Energy Syst. 172 (2025) 111333.

[10] Z. Su, H. Yao, J. Peng, Z. Liao, Z. Wang, H. Yu, H. Dai, T. C. Lueth, LQR-based control strategy for improving human--robot companionship and natural obstacle avoidance, Biomim. Intell. Robot. 4 (4) (2024) 100185.

[11] S. Bittanti, A. Laub, J. Willems, The Riccati equation, Springer-Verlag, 1991.

[12] P. Lancaster, L. Rodman, Algebraic Riccati equations, Clarendon press, 1995.

[13] S. Lanthaler, S. Mishra, G. E. Karniadakis, Error estimates for deeponets: A deep learning framework in infinite dimensions, Trans. Math. Appl. 6 (1) (2022) 1--141.

[14] L. Lu, P. Jin, G. Pang, Z. Zhang, G. E. Karniadakis, Learning nonlinear operators via DeepONet based on the universal approximation theorem of operators, Nature Mach. Intell. 3 (3) (2021) 218--229.

[15] C. A. O. Quero, J. Martinez-Carranza, Physics-informed machine learning for uav control, in: 2024 21st International Conference on Electrical Engineering, Computing Science and Automatic Control (CCE), IEEE, 2024, pp. 1--6.

[16] M. Raissi, P. Perdikaris, G. E. Karniadakis, Physics-informed neural networks: A deep learning framework for solving forward and inverse problems involving nonlinear partial differential equations, J. Comput. Phys. 378 (2019) 686--707.

[17] L. Cui, B. Pang, M. Krstić, Z.-P. Jiang, Learning-based adaptive optimal control of linear time-delay systems: A value iteration approach, Automatica 171 (2025) 111944.

[18] J. Yu, B.-C. Wang, D. Meng, Stochastic Linear Quadratic Optimal Control for Continuous-Time Systems via Reinforcement Learning, Int. J. Robust Nonlinear Control 36 (4) (2026) 1876--1887.

[19] J. W. Choi, Y. B. Seo, LQR design with eigenstructure assignment capability and application to aircraft flight control, IEEE Trans. Aerosp. Electron. Syst. 35 (2) (1999) 700--708.

[20] B. L. Stevens, F. L. Lewis, E. N. Johnson, Aircraft control and simulation: dynamics, controls design, and autonomous systems, John Wiley & Sons, 2015.

[21] V. Klemm, A. Morra, L. Gulich, D. Mannhart, D. Rohr, M. Kamel, Y. de Viragh, R. Siegwart, LQR-assisted whole-body control of a wheeled bipedal robot with kinematic loops, IEEE Robot. Autom. Lett. 5 (2) (2020) 3745--3752.

[22] E. V. Kumar, J. Jerome, Robust LQR controller design for stabilizing and trajectory tracking of inverted pendulum, Procedia Eng. 64 (2013) 169--178.

[23] H. A. U.-Q. Mohammed, H. R. Wasmi, Active vibration control of cantilever beam by using optimal LQR controller, J. Eng. 24 (11) (2018) 1--17.

[24] A. K. Singh, B. C. Pal, Decentralized control of oscillatory dynamics in power systems using an extended LQR, IEEE Trans. Power Syst. 31 (3) (2015) 1715--1728.

[25] K. B. Slimane, Z. Tmar, M. Besbes, Deep reinforcement learning LQR controller design for MIMO systems applied to gas production facility, Int. J. Autom. Control 19 (6) (2025) 669--704.

[26] U. V. Kalabić, I. V. Kolmanovsky, A constraint-separation principle in model predictive control, Automatica 121 (2020) 109190.

[27] D. He, Dual-mode nonlinear MPC via terminal control laws with free-parameters, IEEE/CAA J. Autom. Sin. 4 (3) (2016) 526--533.

[28] V. Reddy, A. Boker, H. Eldardiry, Learning-based optimal control of linear time-varying systems over large time intervals, Syst. Control Lett. 185 (2024) 105750.

[29] J. Xu, C. Wen, D. Xu, Optimal control data scheduling with limited controller-plant communication, Sci. China Inf. Sci. 61 (1) (2018) 012202.
[30] J. Fong, Y. Tan, V. Crocher, D. Oetomo, I. Mareels, Dual-loop iterative optimal control for the finite horizon LQR problem with unknown dynamics, Syst. Control Lett. 111 (2018) 49--57.
[31] B. Deng, Y. Shin, L. Lu, Z. Zhang, G. E. Karniadakis, Approximation rates of DeepONets for learning operators arising from advection-diffusion equations, Neur. Netw. 153 (2022) 411--426.
[32] H. Abou-Kandil, G. Freiling, V. Ionescu, G. Jank, Matrix Riccati equations in control and systems theory, Birkhäuser, 2012.
[33] K. Zhou, J. C. Doyle, K. Glover, et al., Robust and optimal control, Vol. 40, Prentice hall New Jersey, 1996.
[34] B. D. Anderson, J. B. Moore, Optimal control: linear quadratic methods, Courier Corporation, 2007.
[35] D. Bertsekas, Dynamic programming and optimal control: Volume I, Vol. 4, Athena scientific, 2012.
[36] E. A. Coddington, N. Levinson, T. Teichmann, Theory of ordinary differential equations (1956).

* School of Mathematics and Statistics, Beijing Institute of Technology, 100081 Beijing, China.
† Chair of Computational Mathematics, DeustoTech, University of Deusto, Avenida de las Universidades 24, 48007 Bilbao, Basque Country, Spain.

Email address: chenjun_bcbm@163.com

Email address: umberto.biccari@deusto.es

Email address: jmwang@bit.edu.cn