Online Model-based Reinforcement Learning for Optimal Control with Provable Stability Guarantees

Milad Farsi, Ruikun Zhou, Amartya Mukherjee, Christopher Song, and Jun Liu, *Senior Member, IEEE*

***Abstract*—This paper introduces a Model-based Reinforcement Learning (MBRL) framework for solving optimal control problems. We use a Satisfiability Modulo Theories (SMT) solver and numerical simulations to verify the stability and performance of the resulting approximate optimal controllers. Leveraging a structured continuous-time model described in terms of a basis set, we formulate an infinite-horizon optimal control problem to optimize a given cost functional. The system's structure and a value function parameterized in a quadratic form enable the analytical computation of parameter update rules. Moreover, the quadratic form of the value function offers a compact approach to parameter updates, thereby reducing computational complexity. We establish the asymptotic stability and optimality of the resulting control around equilibrium, drawing parallels with the Linear Quadratic Regulator (LQR). In conjunction with a system identification unit, the framework can be employed as an online learning-based algorithm. Hence, we compare the proposed approach with Proximal Policy Optimization (PPO), Successive Galerkin Approximation (SGA), and Adaptive Dynamic Programming (ADP), three well-known Reinforcement Learning (RL) and optimal control techniques. The results demonstrate the advantages of the proposed technique in terms of computational efficiency and performance.**

***Index Terms*—Model-Based Reinforcement Learning (MBRL), optimal control, Satisfiability Modulo Theories (SMT) solver, Hamilton-Jacobi-Bellman (HJB), stability analysis.**

I. Introduction

Model-based Reinforcement Learning (MBRL) techniques outperform many other machine learning methods in terms of data efficiency, making them suitable for demanding applications such as robotics [9]. These techniques demonstrate optimal behavior even within the first few trials [19], [32]. In fact, learning a deterministic or probabilistic transition model for predictions saves much more effort than treating any point in the state-control space individually. For these reasons, MBRL techniques feature prominently in the continuous-time control literature[30], [1], [26].

Various MBRL techniques have been proposed over a period spanning decades[23]. In value function methods such as those described in[17], [20], which are also known as approximate/Adaptive Dynamic Programming (ADP), a value function is used to construct the policy. Value methods to address optimal control problems typically require solving the associated Hamilton-Jacobi-Bellman (HJB) equation. However, many techniques for solving these equations suffer from the curse of dimensionality. To overcome this difficulty, approximate dynamic programming techniques use a model of the value to find an approximate solution [21]. These techniques use the Bellman error, which is acquired by exploring the state space. The model parameters are iteratively estimated using a gradient-descent or least-squares method. For example, in [16], a parametric model is used to approximate the value function and a least-squares minimization technique is used to adjust the parameters according to the Bellman error.

There exist different approaches in the literature that attempt to solve the optimal control problem such as, Successive Galerkin Approximation (SGA) [3] and ADP [14], [17], [20]. Although they demonstrate efficiency, their applicability relies on the presence of an initial stable controller to run the algorithms. This impedes the implementations in many cases where a stabilizing control is not provided for a region of interest within the state domain.

In [10], a new method for approximating solutions to optimal control problems for general nonlinear systems is developed. Like many existing methods, it is underpinned by Bellman theory [21], [35], [5], [16] , while its formulation differs from prior approaches. Specifically, it presents an analytical solution to the optimal control problem. This method thereby eliminates the need to iteratively estimate the value parameters using techniques such as gradient descent or least-squares. Moreover, because of Additionally, a quadratic parametrization of the value function allows the parameters to be stored in a square matrix, improving the algorithm's computational efficiency.

Although Structured Online Learning (SOL) as presented in [10], performs well in simulations, it lacks stability and optimality guarantees, while they are essential for real-world applications. Achieving such provable guarantees has been the bottleneck of many Reinforcement Learning (RL) techniques. The present paper fills in this lacuna, and in the process of doing so reveals connections with Forward-Propagating Riccati Equation (FPRE) techniques for solving Linear Quadratic Regulator (LQR) problems. In more detail, it can be shown that in the RL setting, the differential Riccati equation can be solved in forward time resulting in a similar solution as LQR. This contrasts with optimal control which is generally a backward method requiring an exact model of the system. Using a similar approach, we show that a stabilizing control in terms of some bases may be obtained by solving a state-dependent Riccati equation in forward time. Additionally, to numerically determine whether the controller thus obtained stabilizes the nonlinear system, we use an Satisfiability Modulo Theories (SMT) solver. SMT solvers are able to provide

Manuscript created October 2023;

Milad Farsi is with the Systems Design Engineering Dep., University of Waterloo, Waterloo, Canada (mfarsi@uwaterloo.ca)
Ruikun Zhou, Amartya Mukherjee, Christopher Song, and Jun Liu are with the Applied Mathematics Dep., University of Waterloo, Waterloo, Canada

stability guarantees for nonlinear systems with valid Lyapunov functions, and the corresponding stabilizing controllers by converting the Lyapunov conditions into first-order logic formulae over the state space [8], [36]. We will use this reliable tool to verify the stability of nonlinear systems with the obtained approximate optimal controllers and certify the Lyapunov function candidates. In summary, the focus of this paper is to introduce an efficient MBRL framework with rigorous guarantees, which entails the following contributions:

- We show that with the proposed algorithm, Global uniform exponential stability (GUES) of the closed-loop linear systems based on the solutions of the FPRE can be obtained.
- We present the stability analysis of nonlinear systems with the learned approximate optimal controllers using SOL, while the stability guarantees for the closed-loop systems are further rigorously certified by an SMT solver.
- We illustrate the effectiveness and advantages of SOL with detailed comparisons for a set of classic nonlinear systems with three existing optimal control and RL techniques.

The rest of this paper is organized as follows. In Section II, we give the problem formulation and summarize the approximate optimal control approach from [10]. Section III presents the stability analysis and numerical verification. Section IV develops the SOL learning algorithm. In Section V, we present numerical examples, make comparisons with existing methods, and draw some qualitative conclusions about their relative performance.

Notations. We denote n-dimensional Euclidean space by $\mathbb{R}^n$ and a ball of radius r centered at the origin by $\bar{D}_r$. We further denote by $|k|$ the absolute value of a number k and by $||x||$ the 2-norm of a vector $x \in \mathbb{R}^n$. By $\frac{\partial V}{\partial x}$ we mean the gradient of $V : \mathbb{R}^n \to \mathbb{R}$, and by $\frac{\partial \Phi}{\partial x}$ we mean the Jacobian matrix of $\Phi : \mathbb{R}^n \to \mathbb{R}^p$. A diagonal square matrix A with elements $a_1, \ldots, a_n$ on the diagonal is shortened as $A = \text{diag}([a_1, \ldots, a_n])$. $U(a, b)$ represents a uniform distribution with the bounds a and b.

II. Approximate Optimal Control

In this section, we formulate the problem and summarize the approach presented in [10].

A. Problem Formulation

We consider the nonlinear affine system

$$\dot{x} = f(x) + g(x)u, \tag{1}$$

where $x \in D \subset \mathbb{R}^n$ and $u \in \Omega \subset \mathbb{R}^m$ are respectively the state and the control input. We have $f : D \to \mathbb{R}^n$ and $g : D \to \mathbb{R}^{n\times m}$ and we assume that the functions f_i and g_i can be approximated on the domain of interest by a linear combination of some basis functions $\phi_j \in C^1 : D \to \mathbb{R}$ for $j = 1, 2, \ldots, p$ and $i = 1, 2, \ldots, n$. Substituting these approximations, (1) becomes

$$\dot{x} = W\Phi + \sum_{j=1}^{m} W_j \Phi u_j \tag{2}$$

where W and $W_j \in \mathbb{R}^{n\times p}$ are matrices of coefficients, with subscript W_j corresponding to the jth control input. The bases are chosen so that constant and linear elements comprise the first two rows, for example, $\Phi = \begin{bmatrix} 1 & x_1 \ldots & x_n & \phi_{n+2}(x) \ldots & \phi_p(x) \end{bmatrix}^T$.

We wish to minimize the cost function

$$J(t, x_0, u) = \lim_{T\to\infty} \int_0^T \mathrm{e}^{-\gamma t} \left(x^T Q x + u^T R u \right) \mathrm{d}t, \tag{3}$$

along the solution through $x_0 = x(0)$ where $Q \in \mathbb{R}^{n\times n}$ positive definite, $\gamma \geq 0$ the discount factor, and $R \in \mathbb{R}^{m\times m}$ a diagonal matrix with only positive values are chosen by design criteria. When $\gamma > 0$, this defines a discounted optimal control problem. Discounted optimal control problems are widely used in reinforcement learning to determine the time horizon considered for minimizing the objective [20]; see for example the discussion given in [24], [11].

For feedback control $u = \omega(x(t)), t \in [0, \infty)$, the optimal control is given by $\omega^* = \arg\min_{u(\cdot)\in\bar{\Gamma}(x_0)} J(t, x_0, u(\cdot))$ where $\bar{\Gamma}$ is the set of admissible controls.

Without loss of generality, we express the cost (3) in terms of Φ as

$$J(t, x_0, u) = \lim_{T\to\infty} \int_0^T \mathrm{e}^{-\gamma t} \left(\Phi^T \bar{Q} \Phi + u^T R u \right) \mathrm{d}t \tag{4}$$

where $\bar{Q} = \text{diag}\left(0, Q, \mathbf{0}_{(p-n-1)\times(p-n-1)}\right)$ is a block diagonal matrix.

The optimal cost to go from (t, x) is defined to be $\bar{V}(t, x) := \min_{u[t,T]} J(t, x, u)$, where $u[t, T]$ is the policy on time interval $[t, T]$. Then, by the Hamiltonian

$$H = \nu(t, x, u) + \frac{\partial}{\partial x}\bar{V}(t, x)^T (W\Phi + \sum_{j=1}^{m} W_j \Phi u_j),$$

the corresponding HJB equation can be written as

$$-\frac{\partial}{\partial t}\bar{V}(t, x) = \min_{u(\cdot)\in\Gamma(x_0)} H \tag{5}$$

where $\nu(t, x, u)$ is the running cost in (4). Section 5.1.3 of [22] contains more detail in this regard. Analytical solutions to this equation are not known, however, [4], [20], [25] have shown that approximate solutions can be computed by numerical techniques.

B. Approximate Solution

We follow [10] in presenting an analytical method. For the discounted problem, we assume that the discounted optimal value takes the form $\bar{V}(t, x) = \mathrm{e}^{-\gamma t} V$ where V is the undiscounted optimal value. Now, we use the following approximation

$$V \approx \Phi^T P \Phi \tag{6}$$

where P is a symmetric matrix of parameters that captures the evolution of the learned value parameters over time.

Remark 1: This approximation is justified since V is a positive function. We can write V as an inner product $V = \langle v, v\rangle$ for some $v : D \to \mathbb{R}^{p_1}$, where p_1 is a positive integer, with a trivial choice given by $v = V^{1/2}$. We now consider an approximation $v \approx k_1\Phi$, where $k_1 \in \mathbb{R}^{p_1 \times p}$. This yields $V = \langle v, v\rangle \approx \langle k_1\Phi, k_1\Phi\rangle = \Phi^T k_1^T k_1 \Phi = \Phi^T P \Phi$, where P is symmetric and positive define.

With this approximation, the Hamiltonian H becomes

$$\begin{aligned} H =& \mathrm{e}^{-\gamma t}(\Phi^T \bar{Q}\Phi + u^T R u) \\ &+ \mathrm{e}^{-\gamma t}\Phi^T P \frac{\partial \Phi}{\partial x}\left(W\Phi + \sum_{j=1}^{m} W_j \Phi u_j\right) \\ &+ \mathrm{e}^{-\gamma t}\left(\Phi^T W^T + \sum_{j=1}^{m} u_j^T \Phi^T W_j^T\right)\frac{\partial \Phi}{\partial x}^T P\Phi. \end{aligned}$$

Based on the structure of R, the quadratic term of u is rewritten in terms of its components, where $r_j \neq 0$ is the jth component on the diagonal of R. The minimum of the Hamiltonian occurs at

$$\frac{\partial H}{\partial u_j} = 2 r_j u_j + 2\Phi^T P \frac{\partial \Phi}{\partial x} W_j \Phi = 0, \qquad j = 1, 2, \ldots, m, \tag{7}$$

so the jth optimal control is

$$u_j^* = -\Phi^T r_j^{-1} P \frac{\partial \Phi}{\partial x} W_j \Phi. \tag{8}$$

By substituting the above and the value function in (5), we obtain

$$\begin{aligned} &-\Phi^T \dot{P}\Phi + \gamma \Phi^T P \Phi = \Phi^T \bar{Q}\Phi + \\ &\qquad - \Phi^T P \frac{\partial \Phi}{\partial x}\left(\sum_{j=1}^{m} W_j \Phi r_j^{-1} \Phi^T W_j^T\right)\frac{\partial \Phi}{\partial x}^T P\Phi \\ &\qquad + \Phi^T P \frac{\partial \Phi}{\partial x} W \Phi + \Phi^T W^T \frac{\partial \Phi}{\partial x}^T P \Phi, \end{aligned} \tag{9}$$

where

$$\begin{aligned} -\dot{P} =& \bar{Q} + P\frac{\partial \Phi}{\partial x} W + W^T \frac{\partial \Phi}{\partial x}^T P - \gamma P \\ &- P \frac{\partial \Phi}{\partial x}\left(\sum_{j=1}^{m} W_j \Phi r_j^{-1} \Phi^T W_j^T\right)\frac{\partial \Phi}{\partial x}^T P. \end{aligned} \tag{10}$$

is a sufficient condition for relation (9) to hold.

The differential equation (10) is propagated in the forward time similar to [27] by changing $\mathrm{d}t := -\mathrm{d}t$. Accordingly, on the left-hand side, this will result in a sign flip as $-\dot{P} \rightarrow \dot{P}$. This allows the estimate of the system to be updated on the fly in an MBRL framework. Moreover, because of the general case considered in obtaining (10) where Φ includes arbitrary nonlinear basis functions of the state, state dependency seems inevitable unlike standard Differential Riccati Equation (DRE) for linear systems. Hence, (10) is solved along solution trajectories.

Remark 2: This section presented the control technique for given W and W_j. However, in the implementations, an estimation of these weights will be used, which is discussed later including the learning algorithm. Further discussions are given in the appendix section IX.

III. Stability and Optimality Analysis

In this section, we analyze the stability of the proposed method and make connections with FPRE for linear systems [33], [27]. To do this, we formulate the LQR problem for the linearization of (2). We then show that when close to the origin, the integration of (10) becomes similar to the forward propagated solution of the linearized system.

We begin with some conditions on (2). We assume the origin is an equilibrium point. Accordingly, we construct the bases $\Phi = [1 \quad x^T \quad \Gamma(x)^T]^T$, where Γ includes the rest of the bases which are nonlinear. Then the system (2) may be written as

$$\dot{x} = \begin{bmatrix}0 & W_2 & W_3\end{bmatrix}\begin{bmatrix}1 \\ x \\ \Gamma(x)\end{bmatrix} + \begin{bmatrix}W_{j_1} & W_{j_2} & W_{j_3}\end{bmatrix}\begin{bmatrix}1 \\ x \\ \Gamma(x)\end{bmatrix} u. \tag{11}$$

A. Linear case

Defining $\Gamma_1 = \frac{\partial \Gamma(x)}{\partial x}|_{x=0}$, the linearization of (11) at an equilibrium point is

$$\dot{x} = Ax + \sum_{j=1}^{m} B_j u_j \tag{12}$$

where $A = W_2 + W_3\Gamma_1$, and $B_j = W_{j_1}$.

Now consider the LQR problem with quadratic cost (3) and $\gamma = 0$ for the linearized system (12). The optimal control is $u = -r_j^{-1} B^T \bar{S} x$, where $\bar{S}$ is the solution of the algebraic Riccati equation

$$Q + \bar{S}A + A^T \bar{S} - \bar{S}(\sum_{j=1}^{m} B_j r_j^{-1} B_j^T)\bar{S} = 0. \tag{13}$$

Alternatively, we may consider the forward solution of the differential Riccati equation

$$\begin{aligned} & u_j = -r_j^{-1} B_j^T S x, \\ & \dot{S} = Q + SA + A^T S - S(\sum_{j=1}^{m} B_j r_j^{-1} B_j^T)^T S. \end{aligned} \tag{14}$$

where we update the feedback controller with the solution $S(t)$ at any $t \in [0, \infty)$. Substituting A and B_j, we arrive at

$$u_j = -r_j^{-1} W_{j1}{}^T S x, \tag{15}$$

$$\begin{aligned} \dot{S} =& Q + S(W_2 + W_3\Gamma_1) + (W_2^T + \Gamma_1^T W_3^T)S \\ &- S(\sum_{j=1}^{m} W_{j1} r_j^{-1} W_{j1}{}^T)S. \end{aligned} \tag{16}$$

In the following lemma, we establish the stability of the method. It strengthens the asymptotic convergence of Theorems 1 and 4 in [27] to global uniform exponential stability under slightly weaker assumptions.

Lemma 1: Assume that (A, B) is stabilizable and Q is positive definite. Consider system (12) with control (14), where $S(t)$ is the positive semi-definite solution of the FPRE (14) with $S(0) \geq 0$ for all $t \in [0, \infty)$. The origin of the closed-loop system is Globally Uniformly Exponentially Stable (GUES).

Proof. 1: See section VII in the appendix.

B. General case

For a known system in the form of (11), the optimal control is given by (8) and the value is updated by the evolution of the parameters in (10). The following theorem establishes the stability of the closed-loop system.

Theorem 3.1: Assume that the system (2) with (3) is given and is locally controllable and observable. Suppose that x remains close to the origin and solutions of (10) remain bounded. Then, there exists some $r_1 > 0$ such that the solution $P(t)$ of (10) starting from $P(0) = 0$ establishes an asymptotically stable controller for all $x_0 \in \bar{D}_{r_1}$. Moreover, as $x \to 0$, also by choosing $\gamma \to 0$, the feedback control rule converges to the LQR control of the linearized system given by (13).

Proof. 2: See section VII in the appendix.

C. Stability guarantees with the SMT solver

The stability of equilibrium points of the nonlinear system (1) can be guaranteed by the Lyapunov stability theorem, which is given below.

Theorem 3.2 (Lyapunov Stability Theorem): Consider the nonlinear control-affine system (1) and assume the origin is an equilibrium point. Let $V(x)$ be a continuously differentiable and positive definite function with a negative definite Lie derivative defined on D satisfying

$$V(0) = 0 \text{ and } V(x) > 0 \text{ for all } x \in D \text{ with } x \neq 0, \quad (17)$$

$$\begin{aligned}\dot{V}(x) =& \frac{\partial V}{\partial x}\left(f(x) + g(x)u\right) < 0 \text{ for all } x \neq 0 \\ & \text{and for all admissible control inputs.} \quad (18)\end{aligned}$$

Then the origin is asymptotically stable.

We know that the level sets of the Lyapunov function V provide an estimate of the Region of Attraction (ROA) of the equilibrium point.

Lemma 2 (ROA estimate with Lyapunov Functions): Suppose that V satisfies the conditions of Theorem 3.2. Defining

$$V^c := \{x \in \mathcal{D} \mid V(x) \leq c\},$$

we have V^c a ROA for (1) for every $c > 0$.

Here, the value function in (6) serves as a Lyapunov function candidate with the approximate optimal control (8). In order to use SMT solvers to check the validity of the candidates, we follow [8] in writing the Lyapunov conditions in Theorem 3.2 as falsification constraints in the form of first-order logic formula over reals as follows. We write

$$\Phi_\varepsilon(x) := \left(\sum_{i=1}^{n} x_i^2 \geq \varepsilon\right) \wedge \left(V(x) \leq 0 \vee \dot{V}(x) \geq 0\right), \quad (19)$$

where ε is a numerical error parameter introduced to control numerical sensitivity around the origin. SMT solvers typically return `UNSAT` when there are no violations of the negation of the Lyapunov conditions on the state space, or counterexamples that satisfy the constraints. If it returns `UNSAT` for a given candidate, then the candidate is certified as a Lyapunov function which may be used to estimate the ROA.

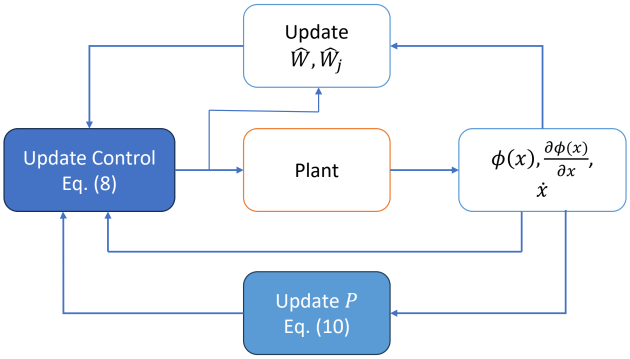


Fig. 1: A schematic view of the learning control framework.

Next, we present an online learning algorithm based on the proposed optimal control framework.

IV. MBRL Algorithm

By considering the nonlinear input affine system in terms of bases as in (2), the state-dependent matrix differential equation (10) is exploited to establish a nonlinear feedback control. We develop this line of thinking into an MBRL algorithm, whose specification comprises this section.

In the appendix IX, we review the components involved in the implementation of the algorithm including the control and the model update units, shown in Fig. 1. Accordingly, we present the learning algorithm as in Algorithm 1.

Algorithm 1

```
1: $P(0) \leftarrow \mathbf{0}_{p \times p}$
2: $\hat{W}$ and $\hat{W}_j \leftarrow$ Small random values
3: for $k = 0, 1, \ldots$ do
4:     acquire samples $(\dot{x}, x, u)$ at $t_k$
5:     evaluate $\Phi$ and $\frac{\partial \Phi}{\partial x}$
6:     update $\hat{W}$ and $\hat{W}_j$ using (39)
7:     update $P$ by integrating (10) for some time length $T_0$
8:     update control $u$ using (8)
9: end for
```

Algorithm 1 may be understood as a continuous-time variant of the generalized policy algorithm [21] by integrating (10) at each step k for some time length T_0. Accordingly, Step 7 can be interpreted as the *Value Update* step as it directly results in the updated value. Then, the *Policy Improvement* step is given by Step 8. One can choose T_0 considerably large so that the value converges at each iteration. The case $t \to \infty$ results in the Policy Iteration (PI) algorithm.

Remark 3: Considering that an identified model is employed, asymptotic convergence to the origin may not be possible due to the uncertainty in the model. However, this can be circumvented by employing robust techniques such as [34], [28]. In the appendix section VIII, we integrate a robust component with the presented learning approach that guarantees asymptotic convergence.

V. Experiments

To illustrate the effectiveness of the algorithm, we apply the Algorithm 1 to a linear system and several benchmark examples including the pendulum, cartpole, and quadrotor systems. The dynamics and settings of simulations can be found in the supplementary material. Furthermore, we give some qualitative comparisons with a set of classic RL and optimal control approaches. In addition, we validate the stability and estimate the ROAs of the nonlinear systems with the obtained controllers with the SMT solver, dReal [12], which handles nonlinear real arithmetic to verify the validity of the Lyapunov functions.

A. Comparison and consistency with LQR

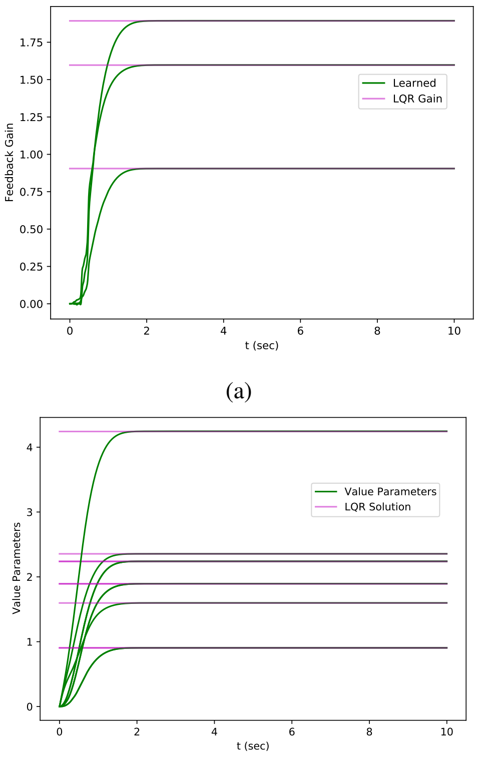
As mentioned in Section III, SOL yields the same results LQR for linear systems and on a small region around the equilibrium point for nonlinear systems where the linearization is valid. In this example, we investigate the convergence of the SOL algorithm to the optimal control for an unknown linear system to show consistency. The optimal control is given by the solution of the associated Algebraic Riccati Equation (ARE). We compare this with the value parameters P obtained by SOL, where the chosen basis includes only constant and linear terms, i.e. $\Phi = [1 \quad x^T]$. Figure 2b shows that solutions of (10) converge to the parameters given by LQR. Likewise, Figure 2a shows the constant linear feedback gain obtained by SOL converging to the optimal LQR gain.

B. Existing RL and optimal control methods for comparison

Before presenting the main results and comparisons, we give brief summaries of the Proximal Policy Optimization (PPO), SGA, and ADP methods used for subsequent comparisons.

PPO is a model-free actor-critic algorithm introduced by [29]. It uses an actor-network $\pi_\theta(a|s)$ that takes the state s as an input and outputs a distribution over actions a, and a value network $V_\phi(s)$ that takes the state s as an input and outputs the expected return. The policy is trained at every episode by solving $\text{argmax}_\theta E[\min(r_\theta(a|s)A(s,a), clip(r_\theta(a|s), 1-\epsilon, 1+\epsilon)A(s,a))]$, where $r_\theta(a|s) = \frac{\pi_\theta(a|s)}{\pi_{\theta_{old}}(a|s)}$ is the ratio term, and $A(s,a)$ is the advantage term. It effectively constrains the ratio $r_\theta(a|s)$ to ensure steady updates. We use the implementation by [2] and train the PPO algorithm on the three environments using Kaggle CPUs. To ensure the reproducibility of our results, we record our hyperparameters in Appendix X-B. The actor network is a neural network with one hidden layer of size 64 to aid in verification. PPO is trained on each environment for 5 seeds for one million time steps each. The additive inverse of the LQR cost function is used as the reward function.

SGA is a method for solving optimal control problems based on policy iteration introduced in [3]. Like other policy iteration algorithms, SGA generates sequences of controllers u_i and Lyapunov functions V_i by computing the latter as the solution to a certain PDE (called the Generalized HJB or GHJB in [3]) and then using it to update the former. This process is detailed on page 27 of [3]. SGA approximates the solution of the GHJB on a certain domain using a Galerkin finite element method and associated weak formulation to obtain the optimal controllers and corresponding Lyapunov functions at the same time.



(a)

(b)

Fig. 2: (a). For the linear system, we illustrate that the feedback gain given by the proposed algorithm asymptotically converges to the optimal gain given by LQR shown in Figure 2b.
(b). The components of P converge to the LQR solution. Accordingly, it is evident that by running SOL algorithm on the linear system the parameters of the value converge to the LQR solution.

The last one is the off-policy semi-global variant of ADP described in Chapter 3 of [14]. Given some initial control policy u_0 that keeps solutions bounded, this method observes solution trajectories of the system under the input $u = u_0 + e$ where e is some noise included for the purpose of exploration. Once enough data are collected, the control is iteratively improved using these data in place of the system dynamics $f(x)$ and $g(x)$, so that the method is model-free. When certain conditions are satisfied, this process generates a sequence u_i of controls (and their corresponding value functions V_i) that converges to the optimal control. A summary of this algorithm is given on page 42 of [14].

C. Inverted Pendulum

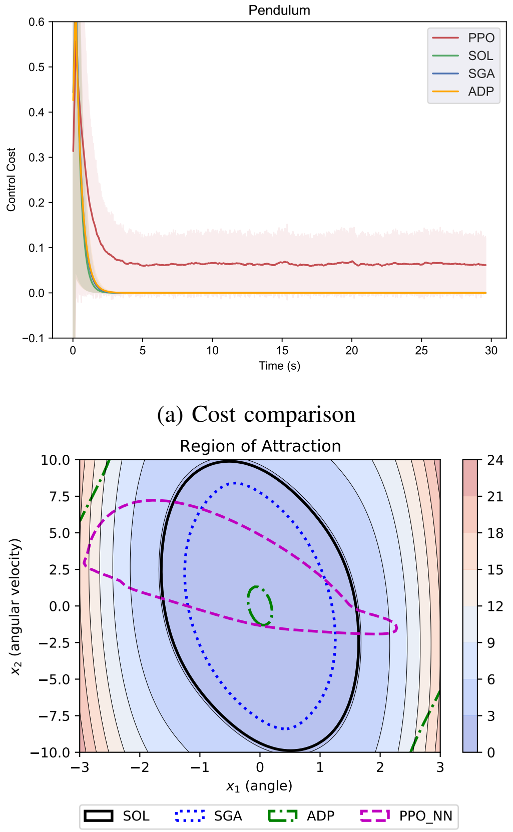
Firstly, we run our algorithm and the aforementioned three algorithms on a well-known nonlinear system, the inverted pendulum. For SGA, the V_i are approximated using a basis consisting of polynomials of even degree. Initial control $u_0 = (x_1 - 19.6\sin(x_i))/40$; is selected. The domain of approximation is $\{(x_1, x_2) : |x_1| \leq 3, |x_2| \leq 10\}$ and the order of the basis is eight. For ADP, we implement this method for the pendulum example. Polynomials of degrees two and four are used as basis functions for V_i, while their derivatives comprise the basis functions for u_i. The initial control must be in the span of the basis, so $u_0 = -x_1$ is selected. The exploration noise e consists of six superimposed sinusoids of different frequencies. The comparison of the key parameters of the four methods under discussion can be found in Table I. The runtime is the average time taken for the model to perform a single evaluation and update.

For runtime, SOL is the fastest by far, and SGA is extremely slow. For training time, this table shows that SGA takes much longer than PPO, which in turn takes longer than SOL and ADP, which are very comparable. SOL uses weighted basis functions to return a controller, which requires significantly fewer parameters compared to actor and critic networks in PPO. In contrast, SGA requires symbolic computation, which is cumbersome when the dimension of the system or basis is high. In fact, the present examples are not far from the limit of practical feasibility for this algorithm, as it is presented in [3]. ADP makes use of off-policy learning, where all observation of system trajectories is completed first, and the same observations are used to compute all subsequent values and control iterates. This reduces the cost associated with simulation. To further illustrate it, Fig. 3a shows that the controllers obtained via SGA, ADP, and SOL have very similar performance on average over 100 trajectories, while PPO has a mean control cost of approximately 0.06 due to its model-free nature, which leads to the lack of stability guarantees in LQR optimal control tasks. Although the former is model-free and the latter is model-based, both ADP and SGA require an initial stabilizing control to be known, and the selection of this control and other things such as the exploration noise can have a significant impact on the convergence of the algorithm. In contrast, SOL requires far fewer specifications to be made in advance.

Moreover, Fig. 3b shows the comparison among the ROA estimates obtained with the four different methods. Here, we use the SMT solver to verify the obtained Lyapunov functions on the aforementioned domain with $\delta = 0.01$. For PPO, we use the algorithm proposed in [8] to learn a neural Lyapunov function, since it only returns an optimal controller as a model-free RL algorithm. As illustrated, SOL returns the largest ROA estimate on the domain, while ADP can only return a valid Lyapunov function on a small region, which leads to a relatively small ROA estimate. Furthermore, PPO does not take the dynamics into consideration when solving the optimal control problem, leading to a peculiar shape. In summary, SOL clearly outperforms PPO and produces a controller with very similar performance to SGA and ADP but with shorter training and runtimes, and requiring less a priori inputs, and SOL captures the largest ROA estimate.

D. Cartpole and Quadrotor

In this subsection, we consider the other two typical nonlinear control problems, Cartpole and Quadrotor for comparison with PPO. The detailed comparison can be found in Table II.



(a) Cost comparison

(b) ROA estimates comparison

Fig. 3: (a) Control cost curves for each algorithm on the inverted pendulum environment averaged over 100 trajectories. The shaded area indicates one standard deviation. Initial conditions are sampled from $U(-1, 1)$. (b) Comparison of ROA estimates of each algorithm for the inverted pendulum.

Environment	Method	Avg cost	Runtime (s)	Training time
Pendulum	SOL	**0.0158**	**0.003**	**2.4 sec**
	PPO	0.0811	0.618	24 min
	SGA	0.0163	21.966	220 sec
	ADP	0.0165	0.127	2.6 sec

TABLE I: Comparison of performance of the four methods for the inverted pendulum

In a similar manner, these two examples also show that SOL outperforms PPO in training time, runtime, and control costs. The detailed visualized comparison can be found in the supplementary material, as it has been omitted from the main text due to page limitations.

Environment	Method	Avg cost	Runtime (s)	Training time
Cartpole	SOL	**0.0815**	**0.002**	**4.7 sec**
	PPO	1094.056	0.597	28 min
Quadrotor	SOL	**33.986**	**0.010**	**61.4 sec**
	PPO	1207.293	0.951	1 hr 20 min

TABLE II: Comparison of performance of SOL and PPO for Cartpole and Quadrotor

VI. Conclusion

This paper proposes an optimal control framework in conjunction with a system identification technique, which can be used as an online learning-based algorithm for nonlinear control-affine systems. We also demonstrate the asymptotic stability and optimality of the resulting control around the equilibrium by investigating its connections with the analogous LQR, while the stability guarantees for the nonlinear system are further certified by an SMT solver. Finally, comparisons are made among the proposed approach and other well-known RL and optimal control techniques, including LQR, PPO, SGA, and ADP. The results demonstrate the advantages of the proposed technique in terms of computational efficiency, performance, and the fact that no initially stable controllers are needed.

References

[1] Christopher G Atkeson and Juan Carlos Santamaria. A comparison of direct and model-based reinforcement learning. In *Proceedings of International Conference on Robotics and Automation*, volume 4, pages 3557–3564. IEEE, 1997.

[2] Nikhil Barhate. Minimal pytorch implementation of proximal policy optimization. https://github.com/nikhilbarhate99/PPO-PyTorch, 2021.

[3] Randal W Beard, George N Saridis, and John T Wen. Galerkin approximations of the generalized hamilton-jacobi-bellman equation. *Automatica*, 33(12):2159–2177, 1997.

[4] Dimitri P Bertsekas. *Approximate dynamic programming*. Citeseer, 2008.

[5] Shubhendu Bhasin, Rushikesh Kamalapurkar, Marcus Johnson, Kyriakos G Vamvoudakis, Frank L Lewis, and Warren E Dixon. A novel actor–critic–identifier architecture for approximate optimal control of uncertain nonlinear systems. *Automatica*, 49(1):82–92, 2013.

[6] Steven L Brunton, Joshua L Proctor, and J Nathan Kutz. Discovering governing equations from data by sparse identification of nonlinear dynamical systems. *Proceedings of the National Academy of Sciences*, 113(15):3932–3937, 2016.

[7] Frank M Callier, Joseph Winkin, and Jacques L Willems. Convergence of the time-invariant Riccati differential equation and LQ-problem: mechanisms of attraction. *International Journal of Control*, 59(4):983–1000, 1994.

[8] Ya-Chien Chang, Nima Roohi, and Sicun Gao. Neural Lyapunov Control. *Advances in neural information processing systems*, 32, 2019.

[9] Yan Duan, Xi Chen, Rein Houthooft, John Schulman, and Pieter Abbeel. Benchmarking deep reinforcement learning for continuous control. In *International conference on machine learning*, pages 1329–1338. PMLR, 2016.

[10] Milad Farsi and Jun Liu. Structured online learning-based control of continuous-time nonlinear systems. *IFAC-PapersOnLine*, 53(2):8142–8149, 2020.

[11] Vladimir Gaitsgory, Lars Grüne, and Neil Thatcher. Stabilization with discounted optimal control. *Systems & Control Letters*, 82:91–98, 2015.

[12] Sicun Gao, Soonho Kong, and Edmund M Clarke. dreal: An SMT solver for nonlinear theories over the reals. In *Automated Deduction–CADE-24: 24th International Conference on Automated Deduction, Lake Placid, NY, USA, June 9-14, 2013. Proceedings 24*, pages 208–214. Springer, 2013.

[13] Joao P Hespanha. *Linear Systems Theory*. Princeton University Press, 2018.

[14] Yu Jiang and Zhong-Ping Jiang. *Robust Adaptive Dynamic Programming*. John Wiley & Sons, 2017.

[15] Eurika Kaiser, J Nathan Kutz, and Steven L Brunton. Sparse identification of nonlinear dynamics for model predictive control in the low-data limit. *Proceedings of the Royal Society A*, 474(2219):20180335, 2018.

[16] Rushikesh Kamalapurkar, Patrick Walters, and Warren E Dixon. Model-based reinforcement learning for approximate optimal regulation. *Automatica (Journal of IFAC)*, 64(C):94–104, 2016.

[17] Rushikesh Kamalapurkar, Patrick Walters, Joel Rosenfeld, and Warren Dixon. Model-based reinforcement learning for approximate optimal control. In *Reinforcement Learning for Optimal Feedback Control*, pages 99–148. Springer, 2018.

[18] Jyrki Kivinen, Alexander J Smola, and Robert C Williamson. Online learning with kernels. *IEEE Transactions on Signal Processing*, 52(8):2165–2176, 2004.

[19] Eric Langlois, Shunshi Zhang, Guodong Zhang, Pieter Abbeel, and Jimmy Ba. Benchmarking model-based reinforcement learning. *arXiv preprint arXiv:1907.02057*, 2019.

[20] Frank L Lewis and Derong Liu. *Reinforcement learning and approximate dynamic programming for feedback control*, volume 17. John Wiley & Sons, 2013.

[21] Frank L Lewis and Draguna Vrabie. Reinforcement learning and adaptive dynamic programming for feedback control. *IEEE Circuits and Systems Magazine*, 9(3):32–50, 2009.

[22] Daniel Liberzon. *Calculus of variations and optimal control theory: a concise introduction*. Princeton university press, 2011.

[23] Athanasios S Polydoros and Lazaros Nalpantidis. Survey of model-based reinforcement learning: Applications on robotics. *Journal of Intelligent & Robotic Systems*, 86(2):153–173, 2017.

[24] Romain Postoyan, L Buşoniu, D Nešić, and Jamal Daafouz. Stability of infinite-horizon optimal control with discounted cost. In *53rd IEEE Conference on Decision and Control*, pages 3903–3908. IEEE, 2014.

[25] Warren B Powell. Perspectives of approximate dynamic programming. *Annals of Operations Research*, 241(1):319–356, 2016.

[26] Warren Buckler Powell. *Handbook of learning and approximate dynamic programming*, volume 2. John Wiley & Sons, 2004.

[27] Anna Prach, Ozan Tekinalp, and Dennis S Bernstein. Infinite-horizon linear-quadratic control by forward propagation of the differential riccati equation [lecture notes]. *IEEE Control Systems Magazine*, 35(2):78–93, 2015.

[28] Zhihua Qu and Jian-Xin Xu. Model-based learning controls and their comparisons using lyapunov direct method. *Asian Journal of Control*, 4(1):99–110, 2002.

[29] John Schulman, Filip Wolski, Prafulla Dhariwal, Alec Radford, and Oleg Klimov. Proximal policy optimization algorithms, 2017.

[30] Richard S Sutton. Integrated architectures for learning, planning, and reacting based on approximating dynamic programming. In *Machine Learning Proceedings 1990*, pages 216–224. Elsevier, 1990.

[31] Steven Van Vaerenbergh and Ignacio Santamaría. Online regression with kernels. In *Regularization, Optimization, Kernels, and Support Vector Machines*, pages 495–521. Chapman and Hall/CRC, 2014.

[32] Tingwu Wang, Xuchan Bao, Ignasi Clavera, Jerrick Hoang, Yeming Wen, Eric Langlois, Shunshi Zhang, Guodong Zhang, Pieter Abbeel, and Jimmy Ba. Benchmarking model-based reinforcement learning. *arXiv preprint arXiv:1907.02057*, 2019.

[33] Avishai Weiss, Ilya Kolmanovsky, and Dennis S Bernstein. Forward-integration riccati-based output-feedback control of linear time-varying systems. In *2012 American Control Conference (ACC)*, pages 6708–6714. IEEE, 2012.

[34] Bin Xian, Darren M Dawson, Marcio S de Queiroz, and Jian Chen. A continuous asymptotic tracking control strategy for uncertain nonlinear systems. *IEEE Transactions on Automatic Control*, 49(7):1206–1211, 2004.

[35] Huaguang Zhang, Lili Cui, Xin Zhang, and Yanhong Luo. Data-driven robust approximate optimal tracking control for unknown general nonlinear systems using adaptive dynamic programming method. *IEEE Transactions on Neural Networks*, 22(12):2226–2236, 2011.

[36] Ruikun Zhou, Thanin Quartz, Hans De Sterck, and Jun Liu. Neural Lyapunov control of unknown nonlinear systems with stability guarantees. In *Advances in Neural Information Processing Systems*, 2022.

VII. Appendix: Proofs and Lemmas

Proof. 3: (Lemma 1) Note that $Q > 0$ implies (A, Q) is observable (and hence detectable). According to [7], [27], $S(t)$ converges to the unique stabilizing solution $\bar{S}$ of the ARE. It can be proved by the Lyapunov test for observability (see, e.g., [13, Theorem 15.10]) that $\bar{S} > 0$. Consider the Lyapunov function

$$V(x) = x^T \bar{S} x.$$

Let $G = BR^{-1}B^T$. Clearly, G is a constant symmetric matrix and $G \geq 0$. Furthermore, $S(t)$ is also symmetric. We have

$$\dot{V}(x) = x^T \bar{S}(A - GS(t))x + x^T \bar{S}(A^T - S(t)G)\bar{S}x$$

$$
\begin{aligned}
&= x^T\left[\bar{S}A + A^T\bar{S} - \bar{S}GS(t) - S(t)G\bar{S}\right]x \\
&= x^T[\bar{S}A + A^T\bar{S} + Q - \bar{S}G\bar{S} - Q + \bar{S}G\bar{S} - \bar{S}GS(t) \\
&\quad - S(t)G\bar{S}]x \\
&= x^T\left[-Q - \bar{S}G\bar{S} + 2\bar{S}G\bar{S} - \bar{S}GS(t) - S(t)G\bar{S}\right]x \\
&\leq -x^T(Q - E(t))x,
\end{aligned}
$$

where $E(t) = 2\bar{S}G\bar{S} - \bar{S}GS(t) - S(t)G\bar{S}$ and we used the ARE (13) satisfied by $\bar{S}$ to obtain the last equality and $\bar{S}G\bar{S} \geq 0$ to obtain the inequality. Since $S(t) \to \bar{S}$ as $t \to \infty$ [27], we have $E(t) \to 0$ as $t \to \infty$. This can be seen by taking the limit in $E(t) = \bar{S}G(\bar{S} - S(t)) + (\bar{S} - S(t))G\bar{S}$ as $t \to 0$. There exists some $T > 0$ and a positive constant μ such that $Q(t) - E(t) - \mu I \geq 0$. It follows that

$$
\dot{V}(x) \leq -\mu\|x\|^2, \quad \forall t \geq T, \, \forall x \in \mathrm{I\!R}^n. \tag{20}
$$

Now, we can show that there exists some $k > 0$ and $c > 0$ such that

$$
\|x(t)\| \leq k\|x(T)\|e^{-c(t-T)}, \quad t \geq T. \tag{21}
$$

By continuity and the fact that $S(t) \to \bar{S}$ as $t \to \infty$, $S(t)$ is bounded on $[0, \infty)$. It follows from the closed-loop system that there exist constants $C > 0$ and $M > 0$ such that

$$
\|x(t)\| \leq M\|x_0\|e^{Ct}, \quad t \geq 0, \tag{22}
$$

where $x_0 = x(0)$. Indeed, a simple comparison argument suffices to show this. We have

$$
\frac{d}{dt}[\|x(t)\|^2] = 2x^T(t)(A - BR^{-1}B^TS(t))x(t) \leq 2C\|x(t)\|^2,
$$

where such a constant $C > 0$ exists because $S(t)$ is bounded on $[0, \infty)$, which implies (22) with $M = 1$. On the interval $[0, T]$, we can rewrite (22) as

$$
\begin{aligned}
\|x(t)\| &\leq Me^{(C+c)t}\|x_0\|e^{-ct} \\
&\leq Me^{(C+c)T}\|x_0\|e^{-ct}, \qquad \forall t \in [0, t].
\end{aligned} \tag{23}
$$

For $t \geq T$, rewrite (21) as

$$
\begin{aligned}
\|x(t)\| &\leq ke^{cT}\|x(T)\|e^{-ct} \\
&\leq kMe^{(C+c)T}\|x_0\|e^{-ct}, \quad \forall t \geq T,
\end{aligned} \tag{24}
$$

where we used $x(T) \leq M\|x_0\|e^{CT}$ from (22). Combining (23) and (24) gives

$$
\|x(t)\| \leq K\|x_0\|e^{-ct}, \quad \forall t \geq 0, \tag{25}
$$

where $K = kMe^{(C+c)T}$ (note that $k \geq 1$ for (21) to hold). This verifies that the origin is GUES for the closed-loop system.

Proof. 4: (Theorem 3.1) Consider a ball of radius r around the origin containing the solutions. We will show that the dominating part of the control obtained by solving (10) is equivalent to the LQR control given by (16). For this purpose, we first consider the Taylor expansion of bases in (11) and its partial derivatives as below,

$$
\Phi = \begin{bmatrix} 1 \\ x \\ \Gamma_1 x + O(x^2) \end{bmatrix}, \text{and} \quad \frac{\partial\Phi}{\partial x} = \begin{bmatrix} 0 \\ \mathbf{I} \\ \Gamma_1 + O(x) \end{bmatrix}, \tag{26}
$$

where $O(x^\alpha)$ denotes higher order terms $x_1^{\alpha_1}x_2^{\alpha_2}\ldots x_n^{\alpha_n}$ in appropriate dimensions with $\alpha = \sum_{i=1}^n \alpha_i$, and non-negative integers α_i . Moreover, $\bar{Q}$ and P are structured matrices including rectangular blocks of appropriate dimensions as below,

$$
\bar{Q} = \begin{bmatrix} 0 & 0 & 0 \\ 0 & Q & 0 \\ 0 & 0 & 0 \end{bmatrix}, \text{and} \quad P = \begin{bmatrix} P_1 & P_2 & P_3 \\ P_2^T & P_4 & P_5 \\ P_3^T & P_5^T & P_6 \end{bmatrix}.
$$

By substituting $\bar{Q}$,P in (10), it is easy to investigate that starting from the initial condition $P(0) = \mathbf{0}$, any term in the equations of $\dot{P}_1$, $\dot{P}_2$, and $\dot{P}_3$ will depend on P_1, P_2, or P_3 as below

$$
\begin{aligned}
\dot{P}_1 = &- P_2K_1K_1^TP_2^T - P_3\Gamma_1K_1K_1^TP_2^T - P_2K_1K_1^TP_3^T \\
&- P_3\Gamma_1K_1K_1^TP_3^T, \\
\dot{P}_2 = &P_2W_2 + P_3\Gamma_1W_2 - P_2K_1K_1^TP_4 - P_3\Gamma_1K_1K_1^TP_4 \\
&- P_2K_1K_1^TP_5^T - P_3\Gamma_1K_1K_1^TP_5^T, \\
\dot{P}_3 = &P_2W_3 + P_3\Gamma_1W_3 - P_2K_1K_1^TP_5 - P_3\Gamma_1K_1K_1^TP_5 \\
&- P_2K_1K_1^TP_6 - P_3\Gamma_1K_1K_1^TP_6,
\end{aligned}
$$

where $K_1 = W_{j_1} + W_{j_2}x + W_{j_3}\Gamma_1 x$. This choice of the initial condition is justified by the fact that no initial controller is assumed in this framework. Therefore, considering the zero derivatives and the zero initial conditions, we can conclude that solutions P_1, P_2, and P_3 will stay at zeros, and the matrix P will only grow on the block $\begin{bmatrix} P_4 & P_5 \\ P_5^T & P_6 \end{bmatrix}$. Therefore, for brevity, we will follow the computations only for this block as long as this simplification does not cause ambiguity. Now, let us take one step back and start with

$$
\begin{aligned}
\Phi(x)^T\dot{P}\Phi = &\Phi^T\bar{Q}\Phi + \Phi^TP\frac{\partial\Phi}{\partial x}W\Phi + \Phi^TW^T\frac{\partial\Phi}{\partial x}^TP\Phi \\
&- \Phi^TP\Phi_xW_j\Phi r_j^{-1}\Phi^TW_j^T\frac{\partial\Phi}{\partial x}^TP\Phi - \gamma\Phi^TP\Phi,
\end{aligned} \tag{27}
$$

which is given by (9) reversed in time to obtain the equation in forward time. For the non-zero block of P_4, P_5, and P_6 with the corresponding bases, the left-hand side can be rewritten as

$$
\begin{aligned}
&\begin{bmatrix} 1 \\ x \\ \Gamma_1 x + O(x^2) \end{bmatrix}^T \begin{bmatrix} 0 & 0 & 0 \\ 0 & \dot{P}_4 & \dot{P}_5 \\ 0 & \dot{P}_5^T & \dot{P}_6 \end{bmatrix} \begin{bmatrix} 1 \\ x \\ \Gamma_1 x + O(x^2) \end{bmatrix} = \begin{bmatrix} 1 \\ x \\ O(x^2) \end{bmatrix}^T \\
&\begin{bmatrix} 0 & 0 & 0 \\ 0 & \dot{P}_4 + \Gamma_1^T\dot{P}_5^T + \dot{P}_5\Gamma_1 + \Gamma_1^T\dot{P}_6\Gamma_1 & \dot{P}_5 + \Gamma_1^T\dot{P}_6 \\ 0 & \dot{P}_5^T + \dot{P}_6\Gamma_1 & \dot{P}_6 \end{bmatrix} \begin{bmatrix} 1 \\ x \\ O(x^2) \end{bmatrix},
\end{aligned} \tag{28}
$$

where we shifted the linear term in the third entry of the bases to the second. In the next step, we will consider the following change of variables throughout the matrix differential equation:

$$
\begin{aligned}
Z_1 &= P_4 + \Gamma_1^TP_5^T + P_5\Gamma_1 + \Gamma_1^TP_6\Gamma_1, \\
Z_2 &= P_5 + \Gamma_1^TP_6, \\
Z_3 &= P_6.
\end{aligned} \tag{29}
$$

For this reason, we apply the same modification of bases to all the terms on the right-hand side of (27). The modification will not affect the first term since $\bar{Q}$ is zero everywhere except in the block corresponding to the second basis, which remained unchanged. Then, let us consider the second term on the right hand side which becomes

$$\begin{aligned}
&\Phi^T P \frac{\partial \Phi}{\partial x} W \Phi = \\
&\begin{bmatrix} 1 \\ x \\ \Gamma_1 x + O(x^2) \end{bmatrix}^T \begin{bmatrix} 0 & 0 & 0 \\ 0 & P_4 & P_5 \\ 0 & P_5^T & P_6 \end{bmatrix} \begin{bmatrix} 0 \\ \mathbf{I} \\ \Gamma_1 + O(x) \end{bmatrix} \\
&\qquad \begin{bmatrix} 0 & W_2 & W_3 \end{bmatrix} \begin{bmatrix} 1 \\ x \\ \Gamma_1 x + O(x^2) \end{bmatrix} \\
&= \begin{bmatrix} 1 \\ x \\ \Gamma_1 x + O(x^2) \end{bmatrix}^T \begin{bmatrix} 0 & 0 & 0 \\ 0 & \Upsilon_1 & \Upsilon_2 \\ 0 & \Upsilon_3 & \Upsilon_4 \end{bmatrix} \begin{bmatrix} 1 \\ x \\ \Gamma_1 x + O(x^2) \end{bmatrix} \\
&= \begin{bmatrix} 1 \\ x \\ O(x^2) \end{bmatrix}^T \begin{bmatrix} 0 & 0 & 0 \\ 0 & \bar{\Upsilon}_1 & \bar{\Upsilon}_2 \\ 0 & \bar{\Upsilon}_3 & \bar{\Upsilon}_4 \end{bmatrix} \begin{bmatrix} 1 \\ x \\ O(x^2) \end{bmatrix},
\end{aligned} \tag{30}$$

where

$$\begin{aligned}
\Upsilon_1 &= P_4 W_2 + P_5 \Gamma_1 W_2 + P_5 O(x) W_2, \\
\Upsilon_2 &= P_4 W_3 + P_5 \Gamma_1 W_3 + P_5 O(x) W_3, \\
\Upsilon_3 &= P_5^T W_2 + P_6 \Gamma_1 W_2 + P_6 O(x) W_2, \\
\Upsilon_4 &= P_5^T W_3 + P_6 \Gamma_1 W_3 + P_6 O(x) W_3, \\
\bar{\Upsilon}_1 &= Z_1 (W_2 + W_3 \Gamma_1) + Z_2 O(x)(W_2 + W_3 \Gamma_1), \\
\bar{\Upsilon}_2 &= Z_1 W_3 + Z_2 O(x) W_3, \\
\bar{\Upsilon}_3 &= Z_2^T (W_2 + W_3 \Gamma_1) + Z_3 O(x)(W_2 + W_3 \Gamma_1), \\
\bar{\Upsilon}_4 &= Z_2^T W_3 + Z_3 O(x) W_3,
\end{aligned}$$

and we used (29) to get

$$\begin{aligned}
&P_4 W_2 + P_5 \Gamma_1 W_2 + P_4 W_3 \Gamma_1 + P_5 \Gamma_1 W_3 \Gamma_1 \\
&\quad + \Gamma_1^T P_5^T W_2 + \Gamma_1^T P_6 \Gamma_1 W_2 + \Gamma_1^T P_5^T W_3 \Gamma_1 + \Gamma_1^T P_6 \Gamma_1 W_3 \Gamma_1 \\
&\quad = (P_4 + \Gamma_1^T P_5^T + P_5 \Gamma_1 + \Gamma_1^T P_6 \Gamma_1)(W_2 + W_3 \Gamma_1) \\
&\quad = Z_1 (W_2 + W_3 \Gamma_1),
\end{aligned}$$

$$\begin{aligned}
&P_5 O(x) W_2 + P_6 O(x) W_3 \Gamma_1 + \Gamma_1^T P_6 O(x) W_2 + \Gamma_1^T P_6 O(x) W_3 \Gamma_1 \\
&\quad = Z_2 O(x)(W_2 + W_3 \Gamma_1),
\end{aligned}$$

$$\begin{aligned}
&P_4 W_3 + P_5 \Gamma_1 W_3 + P_5 O(x) W_3 + \Gamma_1^T P_5^T W_3 + \Gamma_1^T P_6 \Gamma_1 W_3 \\
&\qquad + \Gamma_1^T P_6 O(x) W_3 \\
&\quad = (P_4 + P_5 \Gamma_1 + \Gamma_1^T P_5^T) W_3 + (P_5 + \Gamma_1^T P_6) O(x) W_3 \\
&\qquad + \Gamma_1^T P_6 \Gamma_1 W_3 \\
&\quad = (Z_1 - \Gamma_1^T Z_3 \Gamma 1) W_3 + Z_2 O(x) W_3 + \Gamma_1^T Z_3 \Gamma_1 W_3 \\
&\quad = Z_1 W_3 + Z_2 O(x) W_3,
\end{aligned}$$

$$\begin{aligned}
&P_5^T W_2 + P_6 \Gamma_1 W_2 + P_6 O(x) W_2 + P_5^T W_3 \Gamma_1 + P_6 \Gamma_1 W_3 \Gamma_1 \\
&\qquad + P_6 O(x) W_3 \Gamma_1 \\
&\quad = (P_5^T + P_6 \Gamma_1)(W_2 + W_3 \Gamma_1) + P_6 O(x)(W_2 + W_3 \Gamma_1) \\
&\quad = Z_2^2 (W_2 + W_3 \Gamma_1) + Z_3 O(x)(W_2 + W_3 \Gamma_1),
\end{aligned}$$

$$\begin{aligned}
&P_5^T W_3 + P_6 \Gamma_1 W_3 + P_6 O(x) W_3 \\
&\quad = Z_2^T W_3 + Z_3 O(x) W_3.
\end{aligned}$$

Furthermore, for the last term in the right-hand side of (27), the following hold.

$$\begin{aligned}
&\Phi(x)^T P \frac{\partial \Phi}{\partial x} \sum_{j=1}^{m} (W_j \Phi r_j^{-1} \Phi^T W_j^T) \frac{\partial \Phi}{\partial x}^T P \Phi \\
&= \begin{bmatrix} 1 \\ x \\ \Gamma_1 x + O(x^2) \end{bmatrix}^T \begin{bmatrix} 0 & 0 & 0 \\ 0 & P_4 & P_5 \\ 0 & P_5^T & P_6 \end{bmatrix} \begin{bmatrix} 0 \\ \mathbf{I} \\ \Gamma_1 + O(x) \end{bmatrix} \\
&\sum_{j=1}^{m} \Big(\begin{bmatrix} W_{j_1} & W_{j_2} & W_{j_3} \end{bmatrix} \begin{bmatrix} 1 \\ x \\ \Gamma_1 x + O(x^2) \end{bmatrix} r_j^{-1} \begin{bmatrix} 1 \\ x \\ \Gamma_1 x + O(x^2) \end{bmatrix}^T \\
&\quad \begin{bmatrix} W_{j_1} \\ W_{j_2} \\ W_{j_3} \end{bmatrix} \Big) \begin{bmatrix} 0 \\ \mathbf{I} \\ \Gamma_1 + O(x) \end{bmatrix}^T \begin{bmatrix} 0 & 0 & 0 \\ 0 & P_4 & P_5 \\ 0 & P_5^T & P_6 \end{bmatrix} \begin{bmatrix} 1 \\ x \\ \Gamma_1 x + O(x^2) \end{bmatrix} \\
&= \begin{bmatrix} 1 \\ x \\ \Gamma_1 x + O(x^2) \end{bmatrix}^T \begin{bmatrix} 0 & 0 & 0 \\ 0 & P_4 & P_5 \\ 0 & P_5^T & P_6 \end{bmatrix} \begin{bmatrix} 0 \\ \mathbf{I} \\ \Gamma_1 + O(x) \end{bmatrix} \Omega_2 \\
&\quad \begin{bmatrix} 0 \\ \mathbf{I} \\ \Gamma_1 + O(x) \end{bmatrix}^T \begin{bmatrix} 0 & 0 & 0 \\ 0 & P_4 & P_5 \\ 0 & P_5^T & P_6 \end{bmatrix} \begin{bmatrix} 1 \\ x \\ \Gamma_1 x + O(x^2) \end{bmatrix} \\
&= \begin{bmatrix} 1 \\ x \\ \Gamma_1 x + O(x^2) \end{bmatrix}^T \Big(\begin{bmatrix} 0 \\ P_4 + P_5 \Gamma_1 \\ P_5^T + P_6 \Gamma_1 \end{bmatrix} + \begin{bmatrix} 0 \\ P_5 O(x) \\ P_6 O(x) \end{bmatrix} \Big) \Omega_2 \\
&\quad \Big(\begin{bmatrix} 0 \\ P_4 + P_5 \Gamma_1 \\ P_5^T + P_6 \Gamma_1 \end{bmatrix} + \begin{bmatrix} 0 \\ P_5 O(x) \\ P_6 O(x) \end{bmatrix} \Big)^T \begin{bmatrix} 1 \\ x \\ \Gamma_1 x + O(x^2) \end{bmatrix} \\
&= \begin{bmatrix} 1 \\ x \\ \Gamma_1 x + O(x^2) \end{bmatrix}^T \begin{bmatrix} 0 & 0 & 0 \\ 0 & \Omega_3 & \Omega_4 \\ 0 & \Omega_4^T & \Omega_5 \end{bmatrix} \begin{bmatrix} 1 \\ x \\ \Gamma_1 x + O(x^2) \end{bmatrix} \\
&+ \begin{bmatrix} 1 \\ x \\ \Gamma_1 x + O(x^2) \end{bmatrix}^T \begin{bmatrix} 0 & 0 & 0 \\ 0 & \Omega_6 & \Omega_7 \\ 0 & \Omega_7^T & \Omega_8 \end{bmatrix} \begin{bmatrix} 1 \\ x \\ \Gamma_1 x + O(x^2) \end{bmatrix} \\
&= \begin{bmatrix} 1 \\ x \\ O(x^2) \end{bmatrix}^T \begin{bmatrix} 0 & 0 & 0 \\ 0 & \bar{\Omega}_3 & \bar{\Omega}_4 \\ 0 & \bar{\Omega}_4^T & \bar{\Omega}_5 \end{bmatrix} \begin{bmatrix} 1 \\ x \\ O(x^2) \end{bmatrix} \\
&\quad + \begin{bmatrix} 1 \\ x \\ O(x^2) \end{bmatrix}^T \begin{bmatrix} 0 & 0 & 0 \\ 0 & \bar{\Omega}_6 & \bar{\Omega}_7 \\ 0 & \bar{\Omega}_7^T & \bar{\Omega}_8 \end{bmatrix} \begin{bmatrix} 1 \\ x \\ O(x^2) \end{bmatrix},
\end{aligned} \tag{31}$$

where Ω_2 to Ω_8 are defined as

$$\begin{aligned}
&\Omega_2 = \sum_{j=1}^{m} r_j^{-1} \begin{bmatrix} W_{j_1} & W_{j_2} & W_{j_3} \end{bmatrix} \begin{bmatrix} 1 \\ x \\ \Gamma_1 x \end{bmatrix} \begin{bmatrix} 1 \\ x \\ \Gamma_1 x \end{bmatrix}^T \begin{bmatrix} W_{j_1} \\ W_{j_2} \\ W_{j_3} \end{bmatrix}^T \\
&= \sum_{j=1}^{m} r_j^{-1} \begin{bmatrix} W_{j_1} & W_{j_2} & W_{j_3} \end{bmatrix}
\end{aligned}$$

$$
\begin{aligned}
&\begin{bmatrix} 1 & x^T & x^T\Gamma_1^T \\ x & xx^T & xx^T\Gamma_1^T \\ \Gamma_1 x & \Gamma_1 xx^T & \Gamma_1 xx^T\Gamma_1^T \end{bmatrix}\begin{bmatrix} W_{j_1} \\ W_{j_2} \\ W_{j_3} \end{bmatrix}^T \\
&= \sum_{j=1}^{m} \big(W_{j_1}W_{j_1}^T + W_{j_2}xW_{j_1}^T + W_{j_3}\Gamma_1 xW_{j_1}^T + \\
&\qquad W_{j_1}x^TW_{j_2}^T + W_{j_2}xx^TW_{j_2}^T + W_{j_3}\Gamma_1 xx^TW_{j_2}^T + \\
&\qquad W_{j_1}x^T\Gamma_1^TW_{j_3}^T + W_{j_2}xx^T\Gamma_1^TW_{j_2}^T \\
&\qquad + W_{j_3}\Gamma_1 xx^T\Gamma_1^TW_{j_3}^T\big)r_j^{-1}, \\
&\Omega_3 = (P_4 + P_5\Gamma_1)\Omega_2(P_4 + \Gamma_1^TP_5^T), \\
&\Omega_4 = (P_4 + P_5\Gamma_1)\Omega_2(P_5 + \Gamma_1^TP_6), \\
&\Omega_5 = (P_5^T + P_6\Gamma_1)\Omega_2(P_5 + \Gamma_1^TP_6), \\
&\Omega_6 = (P_4 + P_5\Gamma_1)\Omega_2O(x)P_5^T + P_5O(x)\Omega_2(P_4 + \Gamma_1^TP_5^T) \\
&\qquad + P_5O(x^2)P_5^T, \\
&\Omega_7 = (P_4 + P_5\Gamma_1)\Omega_2O(x)P_6 + P_5O(x)\Omega_2(P_5 + \Gamma_1^TP_6) \\
&\qquad + P_5O(x^2)P_6, \\
&\Omega_8 = (P_5^T + P_6\Gamma_1)\Omega_2O(x)P_6 + P_6O(x)\Omega_2(P_5 + \Gamma_1^TP_6) \\
&\qquad + P_6O(x^2)P_6.
\end{aligned}
$$

Moreover, $\bar{\Omega}_3$ to $\bar{\Omega}_3$, are their analogous blocks after modifying the bases, where they can also be rewritten in terms of the new variables defined in (29) as below

$$
\begin{aligned}
\bar{\Omega}_3 &= (P_4 + \Gamma_1^TP_5^T + P_5\Gamma_1 + \Gamma_1^TP_6\Gamma_1)\Omega_2 \\
&\quad (P_4 + \Gamma_1^TP_5^T + P_5\Gamma_1 + \Gamma_1^TP_6\Gamma_1)^T \\
&= Z_1W_{j_1}W_{j_1}^TZ_1, \\
\bar{\Omega}_4 &= (P_4^T + P_5\Gamma_1 + \Gamma_1^TP_5^T + \Gamma_1^TP_6\Gamma_1)\Omega_2(P_5 + \Gamma_1^TP_6) \\
&= Z_1W_{j_1}W_{j_1}^TZ_2, \\
\bar{\Omega}_5 &= \Omega_5 = Z_2^TW_{j_1}W_{j_1}^TZ_2, \\
\bar{\Omega}_6 &= \Omega_6 + \Gamma_1^T\Omega_7^T + \Omega_7\Gamma_1 + \Gamma_1^T\Omega_8\Gamma_1 \\
&= (P_4 + P_5\Gamma_1 + \Gamma_1^TP_5^T + \Gamma_1^TP_6\Gamma_1)\Omega_2O(x)P_5^T \\
&\quad + (P_5 + \Gamma_1^TP_6)O(x)\Omega_2(P_4 + \Gamma_1^TP_5^T) \\
&\quad + (P_4 + P_5\Gamma_1 + \Gamma_1^TP_5^T + \Gamma_1^TP_6\Gamma_1)\Omega_2O(x)P_6\Gamma_1 \\
&\quad + (P_5 + \Gamma_1^TP_6)O(x)\Omega_2(P_5 + \Gamma_1^TP_6)\Gamma_1 \\
&\quad + (P_5 + \Gamma_1^TP_6)O(x^2)P_5^T + (P_5 + \Gamma_1^TP_6)O(x^2)P_6\Gamma_1 \\
&= (P_4 + P_5\Gamma_1 + \Gamma_1^TP_5^T + \Gamma_1^TP_6\Gamma_1)\Omega_2O(x)(P_5^T + P_6\Gamma_1) \\
&\quad + (P_5 + \Gamma_1^TP_6)O(x)\Omega_2(P_4 + \Gamma_1^TP_5^T + P_5\Gamma_1 + \Gamma_1^TP_6\Gamma_1) \\
&\quad + (P_5 + \Gamma_1^TP_6)O(x^2)(P_5^T + P_6\Gamma_1) \\
&= Z_1\Omega_2O(x)Z_2^T + Z_2O(x)\Omega_2Z_1 + Z_2O(x^2)Z_2^T, \\
\bar{\Omega}_7 &= \Omega_7 + \Gamma_1^T\Omega_8 = Z_1\Omega_2O(x)Z_3 + Z_2O(x)\Omega_2Z_2 + Z_2O(x^2)Z_3, \\
\bar{\Omega}_8 &= \Omega_8 = Z_2^T\Omega_2O(x)Z_3 + Z_3O(x)\Omega_2Z_2 + Z_3O(x^2)Z_3.
\end{aligned}
$$

To derive these equations we also used the fact that the constant term will dominate in Ω_2 as $x \to 0$. Hence, we get $\Omega_2 \to \sum_{j=1}^{m} r_j^{-1}W_{j_1}W_{j_1}^T$.

By substituting (28), (30), and (31) in (27), one can obtain

$$
\begin{aligned}
\dot{Z}_1 =& Q + Z_1(W_2 + W_3\Gamma_1) + (W_2^T + \Gamma_1^TW_3^T)Z_1 \\
&- Z_1(\sum_{j=1}^{m} W_{j_1}r_j^{-1}W_{j_1}^T)Z_1 + Z_2O(x)(W_2 + W_3\Gamma_1) \\
&- \gamma Z_1 + (W_2 + W_3\Gamma_1)^TO(x)Z_2^T - Z_1\Omega_2O(x)Z_2^T \\
&- Z_2O(x)\Omega_2Z_1 - Z_2O(x^2)Z_2^T, \qquad (32)
\end{aligned}
$$

$$
\begin{aligned}
\dot{Z}_2 =& Z_1W_3 + (W_2 + W_3\Gamma_1)^TZ_2 - Z_1(\sum_{j=1}^{m} W_{j_1}r_j^{-1}W_{j_1}^T)Z_2 \\
&- \gamma Z_2 + Z_2O(x)W_3 + (W_2 + W_3\Gamma_1)^TO(x)Z_3^T \\
&- Z_1\Omega_2O(x)Z_3 - Z_2O(x)\Omega_2Z_2 - Z_2O(x^2)Z_3, \qquad (33)
\end{aligned}
$$

$$
\begin{aligned}
\dot{Z}_3 =& Z_2^TW_3 + W_3^TZ_2 - Z_2(\sum_{j=1}^{m} W_{j_1}r_j^{-1}W_{j_1}^T)Z_2 \\
&- \gamma Z_3 + Z_3O(x)W_3 + W_3^TO(x)Z_3^T - Z_2^T\Omega_2O(x)Z_3 \\
&- Z_3O(x)\Omega_2Z_2 - Z_3O(x^2)Z_3. \qquad (34)
\end{aligned}
$$

Moreover, for the optimal control (8) takes the following form,

$$
\begin{aligned}
u_j^* =& -\begin{bmatrix} 1 \\ x \\ \Gamma_1 x + O(x^2) \end{bmatrix}^T r_j^{-1}\begin{bmatrix} 0 & 0 & 0 \\ 0 & P_4 & P_5 \\ 0 & P_5^T & P_6 \end{bmatrix}\begin{bmatrix} 0 \\ \mathbf{I} \\ \Gamma_1 + O(x) \end{bmatrix} \\
&\begin{bmatrix} W_{j_1} & W_{j_2} & W_{j_3} \end{bmatrix}\begin{bmatrix} 1 \\ x \\ \Gamma_1 x + O(x^2) \end{bmatrix} \\
=& -\begin{bmatrix} 1 \\ x \\ \Gamma_1 x + O(x^2) \end{bmatrix}^T r_j^{-1}\begin{bmatrix} 0 \\ P_4 + P_5\Gamma_1 \\ P_5^T + P_6\Gamma_1 \end{bmatrix} \\
&\begin{bmatrix} W_{j_1} & W_{j_2} & W_{j_3} \end{bmatrix}\begin{bmatrix} 1 \\ x \\ \Gamma_1 x + O(x^2) \end{bmatrix} \\
=& -r_j^{-1}x^T(P_4 + P_5\Gamma_1 + \Gamma_1^TP_5^T + \Gamma_1^TP_6\Gamma_1) \\
&(W_{j_1} + W_{j_2}x + W_{j_3}\Gamma_1 x) + O(x^3) \\
=& -r_j^{-1}x^T(P_4 + P_5\Gamma_1 + \Gamma_1^TP_5^T + \Gamma_1^TP_6\Gamma_1)W_{j_1} \\
&- r_j^{-1}x^T(P_4 + P_5\Gamma_1 + \Gamma_1^TP_5^T + \Gamma_1^TP_6\Gamma_1) \\
&(W_{j_2}x + W_{j_3}\Gamma_1 x) + O(x^3) \\
=& -r_j^{-1}x^TZ_1W_{j_1} + H_1(Z_1)O(x^2), \qquad (35)
\end{aligned}
$$

By substituting the control in the system, we obtain the closed-loop system as

$$
\begin{aligned}
\dot{x} =& \begin{bmatrix} 0 & W_2 & W_3 \end{bmatrix}\begin{bmatrix} 1 \\ x \\ \Gamma_1 x + O(x^2) \end{bmatrix} \\
&+ \sum_{j=1}^{m}\begin{bmatrix} W_{j_1} & W_{j_2} & W_{j_3} \end{bmatrix}\begin{bmatrix} 1 \\ x \\ \Gamma_1 x + O(x^2) \end{bmatrix} u_j \\
=& (W_2 + W_3\Gamma_1)x + O(x^2) \\
&+ \sum_{j=1}^{m}(W_{j_1} + W_{j_2}x + W_{j_3}\Gamma_1 x + O(x^2))u_j \\
=& (A - GZ_1 + H_2(Z_1)O(x))x, \qquad (36)
\end{aligned}
$$

where the linear term will dominate as $x \in \bar{D}$. It should be noted that, among the solutions of (32) to (34), only $Z_1(t)$ is directly effective in the control. Equation (32) can be summarized as

$$
\dot{Z}_1 = Q + Z_1A + A^TZ_1 - Z_1(\sum_{j=1}^{m} B_{j_1}r_j^{-1}B_{j_1}^T)Z_1
$$

$$- \gamma Z_1 + O(x) H_3(Z_{1,2}) \tag{37}$$

where we used H_3 to lump together the terms related to Z_1 and Z_2. Equation (37) resembles the Riccati equation for the linearized system (12) with higher-order terms of x and the discounting term.

Now, suppose that a trajectory stays near the origin, i.e. $\|x(t)\|$ is small, and γ is set to be small. Then, $-\gamma Z_1 + O(x)H_3(Z_{1,2})$ can be made arbitrarily small within some radius $r_1 > 0$ from the origin. Moreover, regarding that the equation without these extra terms exponentially converges to the ARE solution [27], Z_1 can be kept close to the solution of (14) with the bounded extra terms, where the distance clearly depends on r_1 and γ. Let us define $S_\delta = Z_1 - S$ where $\|S_\delta\| \leq \delta_r$.

Next, we demonstrate that the obtained control is also stabilizing for the nonlinear closed-loop system (36). It is easy to verify that, $r_1 > 0$ can be chosen small enough so that the nonlinear closed-loop system (36) is also asymptotically stable by employing the Lyapunov function $V_1 = x^T \bar{S} x$ as below

$$\begin{aligned}
\dot{V}_1 &= \dot{x}^T \bar{S} x + x^T \bar{S} \dot{x} \\
&= x^T \big(A - GZ_1 + H_2(Z_1)O(x)\big)^T \bar{S} x + \\
&\quad x^T \bar{S} \big(A - GZ_1 + H_2(Z_1)O(x)\big) x \\
&= x^T \big(A^T \bar{S} + \bar{S} A - Z_1 G \bar{S} - \bar{S} G Z_1 + \bar{H}_2(Z_1)O(x)\big) x \\
&= x^T \big(A^T \bar{S} + \bar{S} A - (S + S_\delta) G \bar{S} - \bar{S} G (S + S_\delta) \\
&\qquad + \bar{H}_2(Z_1)O(x)\big) x \\
&= x^T (-Q + \bar{S} G \bar{S} - S G \bar{S} - \bar{S} G S - S_\delta G \bar{S} - \bar{S} G S_\delta \\
&\qquad + \bar{H}_2(Z_1)O(x)\big) x \\
&= -x^T \big(Q_2 + E_1(t)\big) x,
\end{aligned}$$

where we defined $H_2(Z_1)$ to collect all higher order terms and we used (13) and $E_1(t) = S_\delta G \bar{S} + \bar{S} G S_\delta - \bar{H}_2(Z_1)O(x)\big)x$ in the derivations. It is evident that $Q_2 = Q - \bar{S} G \bar{S} + S G \bar{S} + \bar{S} G S$ is a positive definite matrix since $S \to \bar{S}$, $Q > 0$ and $\bar{S} G \bar{S} \geq 0$. Now, $\bar{D}_{r_1}$ can be considered such that Q_2 dominates, and this results in $Q_2 + E_1(t) > 0$, hence, $\dot{V}_1 < 0$ is obtained, except at the origin. In conclusion, asymptotic stability is guaranteed within $\bar{D}_{r_1}$.

Moreover, assuring the asymptotic stability, we have $x \to 0$. Furthermore, if γ is made sufficiently small, then the steady state solution of (37) will converge to the steady state solution of (16), and hence to the solution of the algebraic Riccati equation (13), i.e $Z_1 \to \bar{S}$.

VIII. Appendix: Asymptotic Convergence with Approximate Dynamics

Consider the system structured as

$$\dot{x} = W\Phi + \sum_{j=1}^{m} W_j \Phi \hat{u}_j + \epsilon. \tag{38}$$

where $\hat{u}_j = -\Phi^T r_j^{-1} \hat{P} \Phi_x \hat{W}_j \Phi$ is the feedback control rule obtained based on the estimation of the system $(\hat{W}, \hat{W}_j)$. Moreover, ϵ is the bounded approximation error in D. By assuming $W = \hat{W} + \tilde{W}$ and $W_j = \hat{W}_j + \tilde{W}_j$, this can be rewritten as

$$\dot{x} = \hat{W}\Phi + \sum_{j=1}^{m} \hat{W}_j \Phi \hat{u}_j + \Delta(t),$$

where unidentified dynamics are lumped together as $\Delta(t)$. By the assumption that the feedback control u_j is bounded in D, we have $\|\Delta(t)\| \leq \bar{\Delta}$. For asymptotic convergence, and also to promote the robustness of the controller, the effect of the uncertainty should be taken into account. Hence, we use an auxiliary vector ρ to get

$$\begin{aligned}
\dot{x} &= \hat{W}\Phi + \sum_{j=1}^{m} \hat{W}_j \Phi \hat{u}_j + \Delta(t) + \rho - \rho \\
&= \hat{W}_\rho \Phi + \sum_{j=1}^{m} \hat{W}_j \Phi \hat{u}_j + \Delta(t) - \rho,
\end{aligned}$$

where assuming that Φ also includes the constant basis, we adjusted the corresponding column in the system matrix to get $\hat{W}_\rho$. In the case $\bar{\Delta} = 0$, by using Theorem 3.1, the controller $\hat{u}$ can be obtained such that the closed system is locally asymptotically stable. For the case $\bar{\Delta} > 0$, although the system will stay stable for small enough $\bar{\Delta}$, it may not asymptotically converge to zero. Then, similar to [34], [28], we obtain ρ as below to help sliding the system state to zero

$$\rho = \int_0^t [k_1 x(\tau) + k_2 \mathrm{sign}\big(x(\tau)\big)] \mathrm{d}\tau,$$

where k_1 and k_2 are positive scalars. It can be shown that over time $\|\Delta(t) - \rho\| \to 0$, and hence the system will asymptotically converge to the origin.

IX. Appendix: Experiments Details

The approach employed in this paper is implemented using Python on a Windows 11 (64-bit) operating system. The hardware setup consists of an Intel(R) Core(TM) i7-12700H processor with 16.0 GB of RAM.

In what follows, we discuss the main steps involved in Algorithm 1 in more detail.

A. Control Update

To initiate the learning process, we begin by executing the system with an initial value of x_0 from the domain D. We then solve the matrix differential equation (10) as the system evolves. In our simulation, we employ a Runge-Kutta solver to numerically integrate the system dynamics, serving as a substitute for the actual system in a real-world application. During this simulation, the solver may take smaller time steps, but we only allow measurements and control updates at specific time instances $t_k = kh$, where h represents the sampling time, and k takes on values $0, 1, 2, \ldots$.

To solve the differential equation (10) in continuous time, we employ the Runge-Kutta solver with a similar configuration. Here, the weights and states in the equation are updated using a system identification algorithm and the measurements

x_k are obtained at each iteration of the control loop, respectively. It is advisable to initialize the matrix $P(0)$ as a zero matrix.

For (10), it is also necessary to compute the partial derivatives $\partial\Phi/\partial x_k$ at each time step. As the basis functions Φ are predetermined, these partial derivatives can be analytically calculated and stored as functional representations. Consequently, they can be evaluated for any x_k in a manner similar to evaluating Φ itself. By solving equation (10), we can determine the control update at any given time step t_k using equation (8).

In the examples presented in this paper, we choose the basis vector Φ as given in Table. III.

System	Φ
Linear System	$[1, x_1, x_2, x_3]$
Inverted Pendulum	$[1, x_1, x_2, sinx_1, sinx_2]$
Cartpole	$[1, x_1, \ldots, x_4, x_1^2, \ldots, x_4^2, sinx_1, \ldots, sinx_4]$
Quadrotor	$[1, x_1, \ldots, x_{12}]$

TABLE III: Basis vector Φ chosen

a) Computational complexity:: The computational complexity involved in updating the parameters using equation (10) is determined by the matrix multiplications of size p. It is worth mentioning that due to the symmetry of the parameter matrix P, this equation updates a total of $L = (p^2 + p)/2$ parameters, which corresponds to the number of bases utilized in the value function. Therefore, in terms of the number of parameters, the complexity of the suggested approach is of the order $\mathcal{O}(L^{3/2})$.

B. Model Update

We considered a given structured nonlinear system as in 2. Therefore, having the control and state samples of the system, we need an algorithm that updates the estimation of system weights. Different system identification techniques found in the literature can be employed. For Pendulum and Cartpole systems we used Sparse Identification of Nonlinear Dynamics (SINDy) for this purpose. As studied in [6], [15], SINDy is a data-efficient tool to extract the underlying sparse dynamics of the sampled data. Hence, we use SINDy to update the weights of the system to be learned. In this approach, along with the identification, the sparsity is also promoted in the weights by minimizing

$$[\hat{W} \quad \hat{W}_1 \ldots \hat{W}_m]_k = \arg\min_{\bar{W}} \|\dot{X}_k - \bar{W}\Theta_k\|_2^2 + \lambda\|\bar{W}\|_1, \tag{39}$$

where k is the time step, $\lambda > 0$, and Θ_k includes a matrix of samples with the columns of

$${\Theta_k}^s = [\Phi^T(x^s) \quad \Phi^T(x^s){u_1}^s \quad \ldots \quad \Phi^T(x^s){u_m}^s]_k^T,$$

for sth sample. In the same order, $\dot{X}$ keeps a table of the states derivatives approximated by $\hat{\dot{x}}_k = (x_k - x_{k-1})/h$.

Updating $\hat{W}_k$ based on a history of samples may not be favored if the number of samples and parameters tend to be large. Especially, real-time implementations may not be possible because of the latency caused by the computations. Considering the linear dependence on the system weights in (2), one may choose the well-known Recursive Least Squares (RLS) update rule that only uses the latest sample of the system and $\hat{W}_{k-1}$, hence will run considerably faster. Considering that the quadrotor system is of higher dimension and higher number of inputs compared to the other given examples, it will involve many parameters. Hence for efficiency, we used RLS instead of SINDy. It can be shown that in this case, the following optimization problem is solved in an online scheme

$$[\hat{W} \quad \hat{W}_1 \ldots \hat{W}_m]_k = \arg\min_{\bar{W}} \|\dot{X}_k - \bar{W}\Theta_k\|_2^2.$$

C. Dataset Update

For using SINDy algorithm, unlike RLS, a dataset of samples is required to recursively perform regressions at each time step. These weights correspond to a library of functions given in Φ. Any sample of the system at time k, includes ${\Theta_k}^s$ and the derivatives of the states approximated by $\hat{\dot{x}}_k$. Samples are stored as matrices $(\dot{X}_k, \Theta_k)$, which can vary over time.

We perform SINDy updates along with the system trajectories, meaning that the dataset has to be gradually built along with the exploration and control. Different approaches can be employed in the choice of samples and building a dataset online. A comparison of these techniques can be found in [18], [31].

We assume a given maximum size of dataset $N_d = 2000$, then we keep adding the samples with larger prediction errors to it. Therefore, at any step we compare the prediction error $\dot{e}_k = \|\dot{x}_k - \hat{\dot{x}}_k\|$ with the average $\bar{\dot{e}}_k = \sum_{i=1}^k \dot{e}_k/k$. Hence, if the condition $\dot{e}_k > \eta\bar{\dot{e}}_k$ holds we add the sample to the database, where the constant $0 < \eta < 1$ adjusts the threshold. Choosing smaller values of η will increase the rate of adding samples to the database. If the maximum number of samples in the dataset is reached, we forget the oldest sample and replace it with the recent one. Therefore, η should not be set too low to avoid fast forgetting of the older useful samples. For the quadrotor example, we set $\eta = 0.9$.

X. Appendix: Supplementary results

A. Control cost curves for cartpole and quadrotor

To assess the performance of SOL, PPO, and ADP, we run the trained models on the cartpole and quadrotor environment. The control cost at each time step is measured by computing $x^TQx + u^TRu$ at the time step. This is repeated for 100 trajectories with randomly sampled initial conditions. The initial position, velocity, angle, and angular velocity in the cartpole system are samples from the same continuous uniform distribution $U(a, b)$ defined by the bounds a and b. We plot the mean control cost and the standard deviation (shown by the shaded area) below.

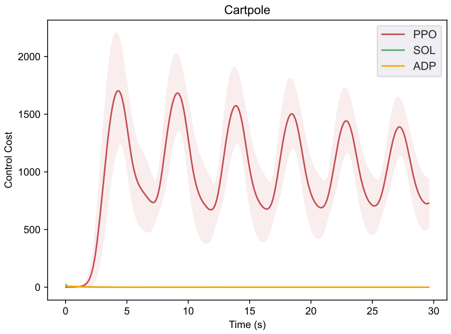


Fig. 4: Control cost curves for each algorithm on the cartpole environment averaged over 100 trajectories. The shaded area indicates one standard deviation. Initial conditions are sampled from $U(-0.05, 0.05)$.

Figure 4 shows that the controller returned from PPO shows unstable behavior and oscillation around the stable equilibrium point. SOL and ADP instead return stable controllers that converge to the equilibrium point. For a comparison between the two stable controllers, we plotted their performance on the cartpole environment with a better resolution and a wider range of initial conditions in Figure 5.

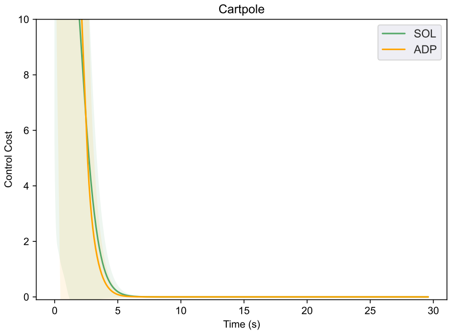


Fig. 5: Control cost curves for SOL and ADP on the cartpole environment averaged over 100 trajectories. The shaded area indicates one standard deviation. Initial conditions are sampled from $U(-0.2, 0.2)$.

Thus, SOL can achieve comparable results to ADP without the requirement of an initially stable controller. In fact, obtaining an initially stable controller for ADP within a larger vicinity of the equilibrium point poses a significant challenge. Consequently, for a broader range of domains, ADP may yield extremely high costs, whereas SOL can still converge successfully. This is shown in Figure 6 where we widen the distribution of initial conditions to $U(-0.5, 0.5)$.

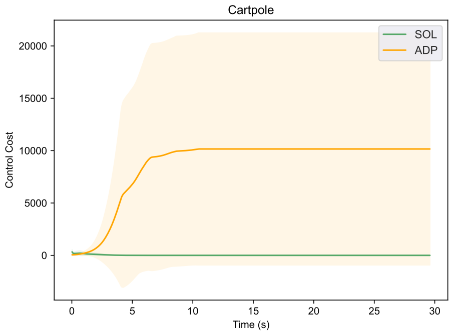


Fig. 6: Control cost curves for SOL and ADP on the cartpole environment averaged over 100 trajectories. The flat line region of ADP indicates that the agents have gone out of bounds. The shaded area indicates one standard deviation. Initial conditions are sampled from $U(-0.5, 0.5)$.

We finally compare the performance of PPO and SOL on the quadrotor environment, shown in Figure 7. Due to the requirement of an initial stable controller for ADP and the complexity that the quadrotor environment poses, no implementation was attempted. PPO shows high control costs and its trajectory goes out of bounds, whereas SOL returns a stable controller that converges to the equilibrium point.

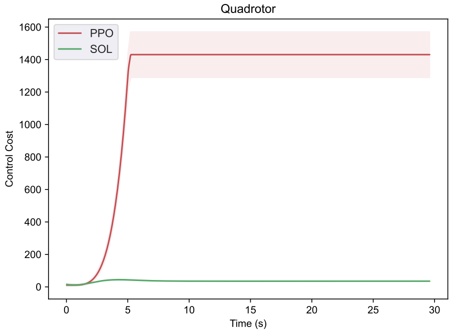


Fig. 7: Control cost curves for each algorithm on the quadrotor environment averaged over 100 trajectories. The flat line region of PPO indicates that the agents have gone out of bounds. The shaded area indicates one standard deviation.

B. Hyperparameters for PPO

Hyperparameter	Value
Horizon (T)	1500
Actor learning rate	3e-04
Critic learning rate	1e-03
Num. epochs	10
Minibatch size	64
Discount (γ)	0.99
GAE parameter (λ)	0.95

TABLE IV: Hyperparameters for PPO

XI. Examples: Dynamics of Systems

A. ***Example 1.*** *(A Linear System)*

Consider the following linear system

$$\dot{x} = \begin{bmatrix} 0 & 1 & 0 \\ 0 & 0 & 1 \\ -0.1 & -0.5 & -0.7 \end{bmatrix} x + \begin{bmatrix} 0 \\ 0 \\ 1 \end{bmatrix} u, \tag{40}$$

taken from [14]. The objective is defined by choosing $Q = I_3$ and $R = 1$.

1) ***Example 2****. (Inverted Pendulum):* The state space description of the system is given as

$$\begin{aligned} \dot{x_1} &= x_2, \\ \dot{x_2} &= \frac{g}{l}\sin(x_1) - \frac{k}{ml^2}x_2 + \frac{u}{ml^2}, \end{aligned} \tag{41}$$

where $m = 0.1kg$, $l = 0.5m$, $k = 0.1$, and $g = 9.8m/s^2$. The performance criteria are defined by the choices of $Q = \text{diag}([1, 1])$, $R = 2$.

2) ***Example 3****. (Cartpole):* The dynamics are given as

$$\begin{aligned} &\dot{x_1} = x_2, \\ &\dot{x_2} = \\ &\frac{-u\cos(x_1) - mLx_2^2\sin(x_1)\cos(x_1) + (M+m)g\sin(x_1)}{L(M + m\sin(x_1)^2)}, \\ &\dot{x_3} = x_4, \\ &\dot{x_4} = \frac{u + m\sin(x_1)(Lx_2^2 - g\cos(x_1))}{M + m\sin(x_1)^2}, \end{aligned} \tag{42}$$

where the state vector is composed of the angle of the pendulum from the upright position, the angular velocity, and the position and velocity of the cart, with $m = 0.1kg$, $M = 1kg$, $L = 0.8m$, and $g = 9.8m/s^2$. Moreover, we choose $Q = \text{diag}([60, 1.5, 180, 45])$, $R = 1$.

3) ***Example 4****. (Quadrotor Model):* The nonlinear dynamics of the quadrotor can be written as

$$\begin{aligned} \dot{y} &= v, \\ \dot{v} &= \begin{bmatrix} 0 \\ 0 \\ -g \end{bmatrix} + \frac{1}{m} R \begin{bmatrix} 0 \\ 0 \\ T \end{bmatrix}, \\ \dot{R} &= RQ(\omega), \\ \dot{\omega} &= J^{-1}(-\omega \times J\omega + \tau), \end{aligned} \tag{43}$$

where the states include the 3D position y, the linear velocity v of the center of gravity in the inertial frame, the rotation matrix R, and the angular velocity ω in the body frame with respect to the inertial frame. It should be noted that the third equation is written in a matrix form, where R takes value in the special orthogonal group SO(3) $= \{R \in \mathbb{R}^{3\times3} | R^{-1} = R^T, \det(R) = 1\}$. Accordingly, the attitude of the quadrotor $\chi = \begin{bmatrix} \phi & \theta & \psi \end{bmatrix}$ can be extracted from R at any time instance, which contains the roll, pitch, and yaw angles, respectively.

The inputs of this system are given by the moments in the body frame

$$\tau = \begin{bmatrix} C_T d(-\bar{\omega}_2^2 - \bar{\omega}_4^2 + \bar{\omega}_1^2 + \bar{\omega}_3^2) \\ C_T d(-\bar{\omega}_1^2 + \bar{\omega}_2^2 + \bar{\omega}_3^2 - \bar{\omega}_4^2) \\ C_D(\bar{\omega}_2^2 + \bar{\omega}_4^2 - \bar{\omega}_1^2 - \bar{\omega}_3^2) \end{bmatrix},$$

and the thrust

$$T = C_T(\bar{\omega}_1^2 + \bar{\omega}_2^2 + \bar{\omega}_3^2 + \bar{\omega}_4^2)$$

generated in the body frame by the rotors, where $\bar{\omega}_i$, d, C_T, and C_D denote the rotational speed of each rotor, the arm length, the lift, and the drag coefficients of the propellers, respectively. Model coefficients are given in Table V.

	Value[unit]		Value[unit]
m	0.33 [Kg]	d	39.73×10^{-3}[m]
I_{xx}	1.395×10^{-5}[Kg×m^2]	C_T	0.2025
I_{yy}	1.436×10^{-5}[Kg×m^2]	C_D	0.11
I_{zz}	2.173×10^{-5} [Kg×m^2]	g	0.98[m/s^2]

TABLE V: The coefficients of the simulated Crazyflie