Latent Linear Quadratic Regulator for Robotic Control Tasks

Yuan Zhang YZHANG@CS.UNI-FREIBURG.DE
University of Freiburg, Germany

Shaohui Yang SHAOHUI.YANG@EPFL.CH
EPFL, Switzerland

Toshiyuki Ohtsuka OHTSUKA@I.KYOTO-U.AC.JP
Kyoto University, Japan

Colin Jones COLIN.JONES@EPFL.CH
EPFL, Switzerland

Joschka Boedecker JBOEDECK@CS.UNI-FREIBURG.DE
University of Freiburg, Germany

Editors: G. Sukhatme, L. Lindemann, S. Tu, A. Wierman, N. Atanasov

Abstract

Model predictive control (MPC) offers high-performance control but remains computationally expensive for nonlinear dynamics, hindering its real-time deployment in robotic tasks. Inspired by the Koopman operator, we propose the **la**tent **l**inear **q**uadratic **r**egulator (LaLQR) framework, which learns an alternative latent linear-quadratic structure enabling efficient LQR-based control for nonlinear systems. LaLQR enforces the fixed Brunovsky canonical form on the latent linear dynamics to ensure controllability and numerical stability, while jointly learning a nonlinear embedding and cost function under latent state and cost prediction objectives. Experiments on diverse MuJoCo simulated robotic tasks show that LaLQR achieves comparable control quality to expensive gradient-based optimization methods while offering superior computational efficiency and better generalization over learning-based baselines.

Keywords: Model Predictive Control, Linear Quadratic Regulator, Imitation Learning, Robotics

1. Introduction

Model predictive control (MPC) has achieved remarkable success in robotic applications such as quadruped locomotion (Di Carlo et al., 2018; Grandia et al., 2023) and agile drone racing (Song and Scaramuzza, 2020). As a model-based control method, MPC optimizes future control sequences by predicting system evolution under a dynamic model $x_{h+1} = f_d(x_h, u_h)$, where x and u denote the system state and control input, respectively. However, the computational burden of solving MPC problems—particularly for nonlinear high-dimensional dynamics—remains a major obstacle to real-time deployment on embedded robotic platforms, e.g., humanoids (Katayama et al., 2023).

Various approaches have been proposed to reduce the complexity of nonlinear MPC. Sequential quadratic programming (SQP) (Nocedal and Wright, 1999) iteratively solves quadratic subproblems based on first- and second-order expansions of the nonlinear model and cost function, but the iterative nature of SQP often makes it too slow for real-time control. Local linearization methods, such as the localized LQR (LoLQR) (Wang et al., 2014), approximate the nonlinear system around an equilibrium point using a Taylor expansion and then apply the efficient LQR solver (Anderson and Moore, 2007). However, their performance quickly degrades when the system operates far from the

linearization point. More recently, imitation learning (IL) (Brohan et al., 2022) has been employed to bypass explicit optimization altogether by directly learning a mapping from states to actions using neural networks. Although IL can achieve excellent performance given sufficient expert data, it often exhibits poor generalization outside the training distribution.

In this work, we propose the **la**tent **l**inear **q**uadratic **r**egulator (LaLQR), a method that combines the efficiency of LQR with the flexibility of nonlinear representation inspired by the Koopman operator framework. Specifically, LaLQR employs a nonlinear embedding ϕ that maps the original state x_h into a latent state $z_h = \phi(x_h)$, where the dynamics evolve linearly and the cost is quadratic. One of our key contributions is the adoption of the Brunovsky canonical form (Brunovskỳ, 1970) to represent latent linear dynamics, owing to its extreme simplicity in structure and superior numerical stability during training. Therefore, only the embedding and cost functions are jointly learned by predicting the next latent state and cost. Subsequently, the LQR method is applied to compute the gain matrix, which, in conjunction with the embedding function, is utilized for efficient online control. Compared with learning-based methods such as IL, LaLQR preserves interpretability, stability, and generalization while reducing computation by an order of magnitude relative to SQP-based MPC. Experiments on multiple MuJoCo robotic control tasks demonstrate that LaLQR achieves a favorable trade-off among control performance, efficiency, and generalization.

2. Related Work

The Koopman Operator for Control. The Koopman operator has become increasingly popular in the control community due to its ability to transform the original state x into the latent state z, in which the system dynamics are linear in z and the control input u. This latent space can potentially assist the theoretical analysis and simplify the control tasks. To leverage the Koopman operator for optimal control, three components are typically required: (1) an embedding function ϕ that maps $x \mapsto z$ (some works (Mondal et al., 2024) also include a control mapping $u \mapsto v$); (2) a linear dynamic model parameterized by (A, B); (3) a stage cost function. Both (2) and (3) operate in the latent space. These parameters can be learned in data-driven techniques such as extended dynamic mode decomposition (eDMD) (Lusch et al., 2018). The learning principle of eDMD can be summarized as *latent state prediction* (LSP), which aligns $\phi(x_{h+1})$ and $A\phi(x_h) + Bu_h$, where h is the step index. Other learning principles have been proposed in the literature: (1) predicting the true cost from the latent state (*cost prediction*, CP) (Li et al., 2020b); (2) reconstructing the original state x from the latent representation (*state reconstruction*, SR) (Watter et al., 2015); (3) learning the cost-to-go function for each latent state(ϕ_{V^*}) (Mondal et al., 2024). Notably, these learning principles all shape the structure of the latent space z through gradient-based learning, effectively imposing different levels of abstraction on the original state x. Representative Koopman-based control methods are summarized in Table 1.

Previous works (Korda and Mezić, 2018; Bruder et al., 2019; Mamakoukas et al., 2019) heavily rely on prior knowledge of the embedding function (basis function) and cost matrices (Q, R), and primarily focus on learning the dynamic model (the Koopman operator) using the LSP principle. In contrast, our approach enables learnable embedding functions and cost matrices, thereby minimizing the required prior knowledge. A key feature of our method is the use of the Brunovsky canonical form $(\bar{A}_b, \bar{B}_b)$ (Brunovskỳ, 1970) for the latent dynamics. The fixed matrices maintain expressiveness through the nonlinear embedding while enhancing training stability (see Section 3.4). Another

1. The Brunovsky canonical form as in Equation 2.

Method	Embedding Function	Dynamic Model	Cost Function	Learning Principle	Optimization
Watter et al. (2015)	NN	$A_t z + B_t u$	$z^T \bar{Q} z + u^T \bar{R} u$	LSP, SR, reg.	iLQG
Korda and Mezić (2018)	BF	$Az + Bu$	$z^T \bar{Q} z + u^T \bar{R} u$	LSP	QP
Bruder et al. (2019)	BF	$Az + Bu$	$z^T \bar{Q}_t z + u^T \bar{R}_t u$	LSP, reg.	QP
Mamakoukas et al. (2019)	BF	$Az + Bu$	$z^T \bar{Q} z + u^T \bar{R} u$	LSP	LQR
Li et al. (2020a)	NN	$Az + Bu$	$z^T \bar{Q} z + u^T \bar{R} u$	SR, reg.	QP
Yin et al. (2022)	NN	$Az + Bu$	$z^T Q z + u^T R u$	LSP, SR, ϕ_{V^*}, reg.	LQR
Mondal et al. (2024)	NN	$Az + Bv$	NN	LSP, CP, ϕ_{V^*}	MPPI
LaLQR (Ours)	NN	$\bar{A}_b z + \bar{B}_b v$[1]	$z^T Q z + v^T R v$	LSP, CP	LQR

Table 1: Representative Koopman-based control methods. NN is short for neural network. BF is short for basis function. $\bar{X}$ means the module X is fixed during training and X_t means X is time-variant. reg. means the additional regularizaiton in latent space.

major distinction from prior works (Watter et al., 2015; Li et al., 2020b; Yin et al., 2022; Mondal et al., 2024) is that we retain only LSP and CP as the essential learning principles. Prior work (Ni et al., 2023) has shown that LSP and CP are sufficient to derive the cost-to-go predictor (ϕ_{V^*}), and that the state reconstruction (SR) is unnecessary since our goal is not to accurately predict the next physical state but to plan directly in the latent space. Moreover, we maintain a quadratic cost form, unlike methods that use neural network–based cost models (Mondal et al., 2024), which allows the use of LQR for efficient control. *In summary, our objective is a minimal yet expressive implementation that learns a latent linear system for efficient control with minimal prior knowledge.*

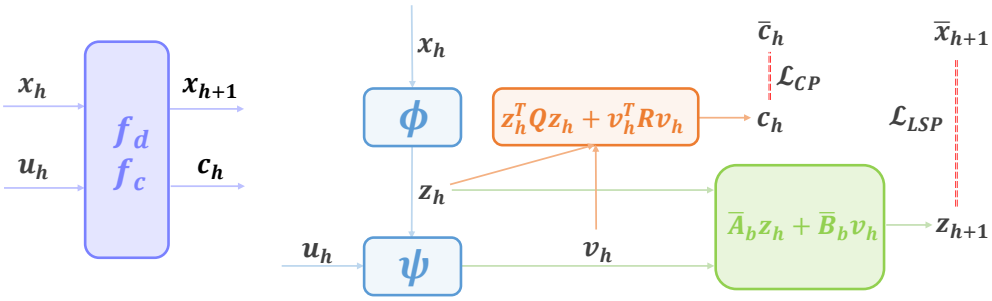
Real-time Predictive Control. For nonlinear systems, beyond the computationally expensive SQP method, several sampling-based predictive control approaches have been proposed for real-time deployment in robotic tasks, such as cross-entropy method (CEM) (de Boer et al., 2005), model predictive path integral (MPPI) (Williams et al., 2015) and predictive sampling (Howell et al., 2022). These methods can achieve speedup with efficient GPU implementations. In comparison, our method requires only a nonlinear embedding function and a linear feedback control law, achieving up to a twenty-fold speedup for complex tasks on CPU-based systems, as shown in Section 4.2. While recent works explore representation learning for control (Zhang et al., 2024; Lutkus et al., 2025; Toso et al., 2025), they primarily address safety or stability in the latent system. In contrast, our paper focuses on leveraging latent structure to specifically improve system efficiency.

3. Latent Linear Quadratic Regulator

We provide a detailed introduction to model predictive control and the Koopman operator in Appendix A. Building on this, we aim to mitigate the high computational burden of the SQP approach in nonlinear MPC, thereby enabling real-time control in robotic tasks. To achieve this, we transform the nonlinear dynamic model into a linear one in a latent space inspired by the Koopman operator. Unlike prior work, we adopt the *Brunovsky canonical form* for latent linear dynamics and *jointly learn* the nonlinear embedding and quadratic cost function, allowing the efficient linear quadratic regulator (LQR) to compute optimal controls. Finally, we analyze the stability of the proposed structure.

3.1. Latent Linear Quadratic Problem

In a standard nonlinear MPC setup, both the dynamic model $x_{h+1} = f_d(x_h, u_h)$ and the cost function $c_h = f_c(x_h, u_h)$ are nonlinear, with state $x_h \in \mathbb{R}^n$ and control $u_h \in \mathbb{R}^m$, as illustrated in



(*a*) Nonlinear Model (*b*) Latent Linear-Quadratic Model

Figure 1: Visualization of different model representations for the same problem. **Left:** nonlinear model with nonlinear dynamic model f_d and cost function f_c. **Right:** equivalent latent linear-quadratic model with linear dynamic model, quadratic cost function and nonlinear embeddings. $\bar{c}$ and $\bar{x}$ represent ground-truth values for given training dataset.

Figure 1(*a*). We instead represent this nonlinear system in a latent linear one, as shown in Figure 1(*b*). Specifically, the state x_h is first lifted to a finite-dimensional latent state $z_h \in \mathbb{R}^N, N > n$ with the embedding function $\phi : \mathbb{R}^n \to \mathbb{R}^N$. The control u_h and latent state z_h are combined to yield a latent control $v_h \in \mathbb{R}^m$ via another embedding $\psi : \mathbb{R}^m \times \mathbb{R}^N \to \mathbb{R}^m$. Furthermore, we assume the latent dynamics are linear: $z_{h+1} = Az_h + Bv_h$, where $A \in \mathbb{R}^{N\times N}$ and $B \in \mathbb{R}^{N\times m}$, and the cost is quadratic: $z_h^T Q z_h + v_h^T R v_h$ with positive semi-definite matrices $Q \in \mathbb{R}^{N\times N}$ and $R \in \mathbb{R}^{m\times m}$. The resulting latent linear–quadratic problem becomes:

$$\min_{v_0,\dots,v_H} \sum_{h=0}^{H} (z_h^\top Q z_h + v_h^\top R v_h), \text{ s.t. } z_{h+1} = Az_h + Bv_h,\ z_0 = \phi(x_0) = \phi(x(t)). \tag{1}$$

This formulation is easier to solve due to its linear dynamics and quadratic cost structure. We first discuss the design of each module before presenting the control solution in Section 3.2.

Embedding Function. The embedding ϕ increases dimensionality ($N > n$) as suggested by the practical Koopman operator approaches and is implemented as a two-layer feedforward network. It needs not be invertible since state reconstruction is unnecessary for planning. The control embedding ψ is implemented as a conditional reversible network, embedding controls as $v = \psi(u, z)$ and recovering u via $u = \psi^{-1}(v, z)$. In practice, ψ is represented with two learnable matrices $M \in \mathbb{R}^{m\times m}$ and $W \in \mathbb{R}^{m\times N}$ that $v = \exp(\frac{M-M^T}{2})(u - Wz)$ and $u = \exp(\frac{M^T-M}{2})v + Wz$.

Dynamic Model. A controllable linear time-invariant (LTI) system with N states and m inputs is constructed in the latent space. Here, we adopt the most fundamental formulation—a parametrization based on controllability indices expressed in the Brunovsky canonical form[2] (Brunovskỳ, 1970)

$$\begin{aligned} \bar{A}_b &= \text{blkdiag}(A_i) \quad A_i = \begin{bmatrix} \mathbf{0}_{(u_i-1)\times 1} & \mathbf{I}_{\mu_i-1} \\ 0 & \mathbf{0}_{1\times(u_i-1)} \end{bmatrix} \in \mathbb{R}^{\mu_i\times\mu_i}, \\ \bar{B}_b &= \text{blkdiag}(B_i) \quad B_i = \begin{bmatrix} \mathbf{0}_{(u_i-1)\times 1} \\ 1 \end{bmatrix} \in \mathbb{R}^{\mu_i\times 1}, \end{aligned} \tag{2}$$

2. We avoid referring to the Brunovsky canonical form as a "Jordan canonical form with zero eigenvalues," since such a description only characterizes $\bar{A}_b$ while neglecting $\bar{B}_b$.

where $\mathbf{I}_j$ denotes identity matrix of size $j \times j$, $\mathbf{0}_{j\times k}$ denotes zero matrix of size $j \times k$, and $\{\mu_i\}_{i=1}^m$ denotes the set of controllability indices satisfying $\sum_{i=1}^m \mu_i = n$. These indices are invariant under any nonsingular state transformation T and input transformation G, i.e., if (A, B) is controllable, then (TAT^{-1}, TBG) has the same controllability indices (Antsaklis and Michel, 2007). In our framework, the linear transformations T and G can be absorbed into the embedding functions ϕ and ψ, allowing us to fix the most compact representation—the Brunovsky canonical form—throughout the learning process. Compared with randomly fixed or freely learnable (A, B) pairs, using the Brunovsky canonical form $(\bar{A}_b, \bar{B}_b)$ offers several advantages: (1) All eigenvalues of $\bar{A}_b$ are zero. In contrast, random A matrices may contain eigenvalues with magnitudes greater than one, leading to numerical instability when simulating the linear dynamics $x^+ = Ax + Bu$ involving matrix powers of A. (2) Solving the LQR problem with $(\bar{A}_b, \bar{B}_b)$ is numerically more efficient (Yang et al., 2025). To the best of the author's knowledge, there was no prior work providing principled guidance on selecting controllability indices of the linear latent system. Nonetheless, when all indices are identical, the resulting deadbeat controller K (which places all eigenvalues of $A - BK$ at zero) is unique (Schlegel, 1982). Consequently, in practice, we set $\mu_i = \lfloor N/m \rfloor, \forall i$ to constrain the solution space and promote training stability. There is another line of work (Lusch et al., 2018; Mondal et al., 2024) using the diagonal structure in the matrix A, but they are either complex-valued or uncontrollable in the latent space. We will analyze the differences from other structures theoretically and empirically in Section 3.4.

Cost Function. $Q \in \mathbb{R}^{N\times N}, R \in \mathbb{R}^{m\times m}$ are positive semi-definite matrices for reasonable control performances. To achieve this, we introduce two auxiliary lower triangular matrices $L_Q \in \mathbb{R}^{N\times N}, L_R \in \mathbb{R}^{m\times m}$ and construct $Q = \mathbf{I}_N + L_Q L_Q^T$ and $R = \mathbf{I}_m + L_R L_R^T$.

3.2. Linear Quadratic Regulator in Latent Space

Given the latent linear system $(\bar{A}_b, \bar{B}_b)$ and quadratic cost (Q, R), the infinite-horizon LQR provides an efficient control law. We extend the horizon H in Equation 1 to ∞ for practical reasons: (1) the practical value H in MPC is usually large enough and its solution is equivalent to the infinite horizon; (2) it is easy to design an efficient control law with infinite horizon (Rawlings et al., 2020). The optimal gain matrix $K \in \mathbb{R}^{m\times N}$ is obtained by solving the discrete-time algebraic Riccati equation. Due to the Brunovsky-form dynamics and the positive semi-definite cost matrices, the Riccati solution is numerically stable (see Section 3.4). In the end, the optimal control in the latent space is calculated by $v^* = -Kz$, and the optimal control in the original action space is recovered by the invertible embedding function as $u^* = \psi^{-1}(v^*, z)$. As the gain matrix K is pre-computed offline, the only online computation per step is calculating $u^* = \psi^{-1}(v^*, z) = \psi^{-1}(-Kz, z) = \psi^{-1}(-K\phi(x), \phi(x))$, given current state x.

3.3. Parameters Identification

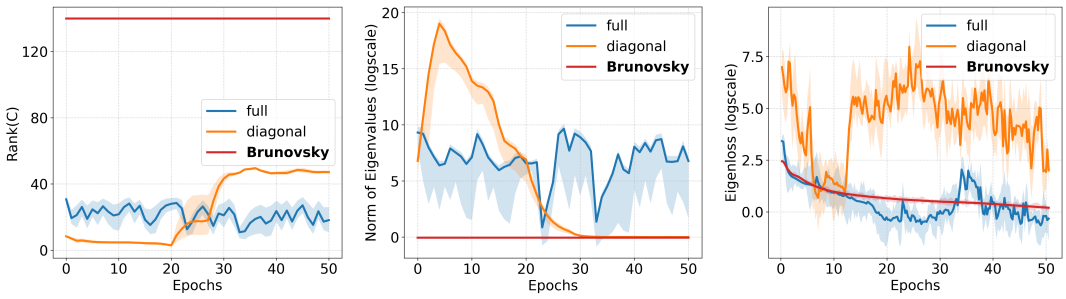
Before applying LaLQR during online deployment, we identify the parameters (ϕ, ψ, Q, R) in a data-driven manner, keeping $(\bar{A}_b, \bar{B}_b)$ fixed. As discussed in Section 2, we retain only two learning objectives: latent state prediction (LSP) and cost prediction (CP), for simplicity and expressiveness. Let $(x_h, u_h, \bar{c}_h, \bar{x}_{h+1})$ denote the ground-truth values directly obtained from the training dataset.

Latent State Prediction (LSP) enforces consistency of the latent dynamics across consecutive steps. Given a sampled transition tuple $(x_h, u_h, \bar{x}_{h+1})$ satisfying the nonlinear dynamic model $\bar{x}_{h+1} = f_d(x_h, u_h)$, we define the latent state prediction loss as $\mathcal{L}_{LSP}(x_h, u_h, \bar{x}_{h+1}; \phi, \psi) =$

$\|\text{sg}(\bar{z}_{h+1}) - (\bar{A}_b z_h + \bar{B}_b v_h)\|_2^2$, where $\bar{z}_{h+1} = \phi(\bar{x}_{h+1}), z_h = \phi(x_h), v_h = \psi(u_h, z_h)$ and sg denotes the stop-gradient operator to prevent representation collapse (no gradients passed to $\bar{z}_{h+1}$). **Cost Prediction (CP)** predicts the cost in the latent space, a crucial but often overlooked component for deriving optimal controllers. Additionally, this objective acts as an implicit regularizer on the latent representation, encouraging it to preserve task-relevant features that are important for control. Given a sampled tuple $(x_h, u_h, \bar{c}_h)$ satisfying the nonlinear cost function $\bar{c}_h = f_c(x_h, u_h)$, we align it with the latent quadratic cost $z_h^T Q z_h + v_h^T R v_h$. Since the cost function f_c can be arbitrarily nonlinear, a direct equivalence with a quadratic cost might be intractable. We assume either that the true form of the cost function is known or that only sampled costs $\bar{c}_h$ are accessible when the cost is computed externally, e.g., in multi-armed bandits (Sutton and Barto, 1998). Instead, we introduce a monotonic function F to match the scales of two costs, and thus the cost prediction loss is defined as $\mathcal{L}_{CP}(x_h, u_h, \bar{c}_h; \phi, \psi, Q, R, F) = \|\bar{c}_h - F(z_h^T Q z_h + v_h^T R v_h)\|_2^2$, where $z_h = \phi(x_h), v_h = \psi(u_h, z_h)$. In practice, we implement F as the Lipschitz monotonic networks (LMN) (Nolte et al., 2022), allowing the monotonic function itself to be also trainable. LMN is an additional component applied on top of the cost function and does not affect the linearity of the latent system.

These two objectives can be seen as different levels of abstraction of the original states x. Prior work (Ni et al., 2023) has shown that LSP and CP are sufficient to derive the abstraction level of the cost-to-go predictor (ϕ_{V^*}). Besides, Ni et al. (2023) observes that the state reconstruction (SR) abstraction is easily influenced by noisy states and outliers, and is unnecessary for planning. The above explains why we employ only the LSP and CP principles for parameter identification. The parameters (ϕ, ψ, Q, R, F) are jointly learned by minimizing a combined objective of LSP and CP: $\min_{(\phi,\psi,Q,R,F)} \frac{1}{|\mathcal{D}|} \sum_{(x_h,u_h,\bar{x}_{h+1},\bar{c}_h)\in\mathcal{D}} \mathcal{L}_{LSP}(x_h, u_h, \bar{x}_{h+1}; \phi, \psi) + \mathcal{L}_{CP}(x_h, u_h, \bar{c}_h; \phi, \psi, Q, R, F)$, where $\mathcal{D}$ denotes the training dataset containing multiple transitions. The approximation errors can be bounded by constraining the magnitudes of all learnable components.

3.4. Stability Analysis



(*a*) Control Rank (*b*) Eigenvalues (*c*) Eigenloss

Figure 2: Stability analysis on the dynamic of LaLQR on `Panda` tasks. Each curve shows the average values over three seeds to follow previous work (Hansen et al., 2022), with shaded regions denoting the standard deviation.

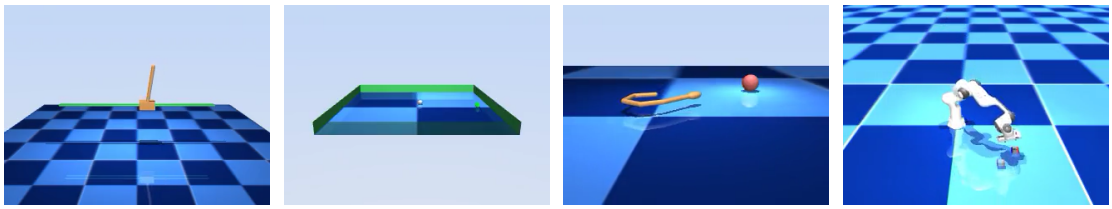
In this section, we evaluate the stability induced by using the Brunovsky canonical form in the latent dynamics. From Equation 1, different feedback laws K arise from different (A, B) structures, and the closed-loop matrix $L = A - BK$ governs latent-space stability in the latent space z. We compare our choice of Brunovsky canonical form $(\bar{A}_b, \bar{B}_b)$ to two additional baselines: "full" means the full matrix A and B are learnable, 'diagonal' means A is a learnable di-

agonal matrix and each dimension of z is independent. All parameters are identified following Section 3.3. For each (A, B, K), we evaluate the controllability via the rank of the controllability matrix $C = [B, AB, \dots A^{n-1}B]$, and stability via the mean absolute value of the eigenvalues of the closed-loop system matrix $L = A - BK$. We also compare the learned latent system with the original nonlinear system at the stable point x_S, u_S. Specifically, the nonlinear system can be linearized around the stable point to yield a closed-loop system: $x_{h+1} = L^o x_h$, where the matrix $L^o = \frac{\partial f_d}{\partial x}\big|_{x=x_S} - \frac{\partial f_d}{\partial u}\big|_{u=u_S} K^o$ with the feedback gain matrix K^o. Meanwhile, for the latent system with closed-loop matrix $L = A - BK$ and Jacobian $J = \frac{\partial \phi}{\partial x}|_{x_S}$, the corresponding latent closed-loop system is $Jx_{h+1} = LJx_h$. The latent closed-loop system $Jx_{h+1} = LJx_h$ accurately represents the linearized closed-loop system $x_{h+1} = L^o x_h$ if, *for every eigenvalue–eigenvector pair* (λ, v) *of* L^o, (λ, Jv) *is also an eigenpair of* L. Accordingly, we define *eigenloss* as $\sum_i \|LJv_i - \lambda_i Jv_i\|_2^2$ computed over all eigenvalue-eigenvector pairs of matrix L^o.

We visualize the above-mentioned metrics in Figure 2. As shown in Figure 2(*a*), our Brunovsky-based model maintains full controllability throughout training, whereas the alternatives exhibit degraded rank in the controllability matrix C with the variations of (A, B). For the full matrix with maximum representational capacity, its effective rank may still be low due to approximation errors during learning. Similarly, the eigenvalue magnitudes (Figure 2(*b*)) remain small for the Brunovsky canonical form, confirming the stable closed-loop autonomous system. Finally, Figure 2(*c*) shows that our model most accurately approximates the original system, while the diagonal structure lacks sufficient expressivity.

4. Experiments

We empirically evaluate LaLQR to verify its ability to learn a latent linear–quadratic structure that enables efficient and stable control across diverse robotic systems.



(*a*) `Cartpole` (*b*) `Particle` (*c*) `Swimmer` (*d*) `Panda`

Figure 3: Visualization of robots used in the experiments, with increasing complexity in action dimension $m \in \{1, 2, 5, 7\}$ and state dimension $n \in \{4, 4, 16, 33\}$.

4.1. Experimental Setup

Task Setup. We use MuJoCo (Todorov et al., 2012) as the simulation platform to evaluate all baselines. Four robotic tasks are selected, `Cartpole`, `Particle`, `Swimmer`, and `Panda`, with increasing complexity in action and state spaces, as illustrated in Figure 3. For `Cartpole`, the controller must balance the pole at the origin. For `Particle` and `Swimmer`, the goal is to reach a specified target position. For `Panda`, the task is to control the robotic arm to pick and place a cube. Detailed task descriptions are provided in Appendix C.1. Training data are collected by rolling out expert controllers (SQP) for 200 episodes, each with a horizon of 500 steps. The use of SQP can

be extended to other efficient—or even imperfect—controllers to facilitate scalable data generation; we provide a detailed analysis of this extension in Section 4.3.

Baselines. We compare our proposed method, **LaLQR**, against the following baselines: **(1) Sequential quadratic programming (SQP)** adopts the simulator's ground-truth nonlinear dynamic model of $x_{h+1} = f_d(x_h, u_h)$ and cost function $f_c(x_h, u_h)$, with the sequential quadratic problem (SQP) method (Tassa et al., 2012) to generate optimal controls. **(2) Cross-entropy method (CEM)** is a sampling-based MPC method to solve Equation 3. **(3) Local LQR (LoLQR)** expands the nonlinear dynamic model at the stable point (x_S, u_S) using a first-order Taylor expansion and applies the resulting linear model globally. Note that identifying an accurate stable point is challenging for some systems, such as `Swimmer`. **(4) Imitation learning (IL)** learns a direct control policy $u_h^* = \pi(x_h)$ using a 3-layer feedforward neural network with a comparable number of parameters to LaLQR for fairness. IL learns to imitate the feedback control law directly; thus, its performance depends heavily on expert data quality and typically generalizes poorly—an issue analyzed further in Section 4.3. Planning hyperparameters for SQP, CEM, and LoLQR, as well as training hyperparameters for IL and LaLQR, are listed in Appendix C.2.

4.2. Main Results

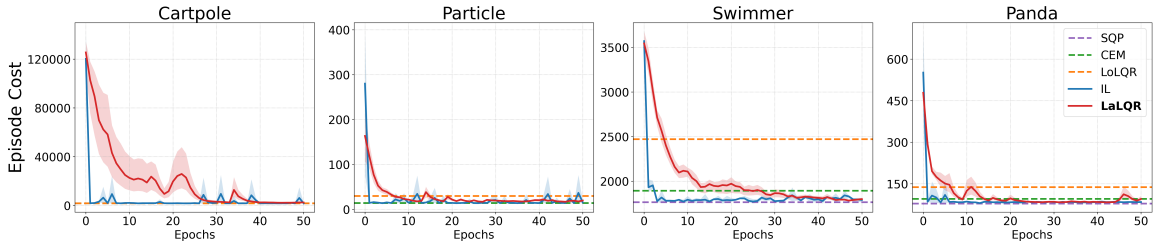


Figure 4: Training curves of all baselines on MuJoCo tasks. Each curve shows the average episode cost over three seeds, with shaded regions denoting the standard deviation.

Tasks	SQP	CEM	LoLQR	IL	LaLQR
`Cartpole`	17.64 ± 7.21	4.64 ± 3.03	0.07 ± 0.04	0.30 ± 0.10	0.29 ± 0.08
`Particle`	20.53 ± 1.41	4.73 ± 2.53	0.09 ± 0.02	0.28 ± 0.04	0.31 ± 0.03
`Swimmer`	47.56 ± 8.86	17.55 ± 4.50	0.08 ± 0.03	4.61 ± 2.85	4.56 ± 1.95
`Panda`	125.49 ± 21.32	37.67 ± 6.78	0.09 ± 0.03	5.76 ± 2.69	5.72 ± 4.61

Table 2: The average time (in milliseconds) for each planning step. All experiments are conducted on an x86_64 CPU with 40 cores, and reported in average and standard deviation over 100 runs.

For each task, IL and LaLQR are trained for 50 epochs using the same expert dataset. At the end of each epoch, both controllers are evaluated over 10 rollouts, and we report the average episode cost (total accumulated cost per episode) in Figure 4. Since SQP, CEM, and LoLQR are non-learning baselines, their results are shown as horizontal reference lines. Additional training losses are provided in Appendix D.1. Figure 4 shows that both IL and LaLQR converge toward the performance of the SQP expert, demonstrating that learning-based feedback controllers can effectively approximate optimal control laws. IL converges faster when trained on perfect expert data, since it imitates only the control outputs rather than jointly learning the embedding and cost functions as in LaLQR. However, as we show in the next section, IL's learning scheme fails to

generalize to unseen conditions. Sampling-based CEM and locally linear LoLQR fail to match SQP on complex tasks such as `Swimmer` and `Panda`. Table 2 summarizes the average computation time per planning step over 100 runs on an x86_64 CPU with 40 cores. Although LoLQR achieves the fastest inference due to its globally linear structure, it performs the worst in control accuracy. CEM improves planning speed over SQP but remains slower than LQR-based methods. Both IL and LaLQR achieve near–real-time inference with only lightweight matrix operations, offering more than an order-of-magnitude speedup over SQP. In summary, LaLQR achieves a favorable trade-off between computational efficiency and control performance.

4.3. Generalization

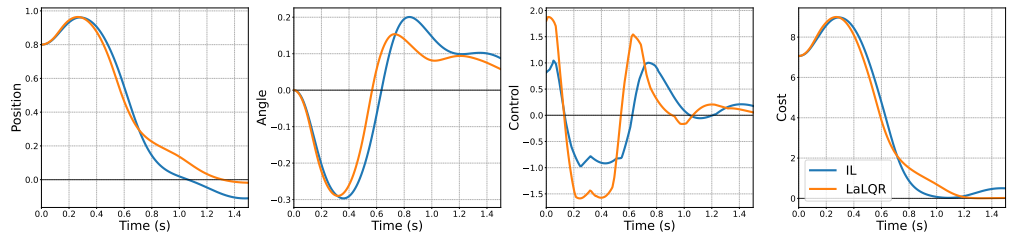


Figure 5: Control process of methods learned from imperfect experts. The x-axis is the real testing time, and the y-axis represents the partial state, control and cost in `Cartpole` task.

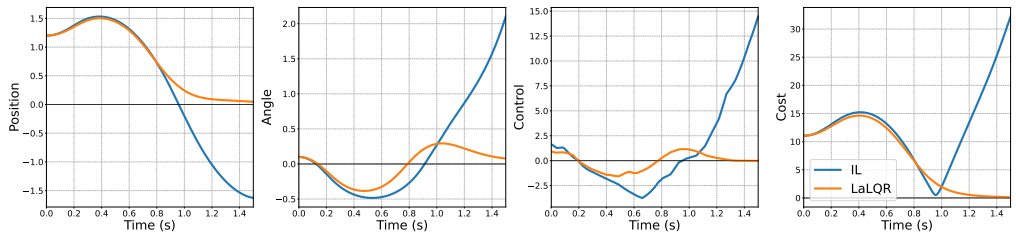


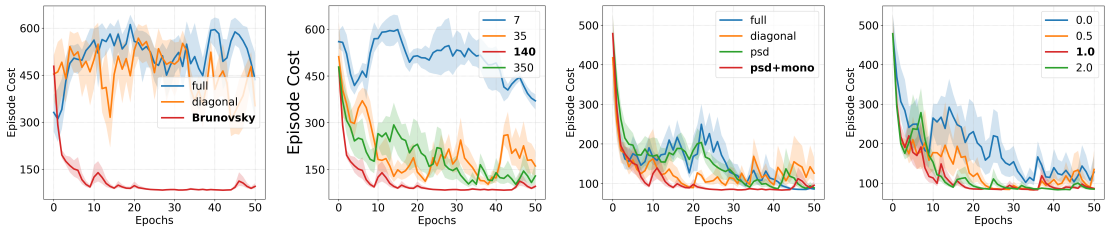
Figure 6: Control process of methods starting from unseen initial states in training. The x-axis is the real testing time, and the y-axis represents the partial state, control and cost in `Cartpole` task.

Learning-based controllers often exhibit poor generalization—they perform well in training-like settings but degrade sharply under distribution shifts. We evaluate IL and LaLQR under two such scenarios: (1) imperfect expert demonstrations and (2) unseen initial states.

Imperfect Expert Demonstrations. In the previous section, both IL and LaLQR were trained on data generated by an optimal expert. Here, we introduce noise to the expert's actions to simulate imperfect supervision: with probability 0.5, a uniform noise sampled from $[-1, 1]$ is added to the expert control at each step. All other setups remain unchanged. As shown in Figure 5, IL directly inherits the expert's suboptimal behavior, resulting in higher final costs. In contrast, LaLQR generalizes beyond the noisy demonstrations and still achieves stable control with near-zero final cost.

Unseen Initial States. We further test generalization by evaluating from unseen initial conditions. During training on the `Cartpole` task, initial positions are sampled uniformly from $[-1, 1]$ with a fixed angle of 0. During testing, we set the initial position to 1.2 and the angle to 0.1. As shown in Figure 6, IL fails to stabilize the system from this unseen state, while LaLQR maintains robust performance. These experiments demonstrate that learning the full MPC structure—rather than imitating actions alone—significantly enhances generalization.

4.4. Ablation Study



(*a*) dynamic model (*b*) latent dim (*c*) cost function (*d*) cost coefficient

Figure 7: Ablation study on key design choices of LaLQR on `Panda` tasks. Each curve shows the average episode return over three seeds, with shaded regions denoting the standard deviation.

We conduct an ablation study on the most challenging task, `Panda`, to analyze the effect of key design choices. First, we investigate the structure of (A, B) in the latent dynamics. As shown in Figure 7(*a*), the Brunovsky canonical form $(\bar{A}_b, \bar{B}_b)$ yields both higher control performance and more stable training than the diagonal ("diagonal") or unconstrained ("full") variants, aligning with our stability analysis in Section 3.4. Figure 7(*b*) studies the lifted dimension N of the Koopman operator. Smaller values (e.g., 7 or 35) lack sufficient expressiveness to model nonlinear dynamics, while overly large values (e.g., 350) degrade control efficiency due to large matrix computations in LQR. We thus set $N = 20m$ in all experiments. Regarding the cost structure (Figure 7(*c*)), both positive semi-definite (PSD) and unconstrained (Q, R) matrices achieve good performance, whereas real-diagonal (Q, R) fail to capture the full eigenstructure of the nonlinear system. The monotonic function F introduced in Section 3.3 accelerates learning by absorbing the nonlinearity between true and predicted costs. Finally, Figure 7(*d*) shows the effect of the cost prediction loss coefficient, which acts as a regularization strength to prevent latent space collapse. Performance remains robust across a wide range of values, and we fix this coefficient to 1 throughout our experiments.

5. Conclusion

In conclusion, the proposed latent linear quadratic regulator (LaLQR) method effectively addresses the computational challenges of model predictive control (MPC) in systems with nonlinear dynamics. Compared to other efficient solutions, LaLQR presents better control performances and enhanced generalization capabilities. This novel approach holds promise for advancing the practical application of MPC in embedded systems, which is crucial for real-time control in complex robotic environments.

For future work, incorporating the embedding's estimation error with the latent LQR robustly is essential for obtaining closed-loop certificates in the original state space. Additionally, our method can be compared with model-based reinforcement learning algorithms (Hansen et al., 2022), which simultaneously learn models and perform optimal control. To enhance real-world transfer, learnable modules can be combined with domain randomization (Peng et al., 2018) techniques to improve the robustness of the MPC framework. Finally, instead of setting the controllability indices of the linear latent system heuristically, a more principled method should be explored based on the geometry of the original nonlinear system, e.g., the nonlinear controllability indices defined through the differential-geometric/Lie-algebraic framework (Isidori, 1985).

Acknowledgments

This research received funding from the European Union's Horizon 2020 research and innovation program under the Marie Skłodowska-Curie grant agreement No. 953348 (ELO-X). This work was part of BrainLinks-BrainTools, which was funded by the Federal Ministry of Economics, Science and Arts of Baden-Württemberg within the sustainability program for projects of the excellence initiative II. This work was supported as a part of NCCR Automation, a National Centre of Competence in Research, funded by the Swiss National Science Foundation (grant number 51NF40_225155). The authors thank Jasper Hoffman for the inspiring discussions and proofreading.

References

Brian DO Anderson and John B. Moore. *Optimal Control: Linear Quadratic Methods*. Courier Corporation, 2007.

Panos J. Antsaklis and Anthony N. Michel. *A Linear Systems Primer*. Springer Science & Business Media, 2007.

Anthony Brohan, Noah Brown, Justice Carbajal, Yevgen Chebotar, Joseph Dabis, Chelsea Finn, Keerthana Gopalakrishnan, Karol Hausman, Alex Herzog, Jasmine Hsu, Julian Ibarz, Brian Ichter, Alex Irpan, Tomas Jackson, Sally Jesmonth, Nikhil J. Joshi, Ryan Julian, Dmitry Kalashnikov, Yuheng Kuang, Isabel Leal, Kuang-Huei Lee, Sergey Levine, Yao Lu, Utsav Malla, Deeksha Manjunath, Igor Mordatch, Ofir Nachum, Carolina Parada, Jodilyn Peralta, Emily Perez, Karl Pertsch, Jornell Quiambao, Kanishka Rao, Michael Ryoo, Grecia Salazar, Pannag Sanketi, Kevin Sayed, Jaspiar Singh, Sumedh Sontakke, Austin Stone, Clayton Tan, Huong Tran, Vincent Vanhoucke, Steve Vega, Quan Vuong, Fei Xia, Ted Xiao, Peng Xu, Sichun Xu, Tianhe Yu, and Brianna Zitkovich. Rt-1: Robotics transformer for real-world control at scale, December 2022.

Daniel Bruder, Brent Gillespie, C. David Remy, and Ram Vasudevan. Modeling and control of soft robots using the koopman operator and model predictive control. In Antonio Bicchi, Hadas Kress-Gazit, and Seth Hutchinson, editors, *Robotics: Science and Systems XV, University of Freiburg, Freiburg Im Breisgau, Germany, June 22-26, 2019*, 2019. doi: 10.15607/RSS.2019.XV.060.

Pavol Brunovskỳ. A classification of linear controllable systems. *Kybernetika*, 6(3):173–188, 1970.

Pieter-Tjerk de Boer, Dirk P. Kroese, Shie Mannor, and Reuven Y. Rubinstein. A tutorial on the cross-entropy method. *Annals of Operations Research*, 134(1):19–67, February 2005. ISSN 1572-9338. doi: 10.1007/s10479-005-5724-z.

Jared Di Carlo, Patrick M. Wensing, Benjamin Katz, Gerardo Bledt, and Sangbae Kim. Dynamic locomotion in the mit cheetah 3 through convex model-predictive control. In *2018 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 1–9, Madrid, October 2018. IEEE. ISBN 978-1-5386-8094-0. doi: 10.1109/IROS.2018.8594448.

Ruben Grandia, Fabian Jenelten, Shaohui Yang, Farbod Farshidian, and Marco Hutter. Perceptive locomotion through nonlinear model-predictive control. *IEEE Trans. Robotics*, 39(5):3402–3421, 2023. doi: 10.1109/TRO.2023.3275384.

Nicklas A. Hansen, Hao Su, and Xiaolong Wang. Temporal difference learning for model predictive control. In *Proceedings of the 39th International Conference on Machine Learning*, pages 8387–8406. PMLR, June 2022.

Taylor Howell, Nimrod Gileadi, Saran Tunyasuvunakool, Kevin Zakka, Tom Erez, and Yuval Tassa. Predictive sampling: Real-time behaviour synthesis with mujoco, December 2022.

Alberto Isidori. *Nonlinear control systems: an introduction*. Springer, 1985.

Sotaro Katayama, Masaki Murooka, and Yuichi Tazaki. Model predictive control of legged and humanoid robots: Models and algorithms. *Adv. Robotics*, 37(5):298–315, 2023. doi: 10.1080/01691864.2023.2168134.

Milan Korda and Igor Mezić. Linear predictors for nonlinear dynamical systems: Koopman operator meets model predictive control. *Automatica*, 93:149–160, 2018.

Yunzhu Li, Hao He, Jiajun Wu, Dina Katabi, and Antonio Torralba. Learning compositional koopman operators for model-based control. In *8th International Conference on Learning Representations, ICLR 2020, Addis Ababa, Ethiopia, April 26-30, 2020*. OpenReview.net, 2020a.

Yunzhu Li, Antonio Torralba, Anima Anandkumar, Dieter Fox, and Animesh Garg. Causal discovery in physical systems from videos. In *Advances in Neural Information Processing Systems*, volume 33, pages 9180–9192. Curran Associates, Inc., 2020b.

Ilya Loshchilov and Frank Hutter. Decoupled weight decay regularization. In *International Conference on Learning Representations*, September 2018.

Bethany Lusch, J. Nathan Kutz, and Steven L. Brunton. Deep learning for universal linear embeddings of nonlinear dynamics. *Nature Communications*, 9(1):4950, November 2018. ISSN 2041-1723. doi: 10.1038/s41467-018-07210-0.

Paul Lutkus, Kaiyuan Wang, Lars Lindemann, and Stephen Tu. Latent representations for control design with provable stability and safety guarantees. In *64th IEEE Conference on Decision and Control, CDC 2025, Rio de Janeiro, Brazil, December 9-12, 2025*, pages 2937–2944. IEEE, 2025. doi: 10.1109/CDC57313.2025.11312739.

Giorgos Mamakoukas, Maria Castano, Xiaobo Tan, and Todd Murphey. Local koopman operators for data-driven control of robotic systems. In *Robotics: Science and Systems XV*. Robotics: Science and Systems Foundation, June 2019. ISBN 978-0-9923747-5-4. doi: 10.15607/RSS.2019.XV.054.

Diganta Misra. Mish: A self regularized non-monotonic activation function. In *31st British Machine Vision Conference 2020, BMVC 2020, Virtual Event, UK, September 7-10, 2020*. BMVA Press, 2020.

Arnab Kumar Mondal, Siba Smarak Panigrahi, Sai Rajeswar, Kaleem Siddiqi, and Siamak Ravanbakhsh. Efficient dynamics modeling in interactive environments with koopman theory. In *The Twelfth International Conference on Learning Representations*, 2024.

Tianwei Ni, Benjamin Eysenbach, Erfan SeyedSalehi, Michel Ma, Clement Gehring, Aditya Mahajan, and Pierre-Luc Bacon. Bridging state and history representations: Understanding self-predictive rl. In *The Twelfth International Conference on Learning Representations*, October 2023.

Jorge Nocedal and Stephen J. Wright. Sequential quadratic programming. In *Numerical Optimization*, pages 526–573. Springer-Verlag, New York, 1999. ISBN 978-0-387-98793-4.

Niklas Nolte, Ouail Kitouni, and Mike Williams. Expressive monotonic neural networks. In *The Eleventh International Conference on Learning Representations*, September 2022.

Xue Bin Peng, Marcin Andrychowicz, Wojciech Zaremba, and Pieter Abbeel. Sim-to-real transfer of robotic control with dynamics randomization. In *2018 IEEE International Conference on Robotics and Automation (ICRA)*, pages 3803–3810, May 2018. doi: 10.1109/ICRA.2018.8460528.

James Blake Rawlings, David Q. Mayne, and Moritz Diehl. *Model Predictive Control: Theory, Computation, and Design*, volume 2. Nob Hill Publishing Madison, WI, 2020.

Matthew Retchin, Brandon Amos, Steven Brunton, and Shuran Song. Koopman constrained policy optimization: A koopman operator theoretic method for differentiable optimal control in robotics. In *ICML 2023 Workshop on Differentiable Almost Everything: Differentiable Relaxations, Algorithms, Operators, and Simulators*, September 2023.

M Schlegel. Parameterization of the class of deadbeat controllers. *IEEE Transactions on Automatic Control*, 27(3):727–729, 1982.

Yunlong Song and Davide Scaramuzza. Learning high-level policies for model predictive control. In *2020 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 7629–7636, October 2020. doi: 10.1109/IROS45743.2020.9340823.

Richard S. Sutton and Andrew G. Barto. *Reinforcement Learning - an Introduction*. Adaptive Computation and Machine Learning. MIT Press, 1998. ISBN 978-0-262-19398-6.

Yuval Tassa, Tom Erez, and Emanuel Todorov. Synthesis and stabilization of complex behaviors through online trajectory optimization. In *2012 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 4906–4913, October 2012. doi: 10.1109/IROS.2012.6386025.

Emanuel Todorov, Tom Erez, and Yuval Tassa. Mujoco: A physics engine for model-based control. In *2012 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 5026–5033, Vilamoura-Algarve, Portugal, October 2012. IEEE. ISBN 978-1-4673-1736-8 978-1-4673-1737-5 978-1-4673-1735-1. doi: 10.1109/IROS.2012.6386109.

Leonardo F. Toso, Lintao Ye, and James Anderson. Learning stabilizing policies via an unstable subspace representation. In *64th IEEE Conference on Decision and Control, CDC 2025, Rio de Janeiro, Brazil, December 9-12, 2025*, pages 7543–7550. IEEE, 2025. doi: 10.1109/CDC57313.2025.11312355.

Yuh-Shyang Wang, Nikolai Matni, and John C. Doyle. Localized lqr optimal control. In *53rd IEEE Conference on Decision and Control*, pages 1661–1668. IEEE, 2014.

Manuel Watter, Jost Springenberg, Joschka Boedecker, and Martin Riedmiller. Embed to control: A locally linear latent dynamics model for control from raw images. In *Advances in Neural Information Processing Systems*, volume 28. Curran Associates, Inc., 2015.

Matthew O. Williams, Ioannis G. Kevrekidis, and Clarence W. Rowley. A data–driven approximation of the koopman operator: Extending dynamic mode decomposition. *Journal of Nonlinear Science*, 25(6):1307–1346, December 2015. ISSN 0938-8974, 1432-1467. doi: 10.1007/s00332-015-9258-5.

Shaohui Yang, Toshiyuki Ohtsuka, and Colin N. Jones. Brunovsky riccati recursion for linear model predictive control. In *2025 American Control Conference, ACC 2025, Denver, CO, USA, July 8-10, 2025*, pages 3367–3372. IEEE, 2025. doi: 10.23919/ACC63710.2025.11107756.

Hang Yin, Michael C. Welle, and Danica Kragic. Embedding koopman optimal control in robot policy learning. In *2022 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 13392–13399, Kyoto, Japan, October 2022. IEEE. ISBN 978-1-66547-927-1. doi: 10.1109/IROS47612.2022.9981540.

Thomas T. C. K. Zhang, Leonardo Felipe Toso, James Anderson, and Nikolai Matni. Sample-efficient linear representation learning from non-iid non-isotropic data. In *The Twelfth International Conference on Learning Representations, ICLR 2024, Vienna, Austria, May 7-11, 2024*. OpenReview.net, 2024.

Appendix A. Background

A.1. Model Predictive Control

Given the current state $x(t) \in \mathbb{R}^n$ at time t and a planning horizon H, discrete-time MPC consists of a dynamic model $x_{h+1} = f_d(x_h, u_h)$, a stage-wise cost function $f_c(x_h, u_h)$, and an optimization algorithm that computes the optimal control sequences $\{u_0^*, u_1^*, \dots u_H^*\}$, where $u_h \in \mathbb{R}^m$. The first control input is applied, and the optimization is re-solved at the next timestep using the updated measurement. The optimization problem is formulated as follows:

$$\min_{u_0,\dots,u_H} \sum_{h=0}^{H} f_c(x_h, u_h), \quad \text{s.t. } x_{h+1} = f_d(x_h, u_h),\ x_0 = x(t) \tag{3}$$

In general, both the dynamics f_d and the cost function f_c are nonlinear. Sequential quadratic programming (SQP) is often employed to solve Equation 3, but it incurs significant computational cost due to repeated solution of QPs and gradient evaluations. The terminal cost function is omitted here, as it is often unavailable in robotic tasks (Howell et al., 2022).

A.2. Linear Predictors for Nonlinear Controlled Systems via the Koopman Operator

Lifting the state space to a higher dimension plays a key role in obtaining predictions of a nonlinear dynamical system $x^+ = f_d(x, u)$ as the output of a linear one. For uncontrolled dynamical systems, this idea can be justified with the Koopman operator theory. For controlled systems, the justification requires the extension of the state space, the new definition of observables, and their residing Hilbert space. In this paper, we follow a both rigorous and practical way of generalizing the Koopman operator $\mathcal{K}$ to controlled systems as in (Korda and Mezić, 2018) [3]:

$$\mathcal{K}\begin{bmatrix}\phi(x)\\ \boldsymbol{u}(0)\end{bmatrix} = \begin{bmatrix}\phi(f(x, \boldsymbol{u}(0)))\\ (\mathcal{S}\boldsymbol{u})(0)\end{bmatrix} = \begin{bmatrix}\phi(f_d(x, \boldsymbol{u}(0)))\\ \boldsymbol{u}(1)\end{bmatrix}, \boldsymbol{u}(0) = u. \tag{4}$$

This leads to the following (approximate) linear evolution in the lifted state space $z = \phi(x) \in \mathbb{R}^N$ while the control space remains unchanged: $z^+ = \phi(x^+) = A\phi(x) + B\boldsymbol{u}(0) = Az + Bu$. In practice, a finite-dimensional lifted state z is used to approximate the behaviour of the original nonlinear system (Lusch et al., 2018; Retchin et al., 2023; Mondal et al., 2024).

Appendix B. Additional Algorithm Details

The pseudocode of the algorithm can be found in Algorithm 1.

Appendix C. Additional Experimental Setups

C.1. Robotic Tasks on MuJoCo

We select 4 robotic tasks `Cartpole`, `Particle`, `Swimmer` and `Panda`, with increasing complexity in state and action spaces as illustrated in Figure 3.

3. $\boldsymbol{u} = \{u_i\}_{i=0}^{\infty}$ denotes the control sequence till infinity, $\mathcal{S}$ is the left shift operator.

Algorithm 1 LaLQR Parameter Identification

Input: Nonlinear MPC's dynamic model $f_d(x_h, u_h)$, cost function $f_c(x_h, u_h)$ and optimization algorithm to generate optimal control $u_h = \text{MPC}(x_h)$, initial state distribution $\Delta_{\mathcal{X}}$
Output: Embedding function ϕ, ψ, quadratic cost parameters Q, R
Initialize dataset $\mathcal{D}_{\text{model}} = \{\}$
for episode number $e = 1, 2, ..., E$
  Sample an initial state $x_0 \sim \Delta_{\mathcal{X}}$
  for step $h = 0, 2, ..., H$
    Generate optimal control u_h at state x_h with control rule $u_h = \text{MPC}(x_h)$
    Rollout for next state $x_{h+1} = f_d(x_h, u_h)$ and cost $c_h = f_c(x_h, u_h)$
    Store in the model dataset $\mathcal{D} \leftarrow \mathcal{D} \cup \{(x_h, u_h, c_h, x_{h+1})\}$
  end for
end for
for iteration $i = 1, 2, ..., I$
  Sample B samples $(x_h^b, u_h^b, c_h^b, x_{h+1}^b)_{b=1}^B$ from the dataset $\mathcal{D}$
  Update parameters $p = (\phi, \psi, Q, R, F) = p - \lambda \nabla_p \sum_{b=1}^B \mathcal{L}_{\text{LSP}}(x_h^b, u_h^b, x_{h+1}^b) + \mathcal{L}_{\text{CP}}(x_h^b, u_h^b, c_h^b)$, where λ is the learning rate
end for

- `Cartpole` has an unactuated pole on top of a moving cart. In this task, the controller should balance the pole in the origin position. The state (4-dim) consists of the horizontal position and the angle from the vertical of the pole and their velocities. The control (1-dim) is the horizontal force on the cart. The cost is calculated as $c(x_t, u_t) = 0.1 \times c_{\text{velocity}}(x_t, u_t) + 0.1 \times c_{\text{control}}(x_t, u_t) + 10.0 \times c_{\text{centered}}(x_t, u_t) + 10.0 \times c_{\text{vertical}}(x_t, u_t)$. $c_{\text{velocity}}(x_t, u_t)$ is the absolute speed in horizontal direction. $c_{\text{control}}(x_t, u_t)$ is the norm of controls. $c_{\text{centered}}(x_t, u_t)$ is the absolute value of the position. $c_{\text{vertical}}(x_t, u_t)$ is the absolute value of the angle. The control frequency is 100 Hz, and each episode runs for a horizon of 500 steps.

- `Particle` is a lightweight 2-D point-mass robot controlled by applying forces along the x and y axes. The robot's goal is to reach a designated target position. The state (4-dim) includes its 2-D position and velocity, while the control (2-dim) represents the planar forces applied. The cost function is defined as $c(x_t, u_t) = 5.0 \times c_{\text{distance}}(x_t, u_t) + 0.1 \times c_{\text{velocity}}(x_t, u_t) + 0.1 \times c_{\text{control}}(x_t, u_t)$, where c_{distance} measures the Euclidean distance to the target, c_{velocity} penalizes high velocities, and c_{control} penalizes control effort. The control frequency is 100 Hz, and each episode runs for a horizon of 500 steps.

- `Swimmer` is a snake-liked robot with 5 controllable actuators. In this task, the robot is driven to reach the target position. The state (16-dim) consists of the 3-D position of the robot, the positions of 5 actuators and their velocities. The control (5-dim) is the force of 5 actuators. The cost is calculated as $c(x_t, u_t) = 10.0 \times c_{\text{distance}}(x_t, u_t) + 0.1 \times c_{\text{control}}(x_t, u_t)$. $c_{\text{distance}}(x_t, u_t)$ is the distance from the target position. $c_{\text{control}}(x_t, u_t)$ is the norm of controls. The control frequency is 100 Hz, and each episode runs for a horizon of 500 steps.

- `Panda` is a 7-DoF robotic manipulator based on the Franka Emika Panda arm. In this task, the robot must pick up a cube and place it at a fixed target location $[1.0, 1.0, 0.05]$. The state (33-dim) includes the cube's pose, joint positions and velocities and the gripper status. The control

(7-dim) corresponds to the end-effector's pose (6-dim) and gripper action (1-dim). The cost function is defined as $c(x_t, u_t) = 1.0 \times c_{\text{reach}}(x_t, u_t) + 0.1 \times c_{\text{bring}}(x_t, u_t) + 0.1 \times c_{\text{control}}(x_t, u_t)$, where c_{reach} measures the distance between the end-effector and the cube, c_{bring} penalizes the deviation of the cube from the target, and c_{control} regularizes torque magnitudes. The control frequency is 111 Hz, and the horizon is 500 steps.

C.2. Hyperparameters of Baselines

We list the hyperparameters for all baselines in the paper.

SQP SQP requires prior knowledge of the nonlinear dynamic model f_d and cost function f_c, which can be directly achieved from the MuJoCo platform. The optimization is calculated based on iLQG (Tassa et al., 2012), essentially a Gauss-Newton method utilizing first- and second-order derivative information. The detailed implementation follows the source code from Howell et al. (2022).

Parameter	Value
Planning horizon	100
Minimum line search step	1.0×10^{-3}
Finite difference tolerance	1.0×10^{-6}
Finite difference mode	0 (0: one-sided, 1: centered)
Minimum regularization value	1.0×10^{-6}
Maximum regularization value	1.0×10^{6}
Regularization type	0 (0: control, 1: feedback, 2: value, 3: none)
Maximum regularization iterations	5

Table 3: Planning and Optimization Hyperparameters of SQP

CEM Similarly to SQP, CEM also requires prior knowledge of the nonlinear dynamic model f_d and cost function f_c, which can be directly achieved from the MuJoCo platform. The detailed implementation follows the source code from Howell et al. (2022).

Parameter	Value
Planning horizon	100
Initial sampling variance	0.1
Minimum variance	0.01
Number of sampled trajectories	20
Number of elite samples	10

Table 4: Planning and Optimization Hyperparameters of CEM

LoLQR LoLQR expands the nonlinear dynamic model at the stable point (x_S, u_S) with Taylor expansion, and uses the ground truth cost function f_c to calculate control laws. The stable point can be achieved by the final solution of the SQP controller.

IL IL directly learns a control policy $u_h = \pi(x_h)$ by imitating the output of SQP with a neural network. The network structure is set as a 3-layer feed-forward neural network with $[512, N]$ hidden units, Mish (Misra, 2020) activation function. The model is trained for 50 epochs with 128 batch

size, and the optimizer is AdamW (Loshchilov and Hutter, 2018) with a learning rate of 0.001. The dimension N of the latent space for each task can be referred to Table 5.

LaLQR (Ours) Most hyperparameters and training datasets of LaLQR are set equally as IL for a fair comparison. The embedding function ϕ is implemented as a 2-layer neural network, with a one-layer feed-forward neural network with $[512]$ hidden units, Mish (Misra, 2020) activation function. The output dimension is N so that the gain matrix K has the size $N \times m$, which has similar parameters as the last layer of the policy network in IL. The model is updated for 50 epochs with 128 batch size, and the optimizer is AdamW (Loshchilov and Hutter, 2018) with a learning rate of 0.001. The dimension N of the latent space for each task can be referred to Table 5. The latent space dimension is empirically set to $20m$ for m control inputs to balance representational capacity and learning difficulty, as demonstrated in Section 4.4. This hyperparameter warrants further investigation and tuning in future work.

Parameters	`Cartpole`	`Particle`	`Swimmer`	`Panda`
latent dimension N	20	40	100	140

Table 5: Dimension of latent space of IL and LaLQR for all environments.

Appendix D. Additional Experimental Results

D.1. Additional Training Results

We plot the training curves of LaLQR on all tasks in Figure 8. In general, this is not a difficult task for our proposed method, as all losses gradually decrease to 0 in all tasks.

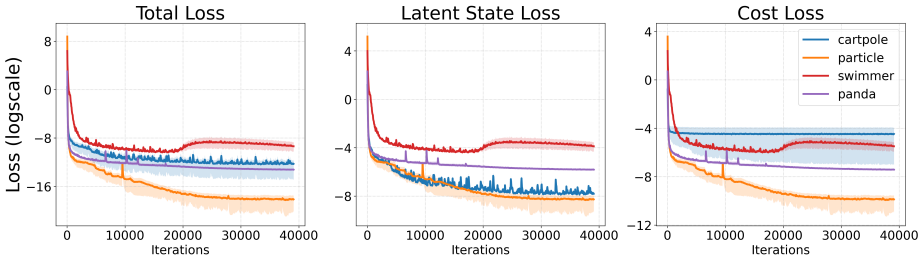


Figure 8: Training curves of LaLQR on all tasks. The a-xis is the training iterations, and the y-axis are training objectives of total loss, latent state loss and cost loss respectively. All results are plotted with the average and standard deviation shading over 3 random seeds