

On-policy Deep Reinforcement Learning Assisted Koopman Bilinear Model Predictive Control for Unknown Dynamical Systems

Ketong Zheng^{1*}, Peng Huang², Jonathan Casas¹ and Gerhard Fettweis^{1,2}

¹Vodafone Chair Mobile Communications Systems, Technische Universität Dresden, 01069, Dresden, Germany

²Barkhausen Institut, 01062, Dresden, Germany

Contacts: {ketong.zheng, peng.huang, jonathan.casas, gerhard.fettweis}@tu-dresden.de (*Corresponding author)

Abstract: Data-driven Koopman operator approximation has gained interest recently for its ability to embed nonlinear systems into a lifted linear state space using only measurements. When control inputs are included, however, the lifted dynamics render a bilinear form, which poses challenges for controller synthesis, such as Model Predictive Control (MPC). This paper proposes an on-policy actor-critic Deep Reinforcement Learning (DRL) framework that simultaneously learns the Koopman bilinear dynamics and a MPC neural cost map. Instead of directly generating control actions, the actor network takes the Koopman-lifted states and produces MPC weight matrices for each prediction step. These state-dependent weight matrices serve as high-level guidance for the control objective, allowing the low-level MPC to run under very short prediction horizon while maintaining stability and enforcing safety constraints. Simulations carried out with the OpenAI Gym library demonstrate that, without requiring explicit knowledge of the dynamics, the proposed Actor-Critic Koopman MPC (ACKMPC) achieves control accuracy and disturbance robustness on par with a model-based ACMPC, and outperforms a pure DRL-learned policy using baseline Proximal Policy Optimization (PPO). It also exceeds standard Koopman MPC (KMPC) in both robustness and computational efficiency.

Keywords: Data-driven control, Koopman operator, Deep reinforcement learning, Model predictive control

1. INTRODUCTION

Data-driven system identification and control has emerged as a powerful methodology for modeling dynamical systems and controller synthesis solely from measurement data [1]. Within this domain, the Koopman operator has attracted significant interest for its ability to represent nonlinear systems linearly in an infinite-dimensional space. To make this infinite dimensionality applicable in practice, extended dynamic mode decomposition (EDMD) is typically applied as a finite-dimensional approximation of the Koopman operator [2]. This involves identifying a set of Koopman observables that approximate a Koopman invariant subspace and project the original system dynamics into this Koopman lifted system. While the Koopman approximation is initially formulated for unactuated systems, many researchers have made the extension to actuated systems with lifted linear control dynamics [3], [4], [5]. However, recent studies demonstrate that controlled dynamics often yield a bilinear form in the Koopman lifted space [6], [7], [8]. This representation enhances long-term prediction accuracy but complicates controller design.

To address this challenge, classical reactive control approaches including optimal control, robust control, and feedback linearization have been developed specifically for Koopman bilinear systems [8], [9], [10]. Yet ac-

curately quantifying the modeling error from the finite-dimensional approximation remains difficult, and this uncertainty can propagate through the control law, causing performance degradation under disturbances and model mismatch. Model Predictive Control (MPC) offers a promising alternative by solving a receding horizon optimization problem with constraint satisfaction, which has been applied to Koopman models in many works [11], [12], [13]. However, Koopman MPC (KMPC) for bilinear systems faces inherent nonconvexity and high dimensionality, resulting in substantial computational cost.

Romero *et al.* recently introduced actor-critic MPC (ACMPC), in which a learned actor network outputs state-dependent weight matrices [14] to parameterize the MPC cost. This actor network is essentially a neural cost map, providing high-level guidance for the optimization task, and the low-level MPC layer ensures constraint satisfaction. According to their results, ACMPC outperforms standard DRL with baseline Proximal Policy Optimization (PPO) where the actor network directly outputs the control action. Nevertheless, like conventional MPC, ACMPC also assumes perfect knowledge of a first-principles model.

To bridge the gap, this paper proposes actor-critic Koopman MPC (ACKMPC), which is an on-policy DRL framework that jointly learns the Koopman bilinear surrogate model and the MPC neural cost map from data. In the offline phase, the deep neural network (DNN) parameterized Koopman observables are trained using initial rollout measurements. During the online phase, the Koopman bilinear dynamics and the neural cost map for Koopman MPC (KMPC) are jointly learned under an on-policy actor-critic framework. The output of the neural cost map is passed to a differential MPC solver [15],

The authors acknowledge the financial support by the Federal Ministry of Education and Research of Germany in the programme of "Souverän. Digital. Vernetzt.". Joint project 6G-life, project identification number: 16KISK001K; and the German Research Foundation (DFG, Deutsche Forschungsgemeinschaft) as part of Germany's Excellence Strategy - EXC 2050/1 - Project ID 390696704 - Cluster of Excellence "Centre for Tactile Internet with Human-in-the-Loop" (CeTI) of Technische Universität Dresden. This work is partially financed on the basis of the budget passed by the Saxon State Parliament.

where the gradient of the MPC output is tractable. To the best of our knowledge, this is the first paper that proposes the end-to-end data-driven learning framework of the Koopman bilinear model and controller design, where the model continuously adapts to trajectories generated by the current policy, and the controller optimizes specifically for the latest dynamics representation. Meanwhile, with the learned neural cost map of the KMPC, the prediction horizon can be shortened to very few steps, enabling real-time execution compared to standard KMPC. The performance is evaluated with the OpenAI Gym library, where the proposed ACKMPC achieves dramatically reduced solve times and superior disturbance robustness relative to the KMPC, while also outperforming the baseline PPO under perturbations, thanks to its predictive capability and constraint handling.

The remainder of the paper is organized as follows: Section 2 reviews the bilinear control form of the Koopman operator, the data-driven Koopman approximation techniques, and DRL fundamentals. Section 3 introduces the ACMPC framework and details the proposed ACKMPC methodology. System settings and simulation results are presented in Section 4. The outlines and future work are concluded in Section 5.

2. PRELIMINARIES

2.1 Koopman Bilinear Representation

Consider a general nonlinear control-affine system

$$\dot{\mathbf{x}} = \mathbf{f}(\mathbf{x}) + \sum_{i=1}^m \mathbf{b}_i(\mathbf{x}) \cdot u_i, \quad (1)$$

where $\mathbf{x} \in \mathcal{X} \subset \mathbb{R}^n$, $\mathbf{u} \in \mathcal{U} \subset \mathbb{R}^m$, u_i denotes the i^{th} element of \mathbf{u} . The Euclidean space is represented by \mathbb{R} , and $\mathbf{f}, \mathbf{b}_{i=1\dots m} : \mathcal{X} \mapsto \mathbb{R}^n$ are the drifted and control vector fields. The Koopman operator is initially defined for the unactuated term $\dot{\mathbf{x}} = \mathbf{f}(\mathbf{x})$, corresponding to the flow map $S_{\mathbf{f}}^t : [0, \infty) \times \mathcal{X} \mapsto \mathcal{X}$ given by $S_{\mathbf{f}}^t(\mathbf{x}_0) := \mathbf{x}_0 + \int_{t_0}^{t_0+t} \mathbf{f}(\mathbf{x}(\tau)) d\tau$, where $\mathbf{x}(t_0) = \mathbf{x}_0$. Let \mathcal{L}^2 denote square-integrable functions. According to the Koopman operator theory, there exists an infinite-dimensional linear operator $\mathcal{M}_{\mathbf{f}}^t : [0, \infty) \times \mathcal{L}^2(\mathcal{X}) \mapsto \mathcal{L}^2(\mathcal{X})$, such that $[\mathcal{M}_{\mathbf{f}}^t g](\mathbf{x}) = g(S_{\mathbf{f}}^t(\mathbf{x}))$ with $g \in \mathcal{L}^2$. The Koopman operator is then defined as the infinitesimal generator of $\mathcal{M}_{\mathbf{f}}^t$, with

$$[\mathcal{K}_{\mathbf{f}} g](\mathbf{x}) := \lim_{t \rightarrow 0^+} ([\mathcal{M}_{\mathbf{f}}^t g](\mathbf{x}) - g(\mathbf{x})) / t = L_{\mathbf{f}} g(\mathbf{x}), \quad (2)$$

where $L_{\mathbf{f}}$ is the Lie derivative along \mathbf{f} .

Definition 2.1: A set of observations $\mathbf{g}(\mathbf{x}) = [g_1(\mathbf{x}), \dots, g_p(\mathbf{x})]^T$ with $g_j \in \mathcal{L}^2(\mathcal{X}) : \mathcal{X} \mapsto \mathbb{C}$ forms a Koopman-invariant subspace $\text{span}\{g_j\}_{j=1}^p$ if $[\mathcal{K}_{\mathbf{f}} \mathbf{g}](\mathbf{x}) = A_c \cdot \mathbf{g}(\mathbf{x})$, where A_c is a $p \times p$ matrix.

Therefore, if the span of general observations $\text{span}\{g_j\}_{j=1}^p$ is rich enough ($p \gg 0$), this observation space can be treated as the finite-dimensional approxima-

tion of the Koopman-invariant subspace, and yields the lifted dynamics for (1):

$$\dot{\mathbf{g}}(\mathbf{x}) = \underbrace{L_{\mathbf{f}} \mathbf{g}(\mathbf{x})}_{=A_c \cdot \mathbf{g}(\mathbf{x})} + \sum_{i=1}^m L_{\mathbf{b}_i} \mathbf{g}(\mathbf{x}) \cdot u_i. \quad (3)$$

Assumption 2.1: For the Koopman lifted dynamics (3), the Lie derivative of the observations along the control-affine vector field $\{L_{\mathbf{b}_i} \mathbf{g}\}_{i=1}^m$ and the state vector \mathbf{x} lies inside $\text{span}\{g_j\}_{j=1}^p$.

Similarly, Assumption 2.1 holds under sufficiently rich Koopman observations. Defining the lifted state $\mathbf{z} := \mathbf{g}(\mathbf{x}) \in \mathcal{Z} \subset \mathbb{R}^p$ and separating the state-independent control term, (3) admits a bilinear form:

$$\dot{\mathbf{z}} = A_c \cdot \mathbf{z} + B_{c_0} \cdot \mathbf{u} + \sum_{i=1}^m B_{c_i} \cdot \mathbf{z} \cdot u_i \quad (4)$$

$$\mathbf{x} = C \cdot \mathbf{z}, \quad (5)$$

where A_c and $B_{c_i}|_{i=1\dots m} \in \mathbb{R}^{p \times p}$, $B_{c_0} \in \mathbb{R}^{p \times m}$, and $C \in \mathbb{R}^{n \times p}$.

2.2 Data-driven Koopman Approximation

From now on, we consider the discrete version of system (1) and (4) as

$$\mathbf{x}_{k+1} = \mathbf{f}_d(\mathbf{x}_k) + \sum_{i=1}^m \mathbf{b}_{d_i}(\mathbf{x}_k) \cdot u_{k_i}, \quad (6)$$

$$\mathbf{z}_{k+1} = A_d \cdot \mathbf{z}_k + B_{d_0} \cdot \mathbf{u}_k + \sum_{i=1}^m B_{d_i} \cdot \mathbf{z}_k \cdot u_{k_i}. \quad (7)$$

Given consecutive trajectories $X = [\mathbf{x}_1, \dots, \mathbf{x}_N]$ and $U = [\mathbf{u}_1, \dots, \mathbf{u}_{N-1}]$, define time-delayed measurements $X_c = [\mathbf{x}_1, \dots, \mathbf{x}_{N-1}]$ and $X_d = [\mathbf{x}_2, \dots, \mathbf{x}_N]$. Let $A_h := [A_d, B_{d_0}, \dots, B_{d_m}]$ be the concatenated Koopman bilinear matrices. With fixed Koopman observations $\{g_j\}_{j=1}^p$, the lifted measurements $Z_c := \mathbf{g}(X_c)$ and $Z_d := \mathbf{g}(X_d)$ yield the optimal solution A_h^* by solving the following least-squares problem through pseudo-inverse:

$$A_h^* := \arg \min_{A_h \in \mathbb{R}^{p \times (mp+p+m)}} \left\| Z_d - A_h \cdot \underbrace{\begin{bmatrix} Z_c \\ U \\ U \otimes Z_c \end{bmatrix}}_{=Z_U} \right\|_F = Z_d \cdot Z_U^\dagger, \quad (8)$$

where $U \otimes Z_c := [\mathbf{u}_1 \otimes \mathbf{z}_1, \dots, \mathbf{u}_{N-1} \otimes \mathbf{z}_{N-1}]$ represents the element-wise Kronecker product between the datasets Z_c and U , $\|\cdot\|_F$ denotes the Frobenius norm, and $(\cdot)^\dagger$ stands for matrix pseudo-inverse. Similarly, $C^* = X_c \cdot Z_c^\dagger$ can be derived in the same way. Koopman observations can be constructed from manually selected libraries (e.g., radial basis functions, polynomials) or learned via DNNs, with the latter often providing better approximations of the Koopman-invariant subspace and state reconstruction.

2.3 Principles of Actor-Critic Methods

Consider a discrete-time Markov decision process (MDP) defined by $(\mathcal{X}, \mathcal{U}, \{\mathbf{f}_d, \mathbf{b}_{d_i}\}_{i=1}^m, r, \gamma)$, where $\{\mathbf{f}_d, \mathbf{b}_{d_i}\}_{i=1}^m$ denotes the discrete dynamics (6), $r(\mathbf{x}_k, \mathbf{u}_k) \in \mathbb{R}$ is the immediate reward function, and $\gamma \in (0, 1]$ is the discount factor, penalizing delayed rewards. Within a canonical actor-critic setup, a stochastic control policy $\pi_{\theta_a}(\cdot|\mathbf{x}_k)$ is implemented as an actor network parameterized by θ_a . The state-value function under policy π_{θ_a} is given as

$$\begin{aligned} V^{\pi_{\theta_a}}(\mathbf{x}_k) &:= \mathbb{E}_{\pi_{\theta_a}} \left[\sum_{t=k}^{\infty} \gamma^{t-k} \cdot r(\mathbf{x}_t, \mathbf{u}_t) \middle| \mathbf{x}_k \right] \\ &= \mathbb{E}_{\pi_{\theta_a}} [r(\mathbf{x}_k, \mathbf{u}_k) | \mathbf{x}_k] + \gamma V^{\pi_{\theta_a}}(\mathbf{x}_{k+1}), \end{aligned} \quad (9)$$

and the corresponding action-value function is written as

$$Q^{\pi_{\theta_a}}(\mathbf{x}_k, \mathbf{u}_k) := r(\mathbf{x}_k, \mathbf{u}_k) + \gamma V^{\pi_{\theta_a}}(\mathbf{x}_{k+1}). \quad (10)$$

Therefore, the advantage of taking action \mathbf{u}_k at state \mathbf{x}_k over the current policy π_{θ_a} is

$$A^{\pi_{\theta_a}}(\mathbf{x}_k, \mathbf{u}_k) := Q^{\pi_{\theta_a}}(\mathbf{x}_k, \mathbf{u}_k) - V^{\pi_{\theta_a}}(\mathbf{x}_k). \quad (11)$$

In the standard actor-critic architecture, the state-value function is also approximated by a critic network with parameters θ_c , and is realized together with the actor network as multi-layer perceptrons. The actor network is optimized to maximize the state-value function (9) – equivalently, to increase the advantage at each state – whereas the critic network is trained to reduce the error between predicted values and the returns observed along collected trajectories.

2.4 Actor-Critic Model Predictive Control

A typical MPC cost function employing quadratic penalties is given by

$$\begin{aligned} J_{\text{MPC}} &:= \sum_{k=0}^{N_p-1} \left(\|\mathbf{x}_k^{\text{ref}} - \mathbf{x}_k\|_{Q_s} + \|\mathbf{u}_k\|_R \right) \\ &\quad + \|\mathbf{x}_{N_p}^{\text{ref}} - \mathbf{x}_{N_p}\|_{Q_T}, \end{aligned} \quad (12)$$

where $N_p > 0$ represents the prediction horizon; $Q_s, Q_T \succeq 0$ are the stage and terminal state-weight matrices, $R \succ 0$ is the input weight matrix, and \mathbf{x}_0 indicates the current state. The corresponding finite-horizon optimal control problem seeks $\{\mathbf{u}_k^*\}_{k=0}^{N_p-1}$ that minimizes the cost, subject to discrete system dynamics and to any state and input inequality constraints. The optimization is carried out in a receding-horizon fashion—only the first control input \mathbf{u}_0^* is applied, and the problem is re-solved at future steps with the horizon shifting forward. Since the weight matrices remain fixed throughout this process, the cost function cannot adapt to changing operating conditions, potentially degrading closed-loop performance.

In ACMPC, instead of having fixed weight matrices, the actor network π_{θ_a} is now viewed as a neural cost

map. By expanding the cost in (12) and grouping the quadratic and linear terms of $[\mathbf{x}_k^T, \mathbf{u}_k^T]^T$, the ACMPC cost function is formulated as

$$J_{\text{ACMPC}} := \sum_{k=0}^{N_p} \left\| \begin{bmatrix} \mathbf{x}_k \\ \mathbf{u}_k \end{bmatrix} \right\|_{Q_{\theta_a}^k(\mathbf{x}_0)} + \boldsymbol{\omega}_{\theta_a}^k(\mathbf{x}_0) \cdot \begin{bmatrix} \mathbf{x}_k \\ \mathbf{u}_k \end{bmatrix}, \quad (13)$$

where the weights for the quadratic and linear components are determined by the output of the neural cost map

$$\begin{aligned} \mathbf{q}_{\theta_a}(\mathbf{x}_0) &= [\mathbf{q}_{\theta_a}^0(\mathbf{x}_0), \dots, \mathbf{q}_{\theta_a}^{N_p}(\mathbf{x}_0), \\ &\quad \boldsymbol{\omega}_{\theta_a}^0(\mathbf{x}_0), \dots, \boldsymbol{\omega}_{\theta_a}^{N_p}(\mathbf{x}_0)] \\ &\quad \forall k = 0 \dots N_p, \mathbf{q}_{\theta_a}^k, \boldsymbol{\omega}_{\theta_a}^k \in \mathbb{R}_{>0}^{n+m} \end{aligned} \quad (14)$$

with $Q_{\theta_a}^k(\mathbf{x}_0) = \text{diag}[\mathbf{q}_{\theta_a}^k(\mathbf{x}_0)]$.

3. ACTOR-CRITIC KOOPMAN MODEL PREDICTIVE CONTROL

In ACMPC, accurate knowledge of the system model is required—yet such models are often unavailable or prohibitively expensive to obtain. To address this limitation, we propose the ACKMPC framework, which simultaneously learns a Koopman bilinear model and trains a neural cost map for KMPC from measurement data, enabling a short-prediction-horizon MPC to run in real-time without explicit model knowledge. ACKMPC training is divided into two stages. In the offline stage, DNN-parameterized Koopman observables are learned from initial rollout data. In the online stage, on-policy actor-critic updates are employed to learn the KMPC cost map and refine the Koopman bilinear matrices.

3.1 Koopman Observables Learning

During the offline stage, we first collect the rollout dataset $\mathcal{D}_{\text{off}} = \{(X^i, U^i)\}_{i=1}^{n_r}$, where each $(X^i, U^i) \in \mathcal{D}_{\text{off}}$ denotes a complete, time-ordered trajectory of states and control inputs. By reorganizing this data into time-delayed snapshots (X_d, X_c, U) and defining Koopman observables parameterized via a DNN mapping \mathbf{g}_{θ_s} , the corresponding offline loss is defined as

$$\begin{aligned} L_{\text{obs}} &:= \underbrace{\|\mathbf{g}_{\theta_s}(X_d) - A_{h(\theta_s)}^* \cdot \mathbf{g}_{\theta_s}(X_c)\|_F^2}_{l_{\text{bilinear}}} \\ &\quad + \underbrace{\|X_d - C_{\theta_s}^* \cdot A_{h(\theta_s)}^* \cdot \mathbf{g}_{\theta_s}(X_c)\|_F^2}_{l_{\text{prediction}}} \\ &\quad + \underbrace{\|X_c - C_{\theta_s}^* \cdot \mathbf{g}_{\theta_s}(X_c)\|_F^2}_{l_{\text{consistency}}} + \underbrace{\|A_{h(\theta_s)}^*\|_{1,1}}_{l_{\text{amp}}} \\ &\quad + \underbrace{\sum_{i=1}^p |\lambda_i(A_{d(\theta_s)}^*) - 1|}_{l_{\text{osc}}}, \end{aligned} \quad (15)$$

where $A_{h(\theta_s)}^* = [A_{d(\theta_s)}^*, B_{d_0(\theta_s)}^*, \dots, B_{d_m(\theta_s)}^*]$ and $C_{\theta_s}^*$ are calculated through (8) with $Z_d = \mathbf{g}_{\theta_s}(X_d)$ and $Z_c = \mathbf{g}_{\theta_s}(X_c)$. The first three loss terms penalize the model ac-

curacy in both Koopman lifted space (through l_{bilinear}) and the original state space (with $l_{\text{prediction}}$ and $l_{\text{consistency}}$). The amplification penalty l_{amp} bounds the maximum singular value of the Koopman bilinear matrices by penalizing its entrywise ℓ_1 -norm, thus suppressing noise amplification. The l_{osc} drives the eigenvalues of the unforced Koopman bilinear system onto the unit circle, preventing exponential growth or decay and promoting long-term prediction stability.

3.2 On-policy Training of Koopman Bilinear Dynamics and KMPC Neural Cost Map

Following offline learning of the Koopman observables, on-policy learning alternates between rollout and training phases. This paper uses a stochastic policy, where the differential MPC outputs the mean of a Gaussian policy $\mathbf{u}_k \sim \mathcal{N}(\mathbf{u}_k^{\text{mean}}, \Sigma)$ with fixed covariance Σ . For each sampled action \mathbf{u}_k , its log-probability $p_k = \log \pi(\mathbf{u}_k | \mathbf{x}_k)$ is recorded for advantage estimation. Trajectories are collected into the rollout dataset $\mathcal{D}_{\text{on}} = \{(X^i, U^i, \hat{X}^i, \hat{U}^i, \mathbf{p}^i, \mathbf{r}^i, \hat{\mathbf{r}}^i)\}_{i=1}^{n_r}$, where \mathbf{p}^i and \mathbf{r}^i are the sequences of probabilities and rewards for the i^{th} trajectory. The \hat{X}^i, \hat{U}^i represents the KMPC predicted trajectories at each rollout step, and $\hat{\mathbf{r}}^i$ is the calculated reward using the predicted trajectory. Once the rollout dataset buffer reaches its capacity, its contents are extracted for training and the buffer is reset, such that the collected data always reflects the most recent policy. This on-policy scheme avoids distributional mismatch and generally yields more stable updates than off-policy methods.

During the training phase, generalized advantage estimation (GAE) is employed to estimate the advantage defined by (11) for each state in the collected rollouts. For a trajectory $\mathcal{D}_{\text{on}}^i \in \mathcal{D}_{\text{on}}$ with length T , the estimated advantage for \mathbf{x}_k is given by

$$\hat{A}_k := \sum_{t=k}^{T-1} (\gamma \lambda)^{t-k} \cdot (r_t + \gamma \cdot c_{\theta_c}(\mathbf{x}_{t+1}) - c_{\theta_c}(\mathbf{x}_t)), \quad (16)$$

where $c_{\theta_c} : \mathcal{X} \mapsto \mathbb{R}$ is the critic network approximating the value function under the current policy. Here, γ is the reward discount factor, and $\lambda \in [0, 1]$ represents the trace-decay parameter that controls the bias-variance trade-off in the advantage estimation.

With the advantage estimates \hat{A}_k from (16), the critic network is trained using two complementary losses. The value-prediction loss L_{vp} fits the estimated value to the GAE-bootstrapped return, while the MPC-consistency loss enforces consistency of the estimated value along the MPC predicted trajectories at each state. Define

$$L_{\text{vp}} := \frac{1}{n_r \cdot T} \sum_{i=1}^{n_r} \sum_{k=0}^{T-1} \left[c_{\theta_c}(\mathbf{x}_k^i) - (\hat{A}_k^i + \tilde{c}_{\theta_c}(\mathbf{x}_k^i)) \right]^2, \quad (17)$$

where n_r is the number of rollout trajectories, and \tilde{c}_{θ_c} is

a frozen copy of the critic (no gradient) used to compute the targets $\hat{A}_k^i + \tilde{c}_{\theta_c}(\mathbf{x}_k^i)$. Let $\{\hat{\mathbf{x}}_{k+j}^i\}_{j=1}^{N_p}$, $\{\hat{\mathbf{u}}_{k+j}^i\}_{j=1}^{N_p-1}$, $\{\hat{\mathbf{r}}_{k+j}^i\}_{j=1}^{N_p}$ be the predicted states, actions, and rewards from \mathbf{x}_k^i . The MPC-consistency loss at \mathbf{x}_k^i is given by

$$L_{\text{mpcc}}(\mathbf{x}_k^i) := \frac{1}{N_p} \sum_{j=0}^{N_p-1} \left[c_{\theta_c}(\hat{\mathbf{x}}_{k+j}^i) - \left(\sum_{t=j}^{N_p-1} \gamma^{t-j} \hat{\mathbf{r}}_{k+t}^i + \gamma^{N_p-j} c_{\theta_c}(\hat{\mathbf{x}}_{k+N_p}^i) \right) \right]^2, \quad (18)$$

and the overall critic loss is

$$L_{\text{critic}} := L_{\text{vp}} + \frac{1}{n_r \cdot T} \sum_{i=1}^{n_r} \sum_{k=0}^{T-1} L_{\text{mpcc}}(\mathbf{x}_k^i). \quad (19)$$

The actor network updates follow a proximal policy optimization (PPO) clipped surrogate objective:

$$L_{\text{actor}} := \frac{-1}{n_r \cdot T} \sum_{i=1}^{n_r} \sum_{k=0}^{T-1} \min \left(\frac{\pi_{\theta_a}(\mathbf{u}_k^i | \mathbf{x}_k^i)}{\pi_{\hat{\theta}_a}(\mathbf{u}_k^i | \mathbf{x}_k^i)} \hat{A}_k^i, \text{clip} \left[\frac{\pi_{\theta_a}(\mathbf{u}_k^i | \mathbf{x}_k^i)}{\pi_{\hat{\theta}_a}(\mathbf{u}_k^i | \mathbf{x}_k^i)}, 1 - \epsilon, 1 + \epsilon \right] \hat{A}_k^i \right), \quad (20)$$

where $\epsilon \in (0, 1)$ is the PPO clipping parameter that constrains the probability ratio, suppressing excessively large policy updates.

During the online learning, the Koopman bilinear matrix A_h is updated by solving an exponentially weighted least-squares problem with the loss function defined as:

$$L_{\text{rls}} := \sum_{t=0}^{t_c} \rho^{t_c-t} \| Z_d(t) - A_h \cdot Z_U(t) \|_F^2, \quad (21)$$

where $\rho \in (0, 1]$ denotes the forgetting factor, and $t = 0 \dots t_c$ indexes all snapshots collected up to the current rollout. Thanks to the quadratic structure of L_{rls} , A_h can be updated using recursive least-squares (RLS) recursions without storing the full data history. This update process is integrated into the online learning loop, which is detailed in Algorithm 1. The projection matrix C follows an identical update rule to A_h and is omitted here for brevity.

4. EXPERIMENTS

The learning process and control performance of the proposed ACKMPC are evaluated on an inverted-pendulum system from the OpenAI Gym library, with the dynamics given by

$$\ddot{\theta} = \frac{3}{2} \left(\frac{g}{l} \cdot \sin(\theta) + \frac{1}{ml^2} \cdot u \right), \quad (22)$$

where the state space is $\mathbf{x} = [\cos(\theta), \sin(\theta), \dot{\theta}]^T$, and $u \in [-2, 2](N \cdot m)$.

During the training, each trajectory consists of $T =$

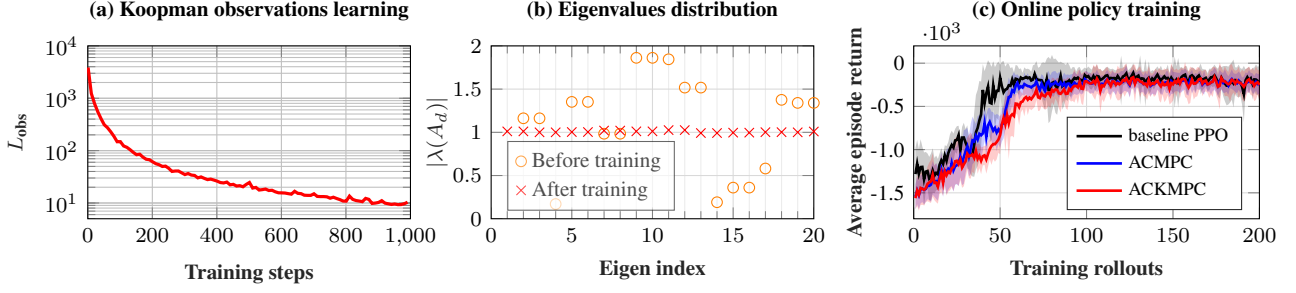


Fig. 1. (a) Offline training process of the Koopman observables g_{θ_s} . (b) Eigenvalue magnitudes distribution of the unactuated Koopman bilinear system A_d . (c) Online training process: mean episode return (solid lines) ± 1 std (shaded) over rollouts, comparing baseline PPO (actor directly outputs control actions), ACMPC, and ACKMPC.

Algorithm 1 ACKMPC Online Training

Require: Koopman observations g_{θ_s} , offline dataset \mathcal{D}_{off} . Initialize the actor network θ_a and the critic network θ_c . Set MPC prediction horizon N_p and inequality constraints.

- 1: Calculate initial A_h through (8) and covariance matrix $P \leftarrow (X_c \cdot X_c^T)^{-1}$ with data from \mathcal{D}_{off} .
- 2: **while** *trainingFlag* **do**
- 3: $\mathcal{D}_{\text{on}} \leftarrow \emptyset$
- 4: # Rollout Phase
- 5: **while** \mathcal{D}_{on} is not FULL **do**
- 6: Get rollout trajectory $\mathcal{D}_{\text{on}}^i$
- 7: $\mathcal{D}_{\text{on}} \leftarrow \mathcal{D}_{\text{on}} \cup \mathcal{D}_{\text{on}}^i$
- 8: **end while**
- 9: # Training Phase
- 10: **for** $\mathcal{D}^{\text{batch}}$ in \mathcal{D}_{on} **do**
- 11: Calculate $L_{\text{critic}}, L_{\text{actor}}$ from (19) and (20)
- 12: $\theta_a \leftarrow \text{OptimizerStep}(\theta_a, \nabla L_{\text{actor}})$
- 13: $\theta_c \leftarrow \text{OptimizerStep}(\theta_c, \nabla L_{\text{critic}})$
- 14: **end for**
- 15: Derive Z_d and Z_U from \mathcal{D}_{on}
- 16: $\tilde{P} \leftarrow (1/\rho) \cdot P$
- 17: $K \leftarrow \tilde{P} \cdot Z_U^T \cdot (I + Z_U^T \cdot \tilde{P} \cdot Z_U)^{-1}$
- 18: $A_h \leftarrow A_h + (Z_d - A_h \cdot Z_U) \cdot K^T$
- 19: $P \leftarrow \tilde{P} - \tilde{P} \cdot Z_U \cdot (I + Z_U^T \cdot \tilde{P} \cdot Z_U)^{-1} \cdot Z_U^T \cdot \tilde{P}$
- 20: **end while**

200 steps with sampling interval $t_s = 0.05s$. In the offline stage, the Koopman observable network lifts the state into a $p = 20$ dimensional space using 3 fully-connected hidden layers of 20 units each, with *Tanh* activations. It is trained on 100K samples with 20K batch size. In the online stage, both actor and critic networks employ 2 fully-connected hidden layers of 128 and 256 units with *ReLU* activations, with the output layer of the actor network using *Sigmoid* activations and the critic network being left linear. Online learning runs for 200 rollouts, each containing $n_r = 50$ trajectories (10K simulation steps per rollout, 2 million steps in total), with each batch having 2K samples and 5 gradient updates. All networks apply *Adam* optimizer with learning rate 0.001. The training was conducted on a workstation with NVIDIA RTX 4090 (24 GB) GPU and AMD Ryzen 9950X CPU.

The training process of ACKMPC is shown in Fig. 1. From Fig. 1(a), the offline training loss L_{obs} falls by more than two orders of magnitude over 1000 gradient steps, indicating that the DNN-represented Koopman observables have significantly improved precision in ap-

proximating a Koopman-invariant subspace and enable accurate linear reconstruction of the original state. The eigenvalue magnitudes $\lambda(A_d)$ of the unactuated system in Fig. 1(b) are driven onto the unit circle after training, preventing exponential growth or decay of the Koopman observables and ensuring long-term prediction stability.

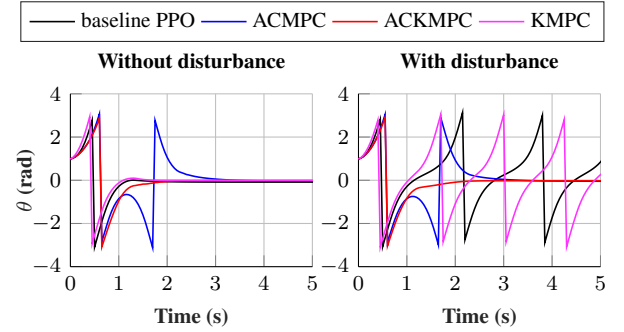


Fig. 2. Tracking performance of the pendulum angle θ under four trained controllers. Left: nominal system matching the training model. Right: system with a 50% increase in pendulum mass.

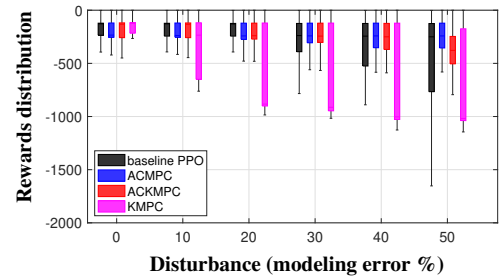


Fig. 3. Cumulative reward distributions of the trained controllers under varying levels of model perturbation, evaluated over 100 trials per disturbance level with random initial states.

The control performance is compared in Fig. 2 and Fig. 3 under varying levels of model perturbation. Each controller was evaluated over 100 independent trials for each disturbance level, with the pendulum mass perturbed by a fixed percentage relative to the training model. Fig. 2 presents two example trajectories of the pendulum angle θ . In the undisturbed case, all four methods drive the pendulum to the upright position rapidly, consistent with the online training results in Fig. 1(c).

When disturbance is introduced, only the ACMPC and ACKMPC with learned neural cost maps maintain stability, while baseline PPO and KMPC exhibit sustained oscillations. This highlights the benefit of a state-dependent cost map in guiding the controller under perturbations while still leveraging the constraint-handling capabilities of MPC. Fig. 3 shows the cumulative rewards distributions across all test runs as the modeling error increases. Baseline PPO and KMPC both degrade significantly under moderate mismatch, whereas ACKMPC closely matches ACMPC across all disturbance levels. These results demonstrate that the proposed purely data-driven ACKMPC approach attains robustness comparable to a model-based ACMPC.

Meanwhile, by leveraging a neural cost map, both ACMPC and ACKMPC achieve the performance shown earlier with a horizon of $N_p = 2$, requiring less than $5ms$ per control update with the IPOPT solver. In contrast, KMPC demands substantially longer horizons to maintain stability, where the solving time grows to over $60ms$ at $N_p = 30$, posing a significant challenge for real-time deployment.

5. CONCLUSIONS

This paper introduced ACKMPC, a fully data-driven framework that identifies DNN parameterized Koopman observables offline, and jointly refines the Koopman bilinear dynamics and trains a state-dependent neural cost map online via on-policy DRL. Leveraging this learned cost map, ACKMPC achieves less than $5ms$ solve times with a 2-step prediction horizon while matching the tracking accuracy and disturbance robustness of a model-based ACMPC. Compared to standard KMPC and baseline PPO, ACKMPC delivers superior robustness to model mismatch and greater computational efficiency.

Future work considers testing ACKMPC on a wider range of systems including real-world hardware platforms, integrating safety and stability criteria directly into the neural cost map training, and leveraging historical trajectory data to learn multiple specialized cost maps alongside a selection network that adapts to diverse control scenarios.

REFERENCES

- [1] S. L. Brunton and J. N. Kutz, *Data-driven science and engineering: Machine learning, dynamical systems, and control*. Cambridge University Press, 2022.
- [2] M. O. Williams, I. G. Kevrekidis, and C. W. Rowley, "A data-driven approximation of the koopman operator: Extending dynamic mode decomposition," *Journal of Nonlinear Science*, vol. 25, pp. 1307–1346, 2015.
- [3] M. Korda and I. Mezić, "Linear predictors for nonlinear dynamical systems: Koopman operator meets model predictive control," *Automatica*, vol. 93, pp. 149–160, 2018.
- [4] Y. Han, W. Hao, and U. Vaidya, "Deep learning of koopman representation for control," in *2020 59th IEEE Conference on Decision and Control (CDC)*. IEEE, 2020, pp. 1890–1895.
- [5] K. Zheng, P. Huang, and G. P. Fettweis, "Optimal control of quadrotor attitude system using data-driven approximation of koopman operator," *IFAC-PapersOnLine*, vol. 56, no. 2, pp. 834–840, 2023.
- [6] A. Surana, "Koopman operator based observer synthesis for control-affine nonlinear systems," in *2016 IEEE 55th Conference on Decision and Control (CDC)*. IEEE, 2016, pp. 6492–6499.
- [7] D. Bruder, X. Fu, and R. Vasudevan, "Advantages of bilinear koopman realizations for the modeling and control of systems with unknown dynamics," *IEEE Robotics and Automation Letters*, vol. 6, no. 3, pp. 4369–4376, 2021.
- [8] H. Shi and M. Q.-H. Meng, "Deep koopman operator with control for nonlinear systems," *IEEE Robotics and Automation Letters*, vol. 7, no. 3, pp. 7700–7707, 2022.
- [9] R. Strässer, M. Schaller, K. Worthmann, J. Berberich, and F. Allgöwer, "Koopman-based feedback design with stability guarantees," *IEEE Transactions on Automatic Control*, 2024.
- [10] K. Zheng, P. Huang, A. Villamil, J. Casas, and G. P. Fettweis, "Learning koopman bilinear models with multiplication-closed observations for linear optimal controller design," in *2025 American Control Conference (ACC)*, 2025, accepted for publication.
- [11] C. Folkestad, S. X. Wei, and J. W. Burdick, "Koopnet: Joint learning of koopman bilinear models and function dictionaries with application to quadrotor trajectory tracking," in *2022 International Conference on Robotics and Automation (ICRA)*. IEEE, 2022, pp. 1344–1350.
- [12] Z. Wang, H. Zhang, and J. Wang, "K-bmpc: Derivative-based koopman bilinear model predictive control for tractor-trailer trajectory tracking with unknown parameters," in *2024 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 2024, pp. 5808–5813.
- [13] K. Worthmann, R. Strässer, M. Schaller, J. Berberich, and F. Allgöwer, "Data-driven mpc with terminal conditions in the koopman framework," *arXiv preprint arXiv:2408.12457*, 2024.
- [14] A. Romero, Y. Song, and D. Scaramuzza, "Actor-critic model predictive control," in *2024 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 2024, pp. 14 777–14 784.
- [15] B. Amos, I. Jimenez, J. Sacks, B. Boots, and J. Z. Kolter, "Differentiable mpc for end-to-end planning and control," *Advances in neural information processing systems*, vol. 31, 2018.