Linear or Bilinear: A Criterion for Koopman Rollouts in Sampling-Based Predictive Control

Kangmin Lee , Sanghyun Kim*

Abstract: Koopman operator models enable sampling-based Model Predictive Path Integral (MPPI) control to roll out thousands of sampled input sequences through a low-cost learned lifted model. However, recent Koopman MPPI controllers keep a constant input matrix, which is suitable for convex Koopman MPC, but not needed for sampling. It fixes a single gain from input to tracked output, yet for many robots the true gain varies with heading or attitude. This paper supplies a criterion for when the states a task visits rule out such a matrix. If the output is read linearly from the lifted state, the largest gap between the predicted and true gains over those states is at least half the true gain's variation. This holds for any lifting features and every horizon step, but the criterion says nothing if the input moves the output only through acceleration or higher derivatives. Where the gain varies strongly, as for a freely turning unicycle or a drone over a wide yaw range, linear rollouts fail. A bilinear rollout, where the input matrix instead depends on the lifted state, stays close to the true-model oracle. Its MPPI update costs under twice the time of a linear update. Where the gain varies little, the linear rollout is less accurate than the bilinear one in most tests but can remain within the task tolerance. Yet passing the criterion is not enough. In a pendulum swing-up, linear models of matching width fail at long horizons even for outputs with a constant one-step input gain, while bilinear models with the same outputs can succeed. The cheaper linear rollout is therefore worth trying at small gain variation, but a closed-loop test is still needed to confirm it works for a given task. Hyperparameters and videos: `https://rcilab.github.io/koopman-mppi`.

Keywords: Bilinear systems, Koopman operator, model predictive path integral control, sampling-based MPC.

1. INTRODUCTION

For many robotic platforms, an accurate analytical model is not available. Friction, actuation dynamics, and platform-specific parameters are hard to derive by hand, and a calibrated model rarely stays accurate in deployment. Data-driven predictive control instead identifies the rollout model from input–state trajectories. The learned model must then predict the tracked outputs accurately and cheaply. Both demands are hardest to meet in sampling-based model predictive control (MPC), most prominently Model Predictive Path Integral (MPPI) control [1,2]. MPPI places no requirements such as convexity, differentiability, or affineness on the rollout dynamics. In exchange, at every control step it propagates many sampled input sequences over the horizon [3]. Sampling-based control therefore needs a structured, parallelizable rollout.

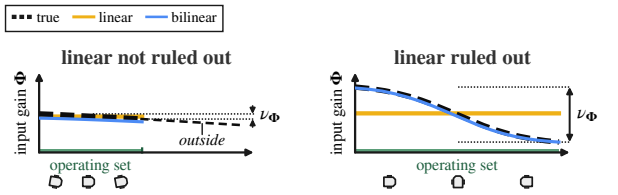
Koopman operator models [4] provide this structure and are identifiable from data [5,6]. Lifting the state through a dictionary of observables makes the autonomous dynamics approximately linear in the lifted space. With the input entering through a constant input matrix, the lifted model is linear, and each rollout then reduces to matrix–vector products. Recent work accordingly reports speedups with MPPI using a learned *linear* Koopman rollout [7].

The *linear* form of these lifted models, however, is suited to a different optimizer. In Koopman MPC, the linear predictor keeps the online problem a convex quadratic program [8]. Bilinear models, in which the input also multiplies the lifted state, have improved prediction and control in optimization-based Koopman MPC [9, 10]. Their MPC problems are nonconvex, and some works make them convex by fixing the lifted state in the bilinear terms over the horizon [9, 11]. Sampling-based control handles such nonconvexity directly, yet the constant input matrix persists in the Koopman MPPI controllers of [7, 12].

A constant input matrix cannot represent a state-dependent input gain. This limitation is the starting point of bilinear Koopman models [9, 10, 13]. They are recommended for nonholonomic robots, whose input gain turns with the heading [14, 15]. To our knowledge, however, no test relates the error of a constant input matrix to the states a task visits, and no such relation has been checked against the closed-loop failure of MPPI with a linear rollout. *When does a task rule out a constant input matrix, and what does the bilinear term cost?*

To answer the first question, we define the input gain $\mathbf{\Phi}$ of the tracked output and its variation ν_Φ over the states

Kangmin Lee and Sanghyun Kim are with the Department of Mechanical Engineering, Kyung Hee University, Yongin, South Korea.
* Corresponding author (e-mail: kim87@khu.ac.kr).



(a) (b)

Fig. 1. The gain-variation criterion under a linear readout, for one vehicle in two tasks. (a) A narrow heading range keeps $\mathbf{\Phi}$ nearly constant. (b) A freely turning heading makes $\mathbf{\Phi}$ vary widely, and a linear lift is ruled out if the bound exceeds the tolerance.

a task visits. Under a linear readout, a linear lifted model predicts the same input gain at every state. At some state, this gain therefore misses the true one by at least half the variation ν_Φ. Fig. 1 shows this bound. We use this elementary bound as a necessary condition testable before identification when the input gain is known, even if the remaining dynamics are not. We then assess it in MPPI with linear and bilinear Koopman rollouts across levels of gain variation, compare alternative rollout classes, and measure the computational cost.

Our contributions are threefold:

- We lower-bound the input-response error of linear lifted models under a linear readout, extend the bound to every horizon step, and give the least-squares floor of a fitted linear model.
- In a graded drone study, the linear models reach the least-squares floor, and closed-loop failure begins at a tolerance-dependent level of gain variation. With large variation, linear rollouts fail on the unicycle and the velocity-commanded drone. On both, the bilinear rollout nearly matches the true-model oracle, at less than twice the MPPI update time of the linear one. A neural rollout does so only on the drone or with clamped inputs, and it needs more than ten times the operations of the bilinear one.
- Small variation leaves room for the cheaper linear model, which can meet the task tolerance, although the bilinear model is usually more accurate. On a torque-driven arm, the linear error stays small for the joint angles but grows with the one-step gain variation for the end effector. Yet a linear model that the criterion does not bar can still fail. In a pendulum swing-up with a horizon of 30 steps or more, bilinear models can succeed, whereas linear models of the same width fail in every tested setting. This holds even when the models predict the angle itself, whose one-step gain is constant.

2. RELATED WORK

Rollout models in sampling-based MPC. In its information-theoretic form, MPPI admits general nonlinear rollout dynamics [1]. Other work builds it into integrated planning and control stacks for ground vehicles [16] or improves the *sampler* around a given model [17]. The rollout model itself spans learned neural networks [1] and full GPU physics engines [3]. We hold the sampler fixed and ask which input structure a learned rollout needs.

Koopman predictors and MPC. Extended dynamic mode decomposition (EDMD) and its controlled variant identify a finite lifted predictor from data [5, 8], and deep networks learn the dictionary itself [6]. Korda and Mezić turn the linear predictor into a convex MPC pipeline [8], and this form has since become common in Koopman-based control [18]. Applications include automated driving [19] and soft robots [20].

Bilinear Koopman models. With fixed dictionaries, bilinear realizations outperform linear ones in prediction and robotic MPC on a simulated torque-driven arm [9] and a planar quadrotor [10]. Dictionaries have also been learned jointly with bilinear models [11]. For nonholonomic robots, in which state and input enter as products, bilinear surrogates are beneficial [14] and have been used in gradient-based MPC with hardware validation [15].

Koopman rollouts in sampling-based control. The closest work couples MPPI with a learned *linear* deep Koopman rollout, swings up a pendulum with it, and reports speedups on a thrust-driven surface vehicle and an acceleration-commanded quadruped [7]. For the surface vehicle, the body-to-world rotation is integrated outside the learned model, which therefore receives a body-frame input. A Koopman model with a constant input matrix also serves as the MPPI rollout for manipulators in [12].

3. CRITERION AND CONTROLLER

3.1. Problem Setup and Preliminaries

System and objective. We consider a control-affine system $\dot{\boldsymbol{x}} = f(\boldsymbol{x}) + \boldsymbol{G}(\boldsymbol{x})\boldsymbol{u}$, with state $\boldsymbol{x} \in \mathbb{R}^n$, admissible input $\boldsymbol{u} \in \mathbb{U} \subset \mathbb{R}^m$, drift f, and state-dependent input matrix $\boldsymbol{G}(\boldsymbol{x})$. Discretization with step Δt under a zero-order hold gives $\boldsymbol{x}_{t+1} = F(\boldsymbol{x}_t, \boldsymbol{u}_t)$, where the transition map F is known only through input–state data $\{(\boldsymbol{x}_t, \boldsymbol{u}_t, \boldsymbol{x}_{t+1})\}$. Over a horizon T, the controller seeks the input sequence $U = (\boldsymbol{u}_0, \dots, \boldsymbol{u}_{T-1})$ minimizing the cost

$$J(U) = \phi(\boldsymbol{x}_T) + \textstyle\sum_{t=0}^{T-1} q(\boldsymbol{x}_{t+1}, \boldsymbol{u}_t), \tag{1}$$

with stage cost q, which may also depend on the previous input (Sec. 4), and terminal cost ϕ. Throughout, $\|\cdot\|$ denotes the Euclidean norm and its induced matrix norm.

MPPI. MPPI draws K candidate input sequences $V^{(k)} = (\boldsymbol{v}_0^{(k)}, \dots, \boldsymbol{v}_{T-1}^{(k)})$, where $\boldsymbol{v}_t^{(k)} = P_c(\boldsymbol{u}_t + \boldsymbol{\varepsilon}_t^{(k)})$, with P_c the

identity or a projection onto $\mathbb{U}$, and $\varepsilon_t^{(k)} \sim \mathcal{N}(\mathbf{0}, \boldsymbol{\Sigma})$ with sampling covariance $\boldsymbol{\Sigma} = \mathrm{diag}(\boldsymbol{\sigma}^2)$. It rolls each sequence through a model of F, accumulating the cost (1) into a trajectory cost $S(V^{(k)})$, and updates each input using the cost-weighted average of the perturbations [2]:

$$\begin{aligned} \boldsymbol{u}_t &\leftarrow \boldsymbol{u}_t + \frac{\sum_k w^{(k)} \varepsilon_t^{(k)}}{\sum_k w^{(k)}}, \\ w^{(k)} &= \exp\Big(-\tfrac{1}{\lambda}\big(S(V^{(k)}) - \rho\big)\Big), \end{aligned} \tag{2}$$

with temperature $\lambda > 0$ and $\rho = \min_k S(V^{(k)})$.

Koopman lifting. Koopman methods lift the state through a dictionary $\boldsymbol{\Psi} : \mathbb{R}^n \to \mathbb{R}^N$ of lifted dimension N and model the evolution of $\boldsymbol{z} = \boldsymbol{\Psi}(\boldsymbol{x})$. The leading coordinates of $\boldsymbol{z}$ are the observables $\boldsymbol{b}(\boldsymbol{x})$ read by the cost, so a constant selector $\boldsymbol{C}$ recovers them, $\boldsymbol{b}(\boldsymbol{x}) = \boldsymbol{C}\boldsymbol{\Psi}(\boldsymbol{x})$. The *linear* and *bilinear* predictors are

$$\text{(linear)} \quad \boldsymbol{z}^+ \approx \boldsymbol{A}\boldsymbol{z} + \boldsymbol{B}\boldsymbol{u}, \tag{3}$$

$$\text{(bilinear)} \quad \boldsymbol{z}^+ \approx \boldsymbol{A}\boldsymbol{z} + \boldsymbol{B}_0\boldsymbol{u} + \textstyle\sum_{i=1}^m u_i(\boldsymbol{B}_i\boldsymbol{z}), \tag{4}$$

with next lifted state $\boldsymbol{z}^+$, state matrix $\boldsymbol{A}$, input matrices $\boldsymbol{B}$ and $\boldsymbol{B}_0$, and coupling matrices $\{\boldsymbol{B}_i\}$. Defining $\boldsymbol{M}(\boldsymbol{u}) := \boldsymbol{A} + \sum_i u_i \boldsymbol{B}_i$ gives the bilinear step $\boldsymbol{z}^+ = \boldsymbol{M}(\boldsymbol{u})\boldsymbol{z} + \boldsymbol{B}_0\boldsymbol{u}$.

3.2. The Criterion

The predictors differ only in how the input enters. Let $\boldsymbol{y} = h(\boldsymbol{x}) \in \mathbb{R}^{n_y}$ be the tracked output, read by the cost, with first-order response over one step

$$\boldsymbol{y}_{t+1} - \boldsymbol{y}_t = \big(\boldsymbol{d}(\boldsymbol{x}_t) + \boldsymbol{\Phi}(\boldsymbol{x}_t)\,\boldsymbol{u}_t\big)\Delta t + \mathcal{O}(\Delta t^2), \tag{5}$$

drift response $\boldsymbol{d}(\boldsymbol{x}) := \nabla h(\boldsymbol{x})\, f(\boldsymbol{x})$, and input gain $\boldsymbol{\Phi}(\boldsymbol{x}) := \nabla h(\boldsymbol{x})\, \boldsymbol{G}(\boldsymbol{x}) \in \mathbb{R}^{n_y \times m}$, where ∇h denotes the Jacobian of h. For f and $\boldsymbol{G}$ of class C^1, an output of class C^2, and bounded inputs, the remainder is uniform on compact sets. We assume that h is realizable through a linear readout, $h(\boldsymbol{x}) = \boldsymbol{C}_h\boldsymbol{\Psi}(\boldsymbol{x})$, where the constant $\boldsymbol{C}_h \in \mathbb{R}^{n_y \times N}$ has full row rank. For $h(\boldsymbol{x}) = \boldsymbol{\Pi}\boldsymbol{b}(\boldsymbol{x})$ with a selection matrix $\boldsymbol{\Pi}$, $\boldsymbol{C}_h = \boldsymbol{\Pi}\boldsymbol{C}$. The operating set $\mathcal{X} \subset \mathbb{R}^n$ is the compact set of states that the task visits. The criterion uses the *gain variation* over $\mathcal{X}$ and the normalized variation,

$$\begin{aligned} \nu_\Phi &:= \sup_{\boldsymbol{x},\boldsymbol{x}' \in \mathcal{X}} \big\|\boldsymbol{\Phi}(\boldsymbol{x}) - \boldsymbol{\Phi}(\boldsymbol{x}')\big\|, \\ \hat{\nu} &:= \nu_\Phi \Big/ \sup_{\boldsymbol{x} \in \mathcal{X}} \|\boldsymbol{\Phi}(\boldsymbol{x})\| \in [0, 2], \end{aligned} \tag{6}$$

with $\hat{\nu} := 0$ if $\boldsymbol{\Phi} \equiv \mathbf{0}$ on $\mathcal{X}$. Both depend on h and $\mathcal{X}$ and, with several input or output channels, on their scaling.

Lemma 1 (Chebyshev radius)**:** For every bounded family of matrices $\{\boldsymbol{\Gamma}(\boldsymbol{x})\}_{\boldsymbol{x} \in \mathcal{X}}$ and every constant matrix $\bar{\boldsymbol{\Gamma}}$, $\sup_{\boldsymbol{x} \in \mathcal{X}} \|\boldsymbol{\Gamma}(\boldsymbol{x}) - \bar{\boldsymbol{\Gamma}}\| \geq \frac{1}{2} \sup_{\boldsymbol{x},\boldsymbol{x}' \in \mathcal{X}} \|\boldsymbol{\Gamma}(\boldsymbol{x}) - \boldsymbol{\Gamma}(\boldsymbol{x}')\|$.

Proof: Take suprema in $\|\boldsymbol{\Gamma}(\boldsymbol{x}) - \boldsymbol{\Gamma}(\boldsymbol{x}')\| \leq \|\boldsymbol{\Gamma}(\boldsymbol{x}) - \bar{\boldsymbol{\Gamma}}\| + \|\boldsymbol{\Gamma}(\boldsymbol{x}') - \bar{\boldsymbol{\Gamma}}\|$. □

Proposition 1 (Criterion)**:** Consider one-step predictions $\hat{\boldsymbol{y}}_{t+1} = \boldsymbol{C}_h\hat{\boldsymbol{z}}_{t+1}$ from $\hat{\boldsymbol{z}}_t = \boldsymbol{\Psi}(\boldsymbol{x}_t)$. (i) For the linear model (3), the predicted input gain $\partial\hat{\boldsymbol{y}}_{t+1}/\partial\boldsymbol{u}_t = \boldsymbol{C}_h\boldsymbol{B}$ is constant, and for every $\boldsymbol{B}$ and every dictionary $\boldsymbol{\Psi}$,

$$\sup_{\boldsymbol{x} \in \mathcal{X}} \big\|\Delta t\, \boldsymbol{\Phi}(\boldsymbol{x}) - \boldsymbol{C}_h\boldsymbol{B}\big\| \;\geq\; \tfrac{1}{2}\,\nu_\Phi\,\Delta t. \tag{7}$$

The input response is the change $\boldsymbol{y}_{t+1}(\boldsymbol{u}) - \boldsymbol{y}_{t+1}(\mathbf{0})$ that an input $\boldsymbol{u}$ causes. By (7), for every input size $s > 0$ such that all inputs with $\|\boldsymbol{u}\| \leq s$ are admissible, the predicted input response misses the true one by at least $\frac{1}{2}\nu_\Phi s\Delta t - \mathcal{O}(\Delta t^2)$ at some $\boldsymbol{x} \in \mathcal{X}$ and $\boldsymbol{u}$ with $\|\boldsymbol{u}\| = s$. (ii) For the bilinear model (4), the predicted input gain $\partial\hat{\boldsymbol{y}}_{t+1}/\partial\boldsymbol{u}_t = \boldsymbol{C}_h\big(\boldsymbol{B}_0 + [\boldsymbol{B}_1\hat{\boldsymbol{z}}_t, \ldots, \boldsymbol{B}_m\hat{\boldsymbol{z}}_t]\big)$ is affine in the lifted state. This gain equals $\Delta t\,\boldsymbol{\Phi}(\boldsymbol{x})$ on $\mathcal{X}$ for some $\boldsymbol{B}_0$ and $\{\boldsymbol{B}_i\}$ if and only if $\boldsymbol{\Phi}(\boldsymbol{x}) = \boldsymbol{\Phi}_0 + [\boldsymbol{W}_1\boldsymbol{\Psi}(\boldsymbol{x}), \ldots, \boldsymbol{W}_m\boldsymbol{\Psi}(\boldsymbol{x})]$ on $\mathcal{X}$ for constant $\boldsymbol{\Phi}_0$ and $\boldsymbol{W}_i$.

Proof: (i) Constancy is immediate from (3), and (7) is Lemma 1 with $\boldsymbol{\Gamma}(\boldsymbol{x}) = \Delta t\,\boldsymbol{\Phi}(\boldsymbol{x})$ and $\bar{\boldsymbol{\Gamma}} = \boldsymbol{C}_h\boldsymbol{B}$. As $\boldsymbol{\Phi}$ is continuous and $\mathcal{X}$ compact, the supremum in (7) is attained at some $\boldsymbol{x}$. Let $\boldsymbol{u} = s\boldsymbol{\eta}$, where the unit vector $\boldsymbol{\eta}$ attains the induced norm of $\Delta t\,\boldsymbol{\Phi}(\boldsymbol{x}) - \boldsymbol{C}_h\boldsymbol{B}$. By (5), the true input response at $\boldsymbol{x}$ is $\Delta t\,\boldsymbol{\Phi}(\boldsymbol{x})\boldsymbol{u} + \mathcal{O}(\Delta t^2)$ while the model's is $\boldsymbol{C}_h\boldsymbol{B}\boldsymbol{u}$. Their difference is $(\Delta t\,\boldsymbol{\Phi}(\boldsymbol{x}) - \boldsymbol{C}_h\boldsymbol{B})\boldsymbol{u} + \mathcal{O}(\Delta t^2)$. Its first term has norm s times the supremum in (7), hence at least $\frac{1}{2}\nu_\Phi s\Delta t$. (ii) Differentiating (4) gives the stated gain. If this gain equals $\Delta t\,\boldsymbol{\Phi}(\boldsymbol{x})$ on $\mathcal{X}$, the condition holds with $\boldsymbol{\Phi}_0 = \boldsymbol{C}_h\boldsymbol{B}_0/\Delta t$ and $\boldsymbol{W}_i = \boldsymbol{C}_h\boldsymbol{B}_i/\Delta t$. Conversely, since $\boldsymbol{C}_h$ has full row rank, there are $\boldsymbol{B}_0$ and $\boldsymbol{B}_i$ with $\boldsymbol{C}_h\boldsymbol{B}_0 = \Delta t\,\boldsymbol{\Phi}_0$ and $\boldsymbol{C}_h\boldsymbol{B}_i = \Delta t\,\boldsymbol{W}_i$. □

By Proposition 1(i), a linear rollout can keep the one-step input-response error below a tolerance κ over $\mathcal{X}$ for all inputs of size s only if $\frac{1}{2}\nu_\Phi s\Delta t - \mathcal{O}(\Delta t^2) \leq \kappa$.

Relation to known results. Bruder et al. give conditions under which a set of observables admits a linear or a bilinear realization and state that control-affine systems admit a bilinear one over an infinite basis of observables [9]. Iacob et al. derive the exact lifted form with inputs and upper-bound the error of approximations with a constant input matrix [13]. Proposition 1(ii) applies the input part of the condition in [9] to the tracked output, to first order in Δt. Part (i) bounds from below, to first order, the output part of the input-matrix deviation in the bound of [13]. Our bound uses the input gain of the tracked output over the operating set and can be evaluated before identification.

Remark 1 (Attained bound and least-squares floor)**:** For the planar position of a velocity-commanded vehicle whose heading covers at least a half turn, $\hat{\nu} = 2$, so $\boldsymbol{C}_h\boldsymbol{B} = \mathbf{0}$ attains equality in (7). For a heading range $[-\bar{\theta}, \bar{\theta}]$ with $\bar{\theta} < \pi/2$, $\boldsymbol{C}_h\boldsymbol{B}$ equal to $\cos\bar{\theta}$ times $\Delta t\,\boldsymbol{\Phi}$ at zero heading attains it. A least-squares fit need not minimize the worst case. Let the training inputs be indepen-

dent of the state, with zero mean and nonsingular covariance Σ_u. Then the input error $(\Delta t\,\Phi(x) - C_h B)u$ and the state-only error, the one-step error at zero input, are uncorrelated, so their mean squares add. Up to $\mathcal{O}(\Delta t^3)$, the one-step mean squared error of every linear lifted model under the training distribution is therefore at least the floor $\Delta t^2\,\mathbb{E}\|(\Phi - \mathbb{E}\Phi)u\|^2$, the minimum of the input-error term over $C_h B$. A model with zero state-only error attains it at the least-squares value $C_h B = \Delta t\,\mathbb{E}[\Phi(x)] + \mathcal{O}(\Delta t^2)$.

Corollary 1 (Multi-step predictions)**:** The sensitivity of the linear prediction $\hat{y}_{t+k}$ to the most recent input u_{t+k-1} is $C_h B$ for every $k \geq 1$. Hence (7) applies at every step, with $\mathcal{X}$ replaced by the true states x_{t+k-1} reached from $\mathcal{X}$. Let $\chi_k(x;U) := y_{t+k}(U) - y_{t+k}(0)$ be the true input response at step k to inputs $U = (u_t, \ldots, u_{t+k-1})$ from $x_t = x$. The linear model predicts the same $\hat{\chi}_k(U) = \sum_{j<k} C_h A^{k-1-j} B u_{t+j}$ from every x, so by Lemma 1, for every A, B, and U, with suprema over $\mathcal{X}$,

$$\sup_x \|\chi_k(x;U) - \hat{\chi}_k(U)\| \geq \tfrac{1}{2} \sup_{x,x'} \|\chi_k(x;U) - \chi_k(x';U)\|.$$

Suppose that $d \equiv 0$, $U = (u, \ldots, u)$, and holding u from any $x \in \mathcal{X}$ leaves Φ unchanged (e.g., a forward command with zero turn rate). Then $\chi_k(x;U) = k\Delta t\,\Phi(x)u$, and the right-hand side equals $\frac{1}{2} k\Delta t \sup_{x,x' \in \mathcal{X}} \|(\Phi(x) - \Phi(x'))u\|$, which grows linearly with k.

Proof: With $\hat{z}_t = \Psi(x)$, iterating (3) gives $\hat{y}_{t+k} = C_h A^k \Psi(x) + \hat{\chi}_k(U)$, where u_{t+k-1} enters through $C_h B$. The first term, the zero-input prediction, cancels in the input response. Under the additional assumptions, $\dot{y} = \Phi(x)u$ along the held trajectory and $\dot{y} = 0$ under zero input. □

Remark 2 (Scope)**:** Proposition 1 gives a necessary condition for a linear rollout. It assumes the smoothness stated after (5) and a linear readout $\hat{y} = C_h \hat{z}$. It does not fully cover three rollout classes (Sec. 4.2). First, with a nonlinear decoder $\hat{y} = g(\hat{z})$ [6], the predicted input gain $\nabla g(\hat{z}) B$ depends on the state. Second, for an input matrix evaluated at the current state and held over the horizon [9, 11], the first statement of Corollary 1 still applies within each horizon. Third, unstructured neural dynamics are not lifted models. Even within its assumptions, a small $\hat{\nu}$ does not imply that a linear model suffices. If every output component has relative degree two or higher, $\Phi \equiv 0$ and the criterion is silent. Lemma 1 still applies to the exact one-step gain $\partial y_{t+1} / \partial u_t$.

Remark 3 (Evaluating the criterion)**:** For a velocity-commanded base, a body-frame command reaches the world-frame planar position through the heading rotation. Hence $\hat{\nu} = 2\sin\bar{\theta}$ if the headings in $\mathcal{X}$ span an interval of width $2\bar{\theta} \leq \pi$, and $\hat{\nu} = 2$ for wider ranges. We fix $\mathcal{X}$ from the task, not from a learned controller. It is a range that the reference or the task prescribes, or the set of states that the oracle visits or the test data contain. We compute Φ from the known kinematics or by finite differences of the simulator. When G is unknown, regressed local gains could replace Φ, but we did not evaluate them.

Algorithm 1 Bilinear Koopman MPPI (one control step)

Require: state x; warm start $U = (u_0, \ldots, u_{T-1})$; model $A, B_0, \{B_i\}, C, \Psi$; parameters $K, T, \Sigma, \lambda, P_c$

$z_0 \leftarrow \Psi(x)$
for $k = 1, \ldots, K$ **in parallel do**
  $v_t^{(k)} \leftarrow P_c(u_t + \varepsilon_t^{(k)})$, $\varepsilon_t^{(k)} \sim \mathcal{N}(0, \Sigma)$, for all t
  $\hat{z} \leftarrow z_0$; $S^{(k)} \leftarrow 0$
  for $t = 0, \ldots, T-1$ **do**
    $\hat{z} \leftarrow M(v_t^{(k)})\,\hat{z} + B_0\,v_t^{(k)}$;
    $S^{(k)} \leftarrow S^{(k)} + q\big(C\hat{z}, v_t^{(k)}\big)$
  end for
  $S^{(k)} \leftarrow S^{(k)} + \phi(C\hat{z})$
end for
$w^{(k)} \leftarrow \exp\big(-\frac{1}{\lambda}(S^{(k)} - \min_j S^{(j)})\big)$;
$u_t \leftarrow u_t + \big(\sum_k w^{(k)} \varepsilon_t^{(k)}\big) / \sum_j w^{(j)}$ for all t
return projection of u_0 onto $\mathbb{U}$; shift U one step

Input conditioning. Under the split $u = \bar{u} + \tilde{u}$ about an operating point $\bar{u}$, the component $\Phi(x)\bar{u}$ is a pure state function. The model's autonomous part can absorb this function if it lies in the lifted span, so only $\Phi(x)\,\tilde{u}$ requires the coupling terms. We therefore center the inputs at $\bar{u}$ and scale them to comparable ranges.

3.3. Bilinear Koopman MPPI

At each control step, we lift the state once, $z_0 = \Psi(x)$, propagate the K candidates through

$$\hat{z}_{t+1} = M(v_t)\,\hat{z}_t + B_0 v_t, \qquad \hat{z}_0 = z_0, \tag{8}$$

score each with the task cost, and apply the softmax-weighted update (2), as summarized in Alg. 1. One step of (8) costs $m+1$ matrix–vector products of size N against one for the linear model (3).

3.4. Prediction-Error Tube

We use an error tube only to analyze the prediction error of the bilinear rollout. The one-step residual of (4) is $r(x,u) := \Psi(F(x,u)) - M(u)\Psi(x) - B_0 u$. Along a sampled sequence $V = (v_0, \ldots, v_{T-1})$, the true lifted state evolves as $z_{t+1} = M(v_t) z_t + B_0 v_t + r_t$ with $r_t = r(x_t, v_t)$. Subtracting (8) shows that the deviation $\xi_t := z_t - \hat{z}_t$ obeys $\xi_{t+1} = M(v_t)\xi_t + r_t$. Suppose that

$$\|r(x,u)\| \;\leq\; c_x \|\Psi(x)\| + c_u \|u\| \tag{9}$$

on $\mathcal{X} \times \mathbb{U}$, with $c_x, c_u \geq 0$. Certified Koopman identification provides such proportional error bounds for a dictionary shifted to vanish at the equilibrium. They hold deterministically for kernel EDMD and, given a bound

Table 1. Settings shared by the linear and bilinear models.

System	Observables $\boldsymbol{b}$	N	Windows$\times H$	Epochs	K	T	Δt (s)	λ	$\boldsymbol{\sigma}$	$\mathbb{U}$	P_c
Unicycle	$(p_x, p_y, \sin\theta, \cos\theta - 1)$	4+4	8000×15	200	1536	40	0.05	0.4	$(0.6, 0.9)$	$\lvert u_i \rvert \le 1.5$	identity
Drone	$(\boldsymbol{p}, \sin\psi, \cos\psi)$	5+8	4000×15	200	1024	15	0.05	0.25	$(0.4, 0.4, 0.4, 0.7)$	$\mathbb{R}^4$	identity
Pendulum	$(\sin\theta, \cos\theta - 1, \dot{\theta})$	3+5	$20{,}000 \times 15$	200	2560	65	0.05	0.8	1.8	$\lvert u \rvert \le 2$	projection
Quadrotor	$(\boldsymbol{p}, \dot{\boldsymbol{p}}, \mathrm{vec}\,\boldsymbol{R}, \boldsymbol{\omega})$	18+6	6500×20	320	1500	20	0.02	0.5	$(1, 0.04, 0.04, 0.04)$	thrust in $[0, 20]$ N	projection

on the projection error, with a prescribed probability for SafEDMD [18, 21]. We know of no such certificate for a learned deep dictionary. With the precomputable norm bound $\mu(\boldsymbol{v}) := \|\boldsymbol{A}\| + \sum_{i=1}^{m} |v_i| \, \|\boldsymbol{B}_i\| \;\ge\; \|\boldsymbol{M}(\boldsymbol{v})\|$, define the tube

$$e_{t+1} = \big(\mu(\boldsymbol{v}_t) + c_x\big)\, e_t + c_x \|\hat{\boldsymbol{z}}_t\| + c_u \|\boldsymbol{v}_t\|, \quad e_0 = 0. \tag{10}$$

Proposition 2 (Tube bound)**:** Let $(\boldsymbol{x}_t, \boldsymbol{v}_t)$ stay for all $t < T$ in the set on which (9) holds. Then $\|\boldsymbol{\xi}_t\| \le e_t$, and hence $\|\boldsymbol{C}\hat{\boldsymbol{z}}_t - \boldsymbol{b}(\boldsymbol{x}_t)\| = \|\boldsymbol{C}\boldsymbol{\xi}_t\| \le \|\boldsymbol{C}\|\, e_t$, for all $t \le T$.

Proof: Induction over t, from the base case $e_0 = 0 = \|\boldsymbol{\xi}_0\|$. Given $\|\boldsymbol{\xi}_t\| \le e_t$, the deviation dynamics and (9) give $\|\boldsymbol{\xi}_{t+1}\| \le \|\boldsymbol{M}(\boldsymbol{v}_t)\|\, e_t + c_x \|\boldsymbol{z}_t\| + c_u \|\boldsymbol{v}_t\|$. Substituting $\|\boldsymbol{z}_t\| \le \|\hat{\boldsymbol{z}}_t\| + e_t$ and $\|\boldsymbol{M}(\boldsymbol{v}_t)\| \le \mu(\boldsymbol{v}_t)$, with all coefficients nonnegative, yields (10). □

The tube can enter the candidate cost as a penalty $\beta \sum_t e_t$ with weight $\beta \ge 0$. With fitted constants (Sec. 4.4) and without input conditioning (Sec. 4.5), no tested penalty improves either quadrotor controller, so all closed-loop results use the plain cost.

Remark 4 (Stability)**:** Practical stability has been established for EDMD-based MPC with proportional error bounds [18, 22]. Transferring such guarantees to MPPI would also require conditions on the sampling update, such as the free-energy decrease shown for MPPI with exact expectations in [23]. We do not pursue this transfer.

3.5. Identification

We train the linear and bilinear models independently with the same architecture, data, loss, and training schedule. The lifted state is $\boldsymbol{z} = \boldsymbol{\Psi}(\boldsymbol{x}) = (\boldsymbol{b}(\boldsymbol{x}), \boldsymbol{\zeta}(\boldsymbol{b}(\boldsymbol{x})))$ with a neural network $\boldsymbol{\zeta}$, so $N = \dim \boldsymbol{b} + \dim \boldsymbol{\zeta}$. For the systems of Table 1, $\boldsymbol{\zeta}$ has two hidden tanh layers of 64 units. We minimize a *multi-step* loss over length-H windows,

$$\mathcal{L} = \tfrac{1}{H} \textstyle\sum_{t=1}^{H} \big(\mathrm{ms}(\boldsymbol{C}\hat{\boldsymbol{z}}_t - \boldsymbol{b}(\boldsymbol{x}_t)) + \gamma\, \mathrm{ms}(\hat{\boldsymbol{z}}_t - \mathrm{sg}[\boldsymbol{z}_t])\big), \tag{11}$$

where $\hat{\boldsymbol{z}}_t$ is rolled from $\hat{\boldsymbol{z}}_0 = \boldsymbol{\Psi}(\boldsymbol{x}_0)$ through (3) or (4), $\mathrm{ms}(\cdot)$ is the mean squared entry over the batch, and sg stops the gradient. The second term encourages latent consistency. We use Adam (learning rate 10^{-3}, batch size 512), set $\gamma = 0.1$, and add weight decay 10^{-4} for the drone and the closed-loop quadrotor models. We initialize the entries of $\boldsymbol{A} - \boldsymbol{I}$, of $\boldsymbol{B}$ or $\boldsymbol{B}_0$, and of $\boldsymbol{B}_i$ as 0.01 times standard normal samples, except that the $\boldsymbol{B}_i$ start at zero for the drone, quadrotor, and torque-driven arm. Unless stated otherwise, training windows start from uniformly drawn states with any heading and $|p_x|, |p_y| \le 2$ m (unicycle), any yaw and $|p_i| \le 1$ m (drone), or any angle and $|\dot{\theta}| \le 8$ rad/s (pendulum). Closed-loop quadrotor windows start with tilts up to 0.6 rad, $|\dot{p}_i| \le 1.2$ m/s, and $|\omega_i| \le 1.2$ rad/s. The inputs are uniform in $\mathbb{U}$ (Table 1) at each step (unicycle, pendulum) or carry a random bias per window so that the heading or attitude varies within a window. The biased inputs stay within ± 1.1 m/s and ± 1.8 rad/s (drone) or within 3.2 N of the hover thrust and ± 0.05 N m (closed-loop quadrotor).

4. EXPERIMENTS

This section compares linear and bilinear rollouts in simulated tasks, both where the criterion can bar the linear one and where it does not. It also evaluates alternative rollout classes and measures what the bilinear term costs.

Systems and controllers. The unicycle and drone use standard kinematic models, and the pendulum uses an analytical model from [7]. The quadrotor and torque-driven arm are simulated in MuJoCo [24]. The arm is evaluated in prediction only. In closed loop, the *linear* controller uses (3), the form of [7, 12], and the *bilinear* controller uses (4). The *oracle* rolls out the true dynamics. All three controllers use $q = \|\boldsymbol{b}(\boldsymbol{x}_{t+1}) - \boldsymbol{b}_\mathrm{r}\|^2_{\boldsymbol{Q}} + r\|\tilde{\boldsymbol{u}}_t\|^2$ and $\phi = \|\boldsymbol{b}(\boldsymbol{x}_T) - \boldsymbol{b}_\mathrm{r}\|^2_{\boldsymbol{Q}_T}$ on the predicted observables, with the goal or reference $\boldsymbol{b}_\mathrm{r}$. Here $\tilde{\boldsymbol{u}}_t = \boldsymbol{u}_t - \bar{\boldsymbol{u}}$, with $\bar{\boldsymbol{u}}$ the hover input for the quadrotor and zero otherwise. With entries ordered as in $\boldsymbol{b}$ (Table 1), the unicycle uses $\boldsymbol{Q} = \mathrm{diag}(1, 1, 0.3, 0.3)$, $\boldsymbol{Q}_T = \mathrm{diag}(30, 30, 8, 8)$, and $r = 0.05$. Its stage cost also includes $0.4\|\boldsymbol{u}_t - \boldsymbol{u}_{t-1}\|^2$. The drone uses $\boldsymbol{Q} = \mathrm{diag}(1, 1, 1, 0.3, 0)$, $\boldsymbol{Q}_T = \boldsymbol{0}$, and $r = 0.02$, with the predicted $(\sin\psi, \cos\psi)$ scaled to unit length. The pendulum uses $\boldsymbol{Q} = \mathrm{diag}(1, 1, 0.04)$, $\boldsymbol{Q}_T = \mathrm{diag}(20, 20, 1)$, and $r = 0.002$. The quadrotor weights $\boldsymbol{p}$, $\dot{\boldsymbol{p}}$, $\boldsymbol{R}\boldsymbol{e}_3$ (goal $\boldsymbol{e}_3$), and $\boldsymbol{\omega}$ by 1, 0.2, 3, and 0.08 in $\boldsymbol{Q}$, and $\boldsymbol{p}$ and $\boldsymbol{R}\boldsymbol{e}_3$ by 8 in $\boldsymbol{Q}_T$, with $r = 0.01$ and $\tilde{\boldsymbol{u}}_t$ taken before P_c.

Parameter selection. Both learned controllers share the settings of Table 1 and the noise realizations, and no parameter was tuned separately for either learned model. During development, the MPPI parameters of the unicycle and the quadrotor, the yaw weight in Sec. 4.1, and the pendulum balance mode in Sec. 4.3 were adjusted until the oracle solved the task reliably. The other drone settings were

Table 2. Comparison of rollout classes. Success rates in %, with 95% intervals in parentheses. Open-loop errors at 40 (unicycle) and 15 (drone) steps. MAC is the number of multiply–accumulate operations per step.

	Unicycle			Drone		
Rollout model	Success (unclamped)	Success (clamped)	Open-loop error (m)	Tracking error (m)	Open-loop error (m)	MAC (uni./drone)
Linear, N=8 / 13	0.8 (0.0–2.0)	1.3 (0.0–3.3)	0.223	1.40 ± 0.89	0.409	80 / 221
Linear, N=29 (drone)	—	—	—	1.38 ± 0.95	0.409	— / 957
Linear + decoder (γ=0)	22.4 (18.0–26.8)	53.3 (44.0–62.7)	0.046	0.049 ± 0.011	0.043	4944 / 5469
Linear + decoder (γ=0.1)	15.6 (11.6–20.0)	58.0 (48.0–68.0)	0.049	training collapsed (9 of 10 runs)		4944 / 5469
Frozen bilinear (convexified)	28.8 (20.8–37.6)	28.7 (18.0–40.0)	0.037	0.048 ± 0.011	0.086	80 / 221
Neural dynamics	53.6 (47.6–59.6)	100	0.014	0.046 ± 0.009	0.025	17,408 / 17,792
Bilinear	90.4 (86.4–94.0)	100	0.0069	0.046 ± 0.009	0.0030	208 / 897
Oracle (true model)	90.8 (87.2–94.0)	100	0	0.046 ± 0.009	0	—

Table 3. Graded drone study. Open-loop position errors (m) at 15 steps and linear successes at the 0.15 m tolerance. LS is constant-gain least squares.

$\bar{\psi}$ (rad)	$\hat{v}$	Lin. error	LS error	Bil. error	Lin. success
0	0	0.0014	0.0014	0.0010	15/15
0.2	0.40	0.041	0.041	0.0015	15/15
0.5	0.96	0.098	0.098	0.0054	15/15
0.8	1.43	0.153	0.155	0.0023	5/15
1.2	1.86	0.221	0.223	0.0013	0/15
$\pi/2$	2	0.279	0.281	0.0010	0/15
π	2	0.409	0.411	0.0031	0/15

chosen once and never changed, and the oracle solves the task with them. The pendulum MPPI parameters of Table 1, however, were adjusted in runs with all three controllers, so Sec. 4.3 tests 16 settings. For the closed-loop studies, the lifted dimension, network width, window length, and number of epochs were fixed beforehand. The effect of the first two is shown by the enlarged drone lift (Sec. 4.2) and the pendulum width sweep (Sec. 4.3).

Statistics. We repeated each unicycle trial with five MPPI noise realizations, for 250 trials per controller. Across the five realizations, even the oracle succeeds in 40 to 49 of the 50 trials. We therefore report pooled success rates with 95% bootstrap intervals that resample the 50 pairs of start pose and trained model. Tracking errors are mean $\pm$ standard deviation, and open-loop errors in Tables 2 and 3 are means over training runs.

4.1. Graded Gain Variation

The drone is commanded by body-frame velocities and a yaw rate, so $\dot{\boldsymbol{p}} = \boldsymbol{R}(\psi)\boldsymbol{v}^{\mathrm{b}}$. It tracks three references (figure-8, square, waypoints) while its yaw follows $\psi_{\mathrm{r}}(t) = \bar{\psi}\sin(0.2\pi t)$, which gives $\hat{v} = 2\sin\bar{\psi}$ over the reference range for $\bar{\psi} \le \pi/2$ (Remark 3). The cost here also weights $\cos\psi$ ($\boldsymbol{Q} = \mathrm{diag}(1,1,1,0.3,0.3)$). The training data keep $|\psi| \le \bar{\psi} + 0.05$ rad. For each $\bar{\psi}$, five linear and five bilinear models were trained.

The linear open-loop error grows with the yaw range (Table 3), whereas the bilinear error stays below 0.006 m. The linear models reach the least-squares floor (Remark 1), although the per-window input bias meets the independence assumption only approximately. Their horizontal position gain divided by Δt matches the least-squares value $\mathbb{E}[\cos\psi]$ of the training data to within 0.02 in every training run. Their error is within 2% of that of a constant-gain least-squares predictor with a finite drift basis. The degradation is thus inherent to the constant input gain rather than to the training.

In closed loop, the bilinear controller's mean tracking error is within 0.006 m of the oracle's in every episode, and both meet a 0.15 m tolerance. The linear controller meets it in all episodes up to $\hat{v} = 0.96$, only on the waypoint reference at $\hat{v} = 1.43$, and in none from $\hat{v} = 1.86$. Every training run gives the same outcome. The transition depends on the tracking tolerance, as all linear episodes succeed up to $\hat{v} = 0.40$, 0.96, and 1.43 for tolerances of 0.10, 0.15, and 0.25 m, respectively. The criterion (Proposition 1) bounds the one-step input-response error, not the tracking error, and therefore does not predict these levels.

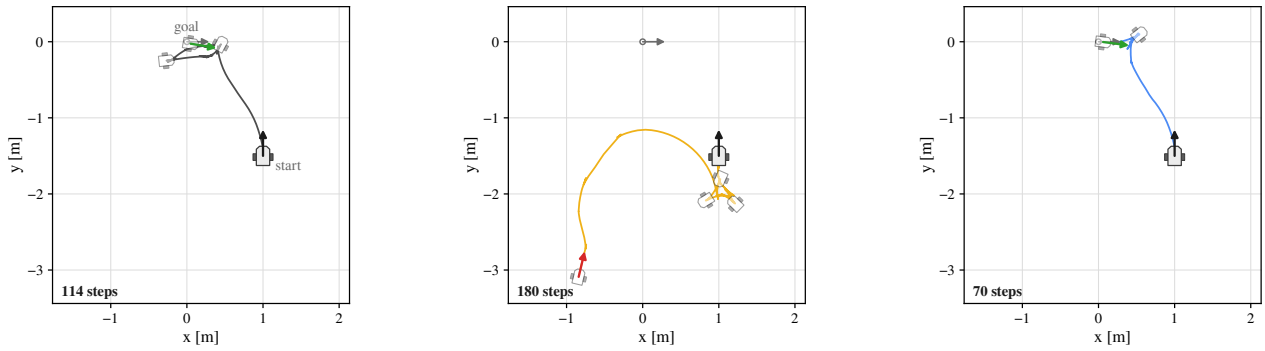
4.2. Large Gain Variation

When $\hat{v}$ is large, as in the following unicycle and drone tasks, the criterion can bar the linear model.

Unicycle. The unicycle drives from ten start poses, whose headings cover the full circle, to a goal at the origin. A trial ends once the position is within 0.08 m of the goal and the heading error is below 0.2 rad, or after 180 steps. The forward-speed input affects position through the heading rotation $(\cos\theta, \sin\theta)$, so $\hat{v} = 2$. In open loop, the bilinear model predicts position over thirty times more accurately than the linear at the planning horizon (Table 2). With a fixed trigonometric dictionary instead of a learned one, the factor is roughly forty.

In closed loop, a trial succeeds if the final position is within 0.3 m of the goal. As Fig. 2 illustrates, the bilinear controller reaches the goal in 90.4% of trials, the oracle in 90.8%, and the linear controller in 0.8% (Table 2). The linear controller ends a median 2.8 m from the goal. With a heading error below 0.2 rad also required, the rates are 77.2% (bilinear) and 76.0% (oracle), and at the termination tolerance, 66.4% and 68.0%. For all three measures, the two differ by at most 1.6 percentage points.

Drone. The 3D drone tracks the three references with



(a) (b) (c)

Fig. 2. Unicycle trials from one start pose under (a) the oracle, (b) linear, and (c) bilinear Koopman MPPI.

a yaw reference along the path tangent. The cost penalizes the yaw only through its sine. The oracle's yaw spans a range 95–142° wide, depending on the reference, so $\hat{\nu} = 1.5$–1.9 (2 for the yaw reference itself). In open loop, the bilinear model predicts world position over a hundred times more accurately at the planning horizon (Table 2).

In closed loop, the bilinear controller matches the oracle's mean tracking error, 0.046 ± 0.009 m over 60 episodes (ten training runs, three references, and planner-state noise of standard deviation 0 or 0.02). The linear controller drifts from every path (1.40 ± 0.89 m).

Alternative rollout classes. Table 2 adds four rollout classes with the same data, training budget, and MPPI, to test whether they also come close to the oracle. The first two are a linear lift with N=29 instead of 13 (drone) and a linear latent model with a nonlinear decoder as in [6]. The other two are the bilinear model with its input matrix frozen over the horizon at the current state (the convexification of [9, 11]) and a neural dynamics model without a Koopman lift [1]. The decoder has two tanh layers of 64 units, and the neural model, $\boldsymbol{b}^{+} = \boldsymbol{b} + \Delta t\, \boldsymbol{f}_{\mathrm{nn}}(\cdot)$, has two tanh layers of 128 units.

The enlarged linear lift does not reduce the error, as predicted (Proposition 1, Remark 1). Every drone rollout with a state-dependent input gain comes within 7% of the oracle, except a decoder whose training collapsed. On the unicycle, whose heading changes quickly within a horizon, the frozen and decoder models reach the goal in only 16–29% of unclamped trials. For the frozen model, this is consistent with the first statement of Corollary 1 applied within each horizon, whereas that bound does not cover the decoder (Remark 2). The neural model succeeds in 53.6% of unclamped trials. With candidates clamped to the training input range and the update computed from the clamped perturbations, it succeeds in all 150 trials (three noise realizations), as do the bilinear model and the oracle.

The neural model, however, requires one to two orders of magnitude more multiply–accumulate operations per step than the bilinear model. On the unicycle, the bilinear rollout is thus the cheapest tested model that comes close to the oracle. On the drone, whose heading changes slowly within a horizon, the frozen model is the cheapest model close to the oracle, at a quarter of the bilinear model's operations.

4.3. Zero or Small First-Order Gain Variation

When $\boldsymbol{\Phi} \equiv \mathbf{0}$ or $\hat{\nu}$ is small, the criterion does not bar the linear model, which can then suffice. In the graded study at small $\hat{\nu}$ (Sec. 4.1), both learned controllers meet the tracking tolerance, although the linear open-loop error is an order of magnitude larger for $\hat{\nu} > 0$. On the torque-driven arm, the linear joint-angle errors also stay small. On the quadrotor, however, the linear controller stays at the goal less often than the bilinear one. The pendulum is a counterexample to sufficiency, as linear models of matching width fail at long swing-up horizons (Remark 2).

Torque-driven arm. The three-link arm moves in a horizontal plane (joint damping 1 N m s/rad, torques up to 0.25 N m, $\Delta t = 0.05$ s). Its joint angles and end effector have relative degree two, so $\boldsymbol{\Phi} \equiv \mathbf{0}$, and we apply Lemma 1 to the exact one-step gain $\partial \boldsymbol{y}_{t+1} / \partial \boldsymbol{u}_t$ (Remark 2). We widened the collision-free joint range around the posture $(0, 45°, 65°)$ in five levels from ± 0.1 to ± 2 rad. For each range, we trained five models of each class on 6000 windows of 12 steps. Their dictionaries have 12 learned features and two hidden layers of 96 units. We standardized the observables with the training statistics and trained with weight decay 10^{-4}, batch 256, and 160 epochs. We sampled the exact one-step gain by central differences (10^{-4} N m) at 1500 test states and inputs. Its normalized variation rises from 0.59 to 1.98 for the end effector but only from 0.06 to 0.56 for the joint angles. The 12-step end-effector input-response error of the linear model grows from 8% to 96% of the response, roughly three to ten times the bilinear error. For the joint angles, the errors span 1.0–3.7% (linear) and 0.4–7.2% (bilinear)

of the response, and the bilinear error is smaller except at the widest range.

Quadrotor. The quadrotor flies from hover to four goals. Thrust and torques reach the position only through acceleration, so the criterion is silent for the position. The velocity, which the cost also reads, receives the thrust along the attitude-dependent direction $\boldsymbol{R}\boldsymbol{e}_3$, whose variation is small near hover ($\hat{\nu} = 0.27$ over the oracle's states). For both models, we conditioned the inputs (Sec. 3.2), centering them at hover (9.81 N thrust, zero torque) and dividing the torques by 0.05 N m. We trained 15 models of each class on five data sets and tested each on the four goals. A trial succeeds if the quadrotor stays within 0.12 m of the goal over the last 2 s of a 10 s flight. The bilinear controller succeeds in 54 of these 60 trials, the linear controller in 20, and the oracle in 20 of 20. The bilinear model succeeds more often for all five data sets (sign-flip test, $p = 0.06$).

We do not attribute the closed-loop advantage to a state-dependent input gain. Both models learn nearly the same thrust gain at hover. The bilinear model has 2304 more parameters, but we did not test a parameter-matched linear model on this system. On the drone, enlarging the linear model does not close the gap (Sec. 4.2), and on the pendulum it only narrows the gap.

In open loop, which quadrotor model predicts better depends on the output channel and the horizon. Separate open-loop models were trained on 12-step windows, so predictions beyond 12 steps extrapolate. The windows, which start at tilts up to 69°, give $\hat{\nu} \approx 2$ for the velocity. At every horizon from 1 to 20 steps, the linear velocity error exceeds the bilinear one by up to about a factor of two. The linear position error is smaller up to eight steps and larger from nine steps on. With unit-normalized channels, the linear full-state error stays within 15% of the bilinear one. The criterion does not bound these total prediction errors.

Pendulum. In torque-limited pendulum swing-up and hold, the torque reaches the angle at second order and the angular velocity through a constant gain, so $\hat{\nu} = 0$. With the MPPI of Table 1, even the oracle swings the pendulum up but does not keep it upright. We therefore added a balance mode that takes over near upright ($1 - \cos\theta < 0.1$ and $|\dot{\theta}| < 1.5$ rad/s) and hands back when $1 - \cos\theta > 0.3$. It uses $K = 2560$, a horizon of 10 or 15 steps, $\lambda = 0.1$, $\sigma = 0.6$ or 1.0, and a nominal input clamped to $|u| \leq 2$. Using the oracle alone, we selected balance-mode parameters for each of 16 swing-up settings. They combine horizons of 15, 20, 30, 45, or 65 steps, two noise levels, and a clamped or (from 30 steps) free nominal input. A trial succeeds if the pendulum stays upright ($1 - \cos\theta < 0.05$) over the last 5 s of a 15 s episode. A trained model succeeds in a setting if at least 6 of its 10 trials do, and a model class fails if at most one of its five models succeeds.

The oracle succeeds in all trials of every setting. With the observables of Table 1 and width 64, at least three of five bilinear models succeed in 14 settings, whereas the linear models fail in all 16. With width 256, three of five linear models succeed in the four settings with a 65-step swing-up horizon and at most one elsewhere. In open loop, we trained five models per width and class on 8000 windows of 12 steps for 110 epochs. At every width from 32 to 256, the mean 12-step observable error of the linear models exceeds that of the bilinear models by more than 30%.

To locate the failure, we exchanged the zero-input prediction and the input response (Corollary 1) between the models. In each of the 14 settings where the bilinear models succeed, at least four of five width-64 linear models succeed with their input response replaced by the bilinear or exact one. If only their zero-input prediction is replaced in the same way, at most one succeeds. The failure thus lies in the input response of the linear models.

On the pendulum, the variation of the exact one-step gain depends on the output. For $(\sin\theta, \cos\theta - 1)$, this gain turns with the angle, so its normalized variation is 2. For the angle itself, which the linear models of [7] predict, the gain is constant. We therefore repeated the 16 settings with models that predict the angle ($\boldsymbol{b} = (\theta, \dot{\theta})$, measured θ wrapped to $[-\pi, \pi)$, $N = 2{+}5$, other entries as in Table 1, cost on the sine and cosine of the predicted angle). All five linear models then succeed at a 15-step swing-up horizon, where those with sine and cosine outputs fail.

At 20 steps, however, at most two of five linear models succeed, and from 30 steps on, they fail in every setting. Bilinear models with the same outputs also fail at 65 steps. Yet at least four of five succeed at 20 steps and in the four clamped settings with 30 or 45 steps. Hence neither $\hat{\nu} = 0$ nor a constant one-step gain of the output ensures that a linear model suffices.

4.4. Computational Cost and Prediction-Error Tube

Cost. We timed the MPPI update on one CPU thread (Intel Core i5-13400F, median of 200 calls) with the settings of Table 1, without the balance mode of the pendulum. One bilinear update takes 4.9, 2.4, 5.8, and 6.4 ms for the unicycle, drone, pendulum, and quadrotor, against 3.5, 1.3, 4.1, and 3.6 ms for the linear one. The bilinear update thus takes less than twice as long and uses 5–32% of the control period. The rollout of all K candidates alone takes less than five times as long with the bilinear model. This ratio stays within 20% of the multiply–accumulate ratio, which grows with m. The neural model of Table 2 takes roughly an order of magnitude longer per rollout and over four times longer per update than the bilinear model.

Prediction-error tube. For the four closed-loop systems, we fitted c_x and c_u in (9) as the smallest envelope of the one-step training residuals (a linear program). On held-out data from the training distribution, inequality (9) is violated in at most 0.01% of the samples per system and in at most 0.03% per model. Violations are much more

frequent near the equilibrium, where our dictionaries do not vanish, and for small inputs, which are rare in the data. Along the unicycle MPPI candidates, which leave the training input range, the one-step inequality is violated in 15.8% of the steps and the tube in 0.94%. For candidates clamped to that range, the rates are 0.40% and 0.005%. The fitted inequality is therefore an empirical envelope of the training distribution, not a bound on $\mathcal{X} \times \mathbb{U}$.

The tube is conservative. Over 2000 held-out rollouts of one unicycle model, it exceeds the true error by a median factor of about five at the first step and about a thousand at step 40. With least-squares constants instead of the envelope, the tube is violated at the first step in 804 of the rollouts.

4.5. Ablations

Input conditioning. Without conditioning, the bilinear products of the quadrotor models receive the raw thrust (about 9.81 N), and the linear model learns roughly a quarter of the physical torque gain for roll and pitch. The bilinear and linear controllers then succeed in only 1 and 4 of 60 trials. The raw thrust also makes the bilinear model sensitive to the initialization of $\boldsymbol{B}_i$. With a small random initialization, its 12-step open-loop error (Sec. 4.3) is 2.9 times that of a zero-initialized model, and all 20 closed-loop trials stray more than 2 m from the goal. With inputs centered at the hover thrust, the two initializations give the same error (ratio 1.005), and no trial strays that far.

Multi-step training. A one-step loss instead of (11) multiplies the error at the training horizon by 2.1 (linear) and 3.4 (bilinear) for the pendulum and by 1.3 and 1.1 for the quadrotor. It does not degrade the unicycle models. The linear-to-bilinear gap then persists for the unicycle and grows for the quadrotor, but vanishes for the pendulum (ratio 0.88 against 1.43). The open-loop pendulum gap therefore depends on the training loss.

5. CONCLUSION

This paper presented a necessary condition for a constant input matrix in Koopman rollouts of sampling-based control. With a linear readout, such a matrix leaves a worst-case gain error bounded below in proportion to the input-gain variation over the visited states. In the graded drone study, linear rollouts begin to fail at a tolerance-dependent level of this variation, which the bound does not predict. Above this level, the bilinear term brings the rollout near the oracle, at most doubling the update time. At small variation, the bilinear rollout is usually, but not always, more accurate, while the cheaper linear one can also meet the tolerance. Yet on our pendulum, with zero first-order gain variation, linear models as wide as the bilinear ones fail at long swing-up horizons, even for an output with a constant one-step gain. Since the condition is only necessary, a linear rollout that it does not bar still needs a closed-loop check to replace a bilinear one.

Several limitations remain. The bound assumes smooth dynamics, so contact-rich or hybrid systems are outside its scope. All results are from simulation, the unicycle and drone use kinematic models, and we computed the gain variation from known models or simulators. The next steps are hardware validation with unmodeled actuation and sensing, and data-based gain estimates.

DECLARATIONS

Conflict of Interest

The authors declare that there is no competing financial interest or personal relationship that could have appeared to influence the work reported in this paper.

Authors' Contributions

Kangmin Lee contributed to the implementation, conducted the experiments, and participated in writing the manuscript. Sanghyun Kim led the research direction and contributed to the writing of the manuscript.

Funding

This work was supported by the Korea Planning & Evaluation Institute of Industrial Technology (KEIT) and the Ministry of Trade, Industry & Energy (MOTIE) of the Republic of Korea (No. RS-2025-11082970), by the Industrial Strategic Technology Development Program funded by MOTIE, Korea (No. RS-2025-25449039), and by the Korea Basic Science Institute (National Research Facilities and Equipment Center) grant funded by the Ministry of Science and ICT (No. RS-2025-00564593).

Data Availability

Hyperparameters and videos are also available at `https://rcilab.github.io/koopman-mppi`.

Use of Generative AI

During the preparation of this manuscript, the authors used Claude (Anthropic) to improve its structure and clarity. The authors subsequently reviewed and revised the content and take full responsibility for the final manuscript.

REFERENCES

[1] G. Williams, N. Wagener, B. Goldfain, P. Drews, J. M. Rehg, B. Boots, and E. A. Theodorou, "Information theoretic MPC for model-based reinforcement learning," in *Proc. IEEE Int. Conf. Robot. Autom. (ICRA)*, 2017, pp. 1714–1721.

[2] G. Williams, A. Aldrich, and E. A. Theodorou, "Model predictive path integral control: From theory to parallel com-

putation," *J. Guid. Control Dyn.*, vol. 40, no. 2, pp. 344–357, 2017.

[3] C. Pezzato, C. Salmi, E. Trevisan, M. Spahn, J. Alonso-Mora, and C. Hernández Corbato, "Sampling-based model predictive control leveraging parallelizable physics simulations," *IEEE Robot. Autom. Lett.*, vol. 10, no. 3, pp. 2750–2757, 2025.

[4] B. O. Koopman, "Hamiltonian systems and transformation in Hilbert space," *Proc. Natl. Acad. Sci.*, vol. 17, no. 5, pp. 315–318, 1931.

[5] M. O. Williams, I. G. Kevrekidis, and C. W. Rowley, "A data-driven approximation of the Koopman operator: Extending dynamic mode decomposition," *J. Nonlinear Sci.*, vol. 25, no. 6, pp. 1307–1346, 2015.

[6] B. Lusch, J. N. Kutz, and S. L. Brunton, "Deep learning for universal linear embeddings of nonlinear dynamics," *Nat. Commun.*, vol. 9, p. 4950, 2018.

[7] W. Hao, Y. Fang, Z. Lu, and S. Mou, "Accelerating sampling-based control via learned linear Koopman dynamics," in *Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst. (IROS)*, 2026, pp. 11 780–11 785.

[8] M. Korda and I. Mezić, "Linear predictors for nonlinear dynamical systems: Koopman operator meets model predictive control," *Automatica*, vol. 93, pp. 149–160, 2018.

[9] D. Bruder, X. Fu, and R. Vasudevan, "Advantages of bilinear Koopman realizations for the modeling and control of systems with unknown dynamics," *IEEE Robot. Autom. Lett.*, vol. 6, no. 3, pp. 4369–4376, 2021.

[10] C. Folkestad and J. W. Burdick, "Koopman NMPC: Koopman-based learning and nonlinear model predictive control of control-affine systems," in *Proc. IEEE Int. Conf. Robot. Autom. (ICRA)*, 2021, pp. 7350–7356.

[11] D. Zhao, B. Li, F. Lu, J. She, and S. Yan, "Deep bilinear Koopman model predictive control for nonlinear dynamical systems," *IEEE Trans. Ind. Electron.*, vol. 71, no. 12, pp. 16 077–16 086, 2024.

[12] F. Meng, Z. Yang, X. Mao, H. Liang, and M. Q.-H. Meng, "Provably safe trajectory generation for manipulators under motion and environmental uncertainties," arXiv:2603.09083v1, 2026.

[13] L. C. Iacob, R. Tóth, and M. Schoukens, "Koopman form of nonlinear systems with inputs," *Automatica*, vol. 162, p. 111525, 2024.

[14] L. Bold, M. Rosenfelder, H. Eschmann, H. Ebel, and K. Worthmann, "On Koopman-based surrogate models for non-holonomic robots," *IFAC-PapersOnLine*, vol. 58, no. 21, pp. 55–60, 2024.

[15] M. Rosenfelder, L. Bold, H. Eschmann, P. Eberhard, K. Worthmann, and H. Ebel, "Data-driven predictive control of nonholonomic robots based on a bilinear Koopman realization: Data does not replace geometry," *Robot. Auton. Syst.*, vol. 194, p. 105156, 2025.

[16] D.-H. Nam and H.-G. Kim, "MPPI-based integrated task assignment, path planning, and control for unmanned ground vehicles," *Int. J. Control Autom. Syst.*, vol. 22, no. 12, pp. 3641–3652, 2024.

[17] S. Lee and S. Kim, "Manifold-constrained MPPI: Real-time sampling-based control for nonlinear equality-constrained robotic systems," arXiv:2605.24813, 2026.

[18] R. Strässer, K. Worthmann, I. Mezić, J. Berberich, M. Schaller, and F. Allgöwer, "An overview of Koopman-based control: From error bounds to closed-loop guarantees," *Annu. Rev. Control*, vol. 61, p. 101035, 2026.

[19] J. S. Kim, Y. S. Quan, and C. C. Chung, "Koopman operator-based model identification and control for automated driving vehicle," *Int. J. Control Autom. Syst.*, vol. 21, no. 8, pp. 2431–2443, 2023.

[20] L. Han, K. Peng, W. Chen, and Z. Liu, "A data-driven Koopman modeling framework with application to soft robots," *Int. J. Control Autom. Syst.*, vol. 23, no. 1, pp. 249–261, 2025.

[21] R. Strässer, M. Schaller, K. Worthmann, J. Berberich, and F. Allgöwer, "SafEDMD: A Koopman-based data-driven controller design framework for nonlinear dynamical systems," *Automatica*, vol. 185, p. 112732, 2026.

[22] L. Bold, L. Grüne, M. Schaller, and K. Worthmann, "Data-driven MPC with stability guarantees using extended dynamic mode decomposition," *IEEE Trans. Autom. Control*, vol. 70, no. 1, pp. 534–541, 2025.

[23] M. Fazlyab, S. Sharifi, and J. Wang, "Model predictive path integral control as preconditioned gradient descent," *IEEE Control Syst. Lett.*, vol. 10, pp. 2089–2094, 2026.

[24] E. Todorov, T. Erez, and Y. Tassa, "MuJoCo: A physics engine for model-based control," in *Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst. (IROS)*, 2012, pp. 5026–5033.



Kangmin Lee received his B.S. degree in Mechanical Engineering from Kyung Hee University, South Korea, in 2026, where he is currently pursuing the Ph.D. degree. His research interests include optimal control, model predictive control, and reinforcement learning.



Sanghyun Kim received the B.S. and Ph.D. degrees from Seoul National University, South Korea, in 2012 and 2020, respectively. From 2020 to 2023, he was a Senior Researcher at the Korea Institute of Machinery and Materials (KIMM), South Korea. He is currently an Assistant Professor in the Department of Mechanical Engineering at Kyung Hee University. His research interests include control and planning for redundant robotic systems.

Publisher's Note Springer Nature remains neutral with regard to jurisdictional claims in published maps and institutional affiliations.