Task-Driven Hybrid Model Reduction for Dexterous Manipulation

Wanxin Jin and Michael Posa

***Abstract*—In contact-rich tasks, like dexterous manipulation, the hybrid nature of making and breaking contact creates challenges for model representation and control. For example, choosing and sequencing contact locations for in-hand manipulation, where there are thousands of potential hybrid modes, is not generally tractable. In this paper, we are inspired by the observation that far fewer modes are actually necessary to accomplish many tasks. Building on our prior work learning hybrid models, represented as linear complementarity systems, we find a reduced-order hybrid model requiring only a limited number of task-relevant modes. This simplified representation, in combination with model predictive control, enables real-time control yet is sufficient for achieving high performance. We demonstrate the proposed method first on synthetic hybrid systems, reducing the mode count by multiple orders of magnitude while achieving task performance loss of less than 5%. We also apply the proposed method to a three-fingered robotic hand manipulating a previously unknown object. With no prior knowledge, we achieve state-of-the-art closed-loop performance within a few minutes of online learning, by collecting only a few thousand environment samples.**

***Index Terms*—hybrid control systems, model reduction, dexterous manipulation, model-based reinforcement learning, model predictive control (MPC).**

I. Introduction

Many robotic tasks, like legged locomotion or dexterous manipulation, involve a robot frequently making and breaking contact with the physical environment or/and objects. The rich-contact behavior makes the robotic system multi-modular and hybrid, characterized by a set of discrete contact modes and continuous physical dynamics within each mode.

The hybrid nature of contact-rich robotic systems poses great challenges in their representation and control. For data-driven modeling, recent results have demonstrated that standard deep learning networks struggle to train on and represent stiffness and multi-modality [1], [2], motivating the learning frameworks explicitly designed to capture hybrid dynamics [3], [4]. Similar challenges exist in planning and control of contact-rich systems, where algorithms must jointly reason over a combinatoric number of discrete contact choices and continuous inputs of physical actuation. This process will quickly become intractable as the number of potential hybrid modes and planning depth grow. The above two aspects become even more critical for real-time closed-loop control of contact-rich robotic systems, where a compromise between computational tractability and task performance has to be made [5].

Towards a goal of real-time planning and control of contact-rich manipulation with tens of thousands of modes, we hypothesize that identifying and utilizing a full hybrid dynamics model is almost certainly unnecessary. Instead, one might ask:

Can a far simpler model, with only a few task-relevant hybrid modes, enable the high performance and real-time control for contact-rich manipulation?

Here, we propose to answer the question in the affirmative by building upon recent progress in hybrid representation learning [4] and real-time contact-rich planning and control [6], [7].

If one observes a multi-finger robot manipulating a cube for a reorientation task, the task-critical contact interactions might be dominated by a few modes: for example, all fingertips stick to the cube, or one fingertip pushing or sliding while others stick to the cube. While other modes might occur, they do so briefly or in a functionally similar manner to another mode. This observation inspires us to study the problem of learning task-driven reduced-order hybrid models. On the technical side, we have seen the recent progress in learning hybrid representations [3], [4]. Particularly in our prior work [4], we have developed an efficient method to learn a piecewise affine system, represented as a linear complementarity system (LCS) [detailed in Section III], with tens of thousands of hybrid modes. We also note recent progress towards fast control and planning of contact-rich robotic systems. For example, in [6], the authors approximate nonlinear contact-rich dynamics using LCS and develop LCS-based model predictive control, achieving real-time control performance for a reasonably-sized manipulation system.

Built on the above observation and the foundational prior work, this paper aims to answer the above question theoretically and algorithmically. Our goal is to find a reduced-order hybrid model, containing only a small number of task-relevant modes, which is sufficient for high-performance, real-time control of contact-rich manipulation tasks. We call the problem ‘task-driven hybrid model reduction’. The primary contributions of this work are:

(i) We study the problem of task-driven hybrid model reduction by formulating it as minimizing the task performance gap between model predictive control (MPC) with the reduced-order hybrid model and MPC with the full hybrid dynamics. We show that the reduced-order model learning with on-policy MPC data provably upper bounds the task performance gap, leading to a simple iterative method to improve the reduced-order model and MPC controller.

(ii) We make use of our prior work of learning LCS [4], and the recent development of real-time LCS-based control

Wanxin Jin is with the School for Engineering of Matter, Transport, and Energy at Arizona State University, Tempe, AZ 85287, and Michael Posa is with the General Robotics, Automation, Sensing and Perception (GRASP) Laboratory, University of Pennsylvania, Philadelphia, PA 19104, USA. Email: wanxin.jin@asu.edu, posa@seas.upenn.edu

on contact-rich systems, such as [6], to develop our practical learning algorithm. The algorithm runs a real-time closed-loop LCS model predictive controller on the complete hybrid dynamical system (environment), enabling improving the reduced-order model and its closed-loop control performance.

(iii) In the first example, we demonstrate the capabilities of the proposed method in reducing synthetic hybrid control systems. We show that the proposed method enables reducing the hybrid mode count by multiple orders of magnitude while achieving a task performance loss of less than 5%. In the second application, we use the proposed method to solve three-finger robotic hand manipulation for unknown object reorientation in simulation environment. With no prior knowledge, we achieve state-of-the-art closed-loop performance within a few minutes of online learning, by collecting only a few thousand environment samples.

The following article is organized as follows. The related work is reviewed in Section II. Section III gives preliminaries and formulates the problem of task-driven hybrid mode reduction. Section IV presents the theoretical analysis and Section V develops the algorithm. Section VI uses the proposed method to solve model reduction on synthetic hybrid systems. Section VII applies the proposed method to solve three-finger robotic hand manipulation. Conclusions are drawn in Section VIII.

II. Related Work

1) Learning Hybrid Dynamics Models: This work heavily leverages the recent results in learning multi-modal dynamics representations. Previous studies [1], [3] have shown that naive neural networks fail to capture the discontinuity and stiffness of physical systems. A prominent line of recent work focuses on learning smoothing approximations by relaxing the hybrid mode boundaries [8]–[12], though at the cost of some approximation error. Instead of using smoothing approximation, this paper considers learning explicit hybrid structures. We focus on a simple yet expressive representation for hybrid systems: continuous piecewise-affine (PWA) models. This is motivated by the fact that many physics simulation engines [13]–[16] locally use linear complementarity models (a compact form of continuous PWA [17]) to handle physical contact events at each simulation step. PWA models bring two benefits. First, they are a well-studied subject in the control community [17]–[19], which captures the multi-modality of a hybrid system, by approximating dynamics using polyhedral partitions with each assigned a mode-dependent linear model. Second, they can be tractably incorporated into planning and control for real-time performance due to recent progress in [6]. Although continuous PWA models cannot capture the discontinuities that arise from impulsive impact events, there is a large range of manipulation tasks in which such events are not prominent, thus we believe PWA models to be sufficient. Also, we noted that a PWA model, via stiffness, can well capture the discontinuity, as adopted by many physics simulation engines [13]–[16].

Identifying PWA models is NP-hard in general [20]. Most existing methods [21], [22] for PWA regression are clustering-based: they alternate data classification and model regression for each class. Those methods normally have a complexity that scales exponentially with the number of data points or hybrid modes. In this paper, learning PWA models is based on our recent work [4]. Specifically, we write a PWA model compactly as a linear complementarity system (LCS), via implicit parameterization [17], and propose an implicit violation-based loss that generalizes the physics-based method in ContactNets [3]. The method does not need explicitly cluster data and can handle tens of thousands of (potentially stiff) modes efficiently. Recent results have proven a superior generalization of this class of methods than explicit loss methods [2].

2) Fast Planning and Control on Multi-Contact Systems: The success of the proposed method also relies on the recent progress in real-time multi-contact planning and control. Planning and control on multi-contact systems are notoriously challenging, as the algorithms must decide when and where to make or break contacts, whose complexity scales exponentially to the number of potential contacts and planning horizon. Traditionally, [23], [24] use the predefined sequence of mode to achieve real-time control on legged locomotion [25] and manipulation [26]. To enable general-purpose fast multi-contact control, [6] and [7] consider the LCS linearization of nonlinear multi-contact robot dynamics. Specifically, in [7], the authors smooth the stiff complementarity constraint and then apply the interior-based method to approximate the solution sequentially. In a different way, [6] maintains the hybrid structures and proposes to decouple the combinatoric complexity from the planning depth and then use the alternating direction method of multipliers (ADMM) to solve the decoupled problem, which can be done in parallel for further acceleration.

In this paper, we include a real-time LCS model predictive controller as part of our learning algorithm for on-policy data collection and closed-loop control. We use a direct method of optimal control to formulate and solve the LCS MPC. This was first proposed in [27]. In our implementation, we utilize the state-of-the-art optimal control solver [28] for fast MPC.

3) Reinforcement Learning for Contact-Rich Manipulation: Reinforcement learning (RL) has achieved impressive results in contact-rich manipulation [29]–[31]. Some representative work includes [31], where in-hand manipulation policies are learned for object reorientation, and [30] for solving TriFinger Manipulation. However, both methods use model-free RL, requiring millions or even billions of environment samples and many hours or even days of training. To alleviate sample inefficiency, model-based RL has been used to robotic manipulation by first learning a dynamics model to aid policy search [29], though requiring a large amount of training data to fit an unstructured deep neural network. Furthermore, control with deep neural network models can be challenging. Commonly used shooting-based methods [32] have a complexity exponential to planning depth and system dimension [33].

In comparison with the work above, the emphasis of this paper is on highly data-efficient hybrid model learning, paired with real-time closed-loop control. Specifically, the tasks that might require hours of data for unstructured learning methods will be trained and completed in minutes.

4) Reduced-order Models for Multi-Contact Robotic Tasks: The idea of using a reduced-order model for hybrid robotic tasks has widely used in robot locomotion [34], [35] for real-time generating behavior plans. However, these reduced-order models are manually designed and may miss some key dynamics aspects of the full-order dynamics [36]. To address those challenges, recent results demonstrate the ability to optimize for a reduced-order model that retains the capabilities of the full-order robot dynamics [5]. In their paper, authors focus on reducing the state dimension needed for planning, while we focus here on the comparatively unexplored problem of hybrid mode reduction. Recently, model-free RL has been used to learn the unmodeled aspects of a reduced-order model to improve locomotion performance [36]. Our method differs from theirs in three aspects. First, their formulation does not *explicitly* encourage the reduction of the performance gap of the reduced-order model, while our formulation is to directly minimize the performance gap. Second, our method is not rooted in model-free RL, which can be data inefficient for our setting, where true dynamics is originally unknown (thus prohibiting sim-to-real transfer). Third, rather than using smooth approximations, we directly identify a hybrid representation.

III. Preliminaries and Problem Formulation

This section presents some preliminaries and formulates the problem of task-driven hybrid model reduction.

A. Hybrid Models for Multi-contact Dynamics

Consider the following generic hybrid system:

$$\begin{aligned} \boldsymbol{x}_{t+1} = \boldsymbol{f}_i(\boldsymbol{x}_t, \boldsymbol{u}_t) \quad \text{with} \quad (\boldsymbol{x}_t, \boldsymbol{u}_t) \in \mathcal{P}_i, \\ \mathcal{P}_i = \{(\boldsymbol{x}, \boldsymbol{u}) \,|\, \boldsymbol{\psi}_i(\boldsymbol{x}_t, \boldsymbol{u}_t) \leq \boldsymbol{0}\}, \quad i \in \{1, 2, \dots, I\}. \end{aligned} \tag{1}$$

Here, $\boldsymbol{x}_t \in \mathbb{R}^n$ and $\boldsymbol{u}_t \in \mathbb{R}^m$ are the system state and input at time step $t = 0, 1, 2, \dots$. $i \in \{1, 2, \dots, I\}$ is the index of the system hybrid modes, and $\mathcal{P}_i \subset \mathbb{R}^n \times \mathbb{R}^m$ denotes the domain of the i-th mode, defined as a sublevel set of $\boldsymbol{\psi}_i(\boldsymbol{x}, \boldsymbol{u})$. $\boldsymbol{f}_i$ is the dynamics model (vector field) in the i-th mode.

A subset of hybrid systems in (1) corresponds to complementarity systems, which have been widely used to describe the multi-contact model of robot dynamics [14], [37], [38]:

$$\boldsymbol{M}(\boldsymbol{q}_t)(\boldsymbol{v}_{t+1} - \boldsymbol{v}_t) = \boldsymbol{C}(\boldsymbol{q}_t, \boldsymbol{v}_t) + \boldsymbol{B}\boldsymbol{u}_t + \sum_{i=1}^{I} \boldsymbol{J}_i(\boldsymbol{q}_t)^{\mathsf{T}} \Lambda_{i,t}, \tag{2}$$

with the i-th contact impulse $\Lambda_{i,t}$ satisfying the complementarity constraint

$$\boldsymbol{0} \leq \Lambda_{i,t} \perp \Phi_{i,t}(\boldsymbol{q}_t, \boldsymbol{v}_{t+1}, \Lambda_{i,t}) \geq \boldsymbol{0}, \quad i = 1, 2, \dots, I. \tag{3}$$

Here, $(\boldsymbol{q}_t, \boldsymbol{v}_t)$ is the generalized coordinate and velocity of a robot system. $\boldsymbol{u}_t$ is the actuation impulse with input projection matrix $\boldsymbol{B}$. $\boldsymbol{M}(\boldsymbol{q}_t)$ is the inertia matrix, and $\boldsymbol{C}(\boldsymbol{q}_t, \boldsymbol{v}_t)$ includes all non-contact impulses resulting from gravity and gyroscopic forces. $\Lambda_{i,t}$ is i-th contact impulse between the robot and objects/environments, and $\boldsymbol{J}_i(\boldsymbol{q}_t)$ is its Jacobian matrix. The complementarity constraint (3) means either the contact impulse $\Lambda_{i,t}$ or the value of its distance-related function, $\Phi_{i,t}(\boldsymbol{q}_t, \boldsymbol{v}_{t+1}, \Lambda_{i,t})$, is zero, but both cannot be negative, i.e., contact interaction between robot and object/environment cannot pull or penetrate into each other. Coulomb friction can be similarly described (e.g. [14]).

Define $\boldsymbol{x} := [\boldsymbol{q}, \boldsymbol{v}]^{\mathsf{T}}$, $\boldsymbol{\Lambda} := [\Lambda_1, \Lambda_1, ..., \Lambda_I]^{\mathsf{T}}$, and $\boldsymbol{\Phi} := [\Phi_1, \Phi_2, ..., \Phi_I]^{\mathsf{T}}$. One can abstractly write the multi-contact dynamics (2)-(3) into the general form below, denoted as $\boldsymbol{f}()$,

$$\boldsymbol{f}(): \qquad \begin{cases} \boldsymbol{F}(\boldsymbol{x}_{t+1}, \boldsymbol{x}_t, \boldsymbol{u}_t, \boldsymbol{\Lambda}_t) = \boldsymbol{0}, \\ \boldsymbol{0} \leq \boldsymbol{\Lambda}_t \perp \boldsymbol{\Phi}_t(\boldsymbol{x}_{t+1}, \boldsymbol{x}_t, \boldsymbol{u}_t, \boldsymbol{\Lambda}_t) \geq \boldsymbol{0}. \end{cases} \tag{4}$$

Connecting (4) to (1), here the active or inactive constraints in $\boldsymbol{\Phi} \geq 0$ determine the domain of hybrid modes, and $\boldsymbol{F}$ and $\boldsymbol{\Phi}$ jointly and implicitly determine the dynamics model of each hybrid mode. A significant amount of recent work focuses on identifying/learning the above complementary-based hybrid dynamics, such as [2]–[4], [8], [11].

The above complementary-based hybrid dynamics $\boldsymbol{f}()$ contains all potential contact modes in the aggregated contact impulse vector $\boldsymbol{\Lambda}$ and the corresponding distance vector function $\boldsymbol{\Phi}$. Thus, we call $\boldsymbol{f}()$ the *full-order hybrid dynamics*.

B. Full-Order Model Predictive Control

We consider the model predictive control (MPC) with the full-order hybrid dynamics $\boldsymbol{f}()$ in (4) for a given set of tasks:

$$\begin{aligned} \min_{\boldsymbol{u}_{0:T-1}} \quad & J_{\boldsymbol{\beta}} = \sum_{t=0}^{T-1} c_{\boldsymbol{\beta}}(\boldsymbol{x}_t, \boldsymbol{u}_t) + h_{\boldsymbol{\beta}}(\boldsymbol{x}_T) \\ \text{s.t.} \quad & \boldsymbol{x}_{t+1} = \boldsymbol{f}(\boldsymbol{x}_t, \boldsymbol{u}_t), \ \text{given} \ \boldsymbol{x}_0 \sim p_{\boldsymbol{\beta}}(\boldsymbol{x}_0), \end{aligned} \tag{5}$$

where T is the MPC horizon; $\boldsymbol{f}()$ is the hybrid dynamics model in (4); and $J_{\boldsymbol{\beta}}$ is a cost function for given tasks. Here, $\boldsymbol{\beta}$ is a general hyperparameter, indexing a set of robot tasks of interest, subject to a known task distribution $\boldsymbol{\beta} \sim p(\boldsymbol{\beta})$. For example, if the tasks of interest are that a robotic hand moving an object to different target poses, $\boldsymbol{\beta}$ can parameterize the set of target poses of the object, subject to a given distribution $p(\boldsymbol{\beta})$ reflecting the frequency of appearance of target pose β; if the tasks of interest contain different types of robot tasks, such as inserting a peg, turning a crank, etc., $\boldsymbol{\beta}$ can be the task index, and $p(\boldsymbol{\beta})$ could be a uniform distribution.

For notation simplicity, we write the system input and state trajectories compactly as $\mathbf{u} := \{\boldsymbol{u}_0, \boldsymbol{u}_1, \boldsymbol{u}_2, \dots, \boldsymbol{u}_{T-1}\}$ and $\mathbf{x} := \{\boldsymbol{x}_0, \boldsymbol{x}_1, \dots, \boldsymbol{x}_{T-1}, \boldsymbol{x}_T\}$, respectively. The system state trajectory given input trajectory $\mathbf{u}$ and initial $\boldsymbol{x}_0$ is written as

$$\mathbf{F}(\mathbf{u}, \boldsymbol{x}_0) := \Big\{ \mathbf{x} \,\Big|\, \boldsymbol{x}_{t+1} {=} \boldsymbol{f}(\boldsymbol{x}_t, \boldsymbol{u}_t), \ \text{given} \ \boldsymbol{x}_0 \ \text{and} \ \mathbf{u} \Big\}. \tag{6}$$

The solution to (5) then can be compactly written as

$$\begin{aligned} \boldsymbol{f}\text{-MPC}: \quad \mathbf{u}^{\boldsymbol{f}}(\boldsymbol{x}_0, \boldsymbol{\beta}) := \arg\min_{\mathbf{u}} J_{\boldsymbol{\beta}}\big(\mathbf{u}, \mathbf{F}(\mathbf{u}, \boldsymbol{x}_0)\big), \\ \boldsymbol{x}_0 \sim p_{\boldsymbol{\beta}}(\boldsymbol{x}_0), \ \boldsymbol{\beta} \sim p(\boldsymbol{\beta}). \end{aligned} \tag{7}$$

The full-order dynamics MPC in (7) is applied to the multi-contact robot system $\boldsymbol{f}()$ in a closed-loop (receding) fashion. Specifically, at rollout time step $k = 1, 2, 3, ...$, $\boldsymbol{x}_0$ in (7) is set to the robot's actual state: $\boldsymbol{x}_0 = \boldsymbol{x}_k^{\boldsymbol{f}}$. After solving $\mathbf{u}^{\boldsymbol{f}}(\boldsymbol{x}_k^{\boldsymbol{f}}, \boldsymbol{\beta}) = \{\boldsymbol{u}_0^{\boldsymbol{f}}, \dots, \boldsymbol{u}_{T-1}^{\boldsymbol{f}}\}$ from (7), only the first input $\boldsymbol{u}_0^{\boldsymbol{f}}$ is applied to the robot for execution and drive the robot to the next state: $\boldsymbol{x}_{k+1}^{\boldsymbol{f}}$ via $\boldsymbol{x}_{k+1}^{\boldsymbol{f}} = \boldsymbol{f}(\boldsymbol{x}_k^{\boldsymbol{f}}, \boldsymbol{u}_0^{\boldsymbol{f}})$. Then, this process repeats at the robot new state $\boldsymbol{x}^{\boldsymbol{f}}_{t+1}$. The above MPC leads to a closed-loop control policy: mapping from robot's current state $\boldsymbol{x}^{\boldsymbol{f}}_{k}$ to its control input $\boldsymbol{u}^{\boldsymbol{f}}_{0}$.

As indicated by the full-order dynamics $\boldsymbol{f}()$ in (4), solving the MPC in (7) requires reasoning over the sequence of contact impulses $\{\boldsymbol{\Lambda}_0, \boldsymbol{\Lambda}_1, \dots, \boldsymbol{\Lambda}_{T-1}\}$ in addition to $\mathbf{u}$ and $\mathbf{x}$. Its combinatoric complexity is 2^{TI} ($I = \dim \boldsymbol{\Lambda}$). Despite recent progress, particularly on modestly sized problems [6], [8], [11], [19], a large number of potential contact interactions (e.g., large I) will make solving (7) intractable.

In this paper, we hypothesize that identifying and utilizing a full-order dynamics $\boldsymbol{f}()$ for MPC in (7) is almost certainly unnecessary, because far fewer modes are actually necessary to accomplish many tasks. Thus, we will find a reduced-order hybrid model proxy to replace $\boldsymbol{f}()$ in (7), to enable real-time control and sufficiently achieve high task performance.

C. Linear Complementarity Systems

To find a reduced-order hybrid representation for the full-order hybrid dynamics $\boldsymbol{f}()$ in (4), we consider piecewise affine (PWA) models. This is motivated by the fact that many physics simulation engines [13]–[16] locally use linear complementarity models (a compact formation of continuous PWA [17]) to handle physical contacts at each simulation step. A PWA model can sufficiently describe multi-modality but is tractable enough for planning and control tasks due to their simple (affine) structures. As in our previous work [4], we compactly represent PWA models as a linear complementarity system (LCS), defined as $\boldsymbol{g}()$,

$$\boldsymbol{g}(): \quad \begin{aligned} &\boldsymbol{x}_{t+1} = A\boldsymbol{x}_t + B\boldsymbol{u}_t + C\boldsymbol{\lambda}_t + \boldsymbol{d} \\ &\mathbf{0} \leq \boldsymbol{\lambda}_t \perp D\boldsymbol{x}_t + E\boldsymbol{u}_t + F\boldsymbol{\lambda}_t + \boldsymbol{c} \geq \mathbf{0}. \end{aligned} \tag{8}$$

Here, the first line of (8) is the affine dynamics and the second line is the complementarity equation. $(A, B, C, \boldsymbol{d}, D, E, F, \boldsymbol{c})$ are system matrix parameters with compatible dimensions. $\boldsymbol{\lambda}_t \in \mathbb{R}^r$ is the complementarity variable and solved from the complementarity equation given $(\boldsymbol{x}_t, \boldsymbol{u}_t)$. Depending on the active or inactive inequalities in $D\boldsymbol{x}_t + E\boldsymbol{u}_t + F\boldsymbol{\lambda}_t + \boldsymbol{c} \geq \mathbf{0}$ (corresponding to different partitions of the state-input space), $\boldsymbol{\lambda}_t \in \mathbb{R}^r$ is a piecewise function of $(\boldsymbol{x}_t, \boldsymbol{u}_t)$. By composing with affine dynamics, $\boldsymbol{x}_{t+1}$ is eventually a piecewise function of $(\boldsymbol{x}_t, \boldsymbol{u}_t)$, and each linear piece is a hybrid mode. Thus, the maximum number of the hybrid modes the LCS in (8) can represent is $2^{\dim \boldsymbol{\lambda}}$. For any given $(\boldsymbol{x}_t, \boldsymbol{u}_t)$, to guarantee the existence and uniqueness of $\boldsymbol{\lambda}_t$ solved from the complementary equation, we impose the restriction that the symmetric part of F be positive definite, $F^{\mathsf{T}} + F \succ 0$ [39], [40]. This property can be accomplished by parameterizing F as

$$F := GG^{\mathsf{T}} + H - H^{\mathsf{T}}, \tag{9}$$

with G and H matrices with the same dimension as F.

As we will seek to learn a reduced-order LCS $\boldsymbol{g}()$, we can explicitly restrict the number of potential modes in $\boldsymbol{g}()$ by setting the dimension of the complementary variable, $\dim \boldsymbol{\lambda}$. Compared to the full-order dynamics $\boldsymbol{f}()$ in (4), $\boldsymbol{g}()$ has

$$\dim \boldsymbol{\lambda} < \dim \boldsymbol{\Lambda}. \tag{10}$$

Note that, we do not expect a tight connection between $\boldsymbol{\lambda}$ in $\boldsymbol{g}()$ and the physical contact impulse vector $\boldsymbol{\Lambda}$ in $\boldsymbol{f}()$. Instead, $\boldsymbol{\lambda}$ in $\boldsymbol{g}()$ here will represent general multi-modality, and while we will later observe that $\boldsymbol{\lambda}$ is empirically related to the contact forces, it is not exactly the same.

D. Problem Formulation

We aim to find a reduced-order LCS model $\boldsymbol{g}()$ in (8) for the given set of tasks $J_{\boldsymbol{\beta}}$ in (5), and establish the following reduced-order $\boldsymbol{g}$-MPC (using the notation convention in (7)):

$$\begin{aligned} \boldsymbol{g}\text{-MPC}: \quad \mathbf{u}^{\boldsymbol{g}}(\boldsymbol{x}_0, \boldsymbol{\beta}) := \arg\min_{\mathbf{u}} J_{\boldsymbol{\beta}}\big(\mathbf{u}, \mathbf{G}(\mathbf{u}, \boldsymbol{x}_0)\big), \\ \boldsymbol{x}_0 \sim p_{\boldsymbol{\beta}}(\boldsymbol{x}_0), \ \boldsymbol{\beta} \sim p(\boldsymbol{\beta}), \end{aligned} \tag{11}$$

such that when running the reduced-order $\boldsymbol{g}$-MPC on the full-order robot dynamics $\boldsymbol{f}()$, one can achieve a task performance as similar to the task performance of running $\boldsymbol{f}$-MPC on $\boldsymbol{f}()$ as possible. Here, we also compactly write the state trajectory of $\boldsymbol{g}()$ given $\mathbf{u}$ and $\boldsymbol{x}_0$ as

$$\mathbf{G}(\mathbf{u}, \boldsymbol{x}_0) := \Big\{\mathbf{x} \,\Big|\, \boldsymbol{x}_{t+1} {=} \boldsymbol{g}(\boldsymbol{x}_t, \boldsymbol{u}_t), \text{ given } \boldsymbol{x}_0 \text{ and } \mathbf{u}\Big\}. \tag{12}$$

Therefore, the goal of task-driven reduced-order model learning is to find the reduced-order LCS $\boldsymbol{g}()$ which minimizes the following *task performance gap*:

$$\begin{aligned} \mathcal{L}(\boldsymbol{g}) := \mathbb{E}_{\boldsymbol{\beta} \sim p(\boldsymbol{\beta})} \, \mathbb{E}_{\boldsymbol{x} \sim p_{\boldsymbol{\beta}}(\boldsymbol{x}_0)} \Big[& J_{\boldsymbol{\beta}}\big(\mathbf{u}^{\boldsymbol{g}}, \mathbf{F}(\mathbf{u}^{\boldsymbol{g}}, \boldsymbol{x}_0)\big) \\ & - J_{\boldsymbol{\beta}}\big(\mathbf{u}^{\boldsymbol{f}}, \mathbf{F}(\mathbf{u}^{\boldsymbol{f}}, \boldsymbol{x}_0)\big) \Big], \end{aligned} \tag{13}$$

where the first cost $J_{\boldsymbol{\beta}}\big(\mathbf{u}^{\boldsymbol{g}}, \mathbf{F}(\mathbf{u}^{\boldsymbol{g}}, \boldsymbol{x}_0)\big)$ is the task performance of running reduced-order $\boldsymbol{g}$-MPC on the robot system $\boldsymbol{f}()$, and the second cost $J_{\boldsymbol{\beta}}\big(\mathbf{u}^{\boldsymbol{g}}, \mathbf{F}(\mathbf{u}^{\boldsymbol{g}}, \boldsymbol{x}_0)\big)$ is the task performance of running full-order $\boldsymbol{f}$-MPC on the robot system $\boldsymbol{f}()$. Here, $\mathbf{u}^{\boldsymbol{g}}$ is the solution to the reduced-order $\boldsymbol{g}$-MPC in (11) and $\mathbf{u}^{\boldsymbol{f}}$ is the solution to the full-order $\boldsymbol{f}$-MPC in (7).

We make the following remarks on the above problem statement. First, the reduced-order LCS $\boldsymbol{g}()$ shares the same dimensions of states and inputs as the full-order dynamics $\boldsymbol{f}()$, but has far fewer hybrid modes by setting $\dim \boldsymbol{\lambda} \leq \dim \boldsymbol{\Lambda}$. Compared to $\boldsymbol{f}$-MPC, the reduced-order $\boldsymbol{g}$-MPC is more computationally tractable for real-time implementation. Second, the learning criterion (13) is to minimize the performance gap between the reduced-order $\boldsymbol{g}$-MPC and full-order $\boldsymbol{f}$-MPC, both MPC controllers running on the full-order dynamics $\boldsymbol{f}()$, which is the original hybrid system. Thus, a minimal performance gap means that one can confidently use the reduced-order LCS $\boldsymbol{g}()$ to achieve the given tasks $J_{\boldsymbol{\beta}}$, $\boldsymbol{\beta} \sim p(\boldsymbol{\beta})$.

Directly minimizing the task performance gap $\mathcal{L}(\boldsymbol{g})$ requires access to and optimization with the full-order dynamics model $\boldsymbol{f}()$, because of the coupling between $J_{\boldsymbol{\beta}}()$ and $\boldsymbol{f}()$ in (13). However, this is unlikely to be tractable as the full-order model is both unknown and too complex to optimize with. In the following section, we develop a method to approximately solve (13) without requiring knowledge of the model $\boldsymbol{f}()$.

IV. Theoretical Results

In this section, we will show that instead of directly solving (13), one can minimize its upper bound. This will lead to developing a method that is much easier to implement and only requires samples (zero-order information) of $\boldsymbol{f}()$. To start, we pose a mild assumption about the Lipschitz continuity of task cost function $J_{\boldsymbol{\beta}}(\mathbf{u}, \mathbf{x})$ for any $\boldsymbol{\beta} \sim p(\boldsymbol{\beta})$.

Assumption 1. *For any task sample $\boldsymbol{\beta} \sim p(\boldsymbol{\beta})$, the task cost function $J_{\boldsymbol{\beta}}(\mathbf{u}, \mathbf{x})$ is M-Lipschitz continuous, i.e., for any $\mathbf{z}_1 := (\mathbf{u}_1, \mathbf{x}_1)$, $\mathbf{z}_2 := (\mathbf{u}_2, \mathbf{x}_2)$,*

$$|J_r(\mathbf{z}_1) - J_r(\mathbf{z}_2)| \leq M\|\mathbf{z}_1 - \mathbf{z}_2\| \tag{14}$$

with $\|\|$ denoting the l_2 norm.

The above assumption is mild, as the cost function is usually defined manually and can easily satisfy this condition. With Assumption 1, we have the following lemma stating the upper bound of the task performance gap $\mathcal{L}(\boldsymbol{g})$ in (13):

Lemma 1. *Suppose Assumption 1 holds. For any reduced-order model $\boldsymbol{g}()$, the following inequality holds:*

$$\mathcal{L}(\boldsymbol{g}) \leq M\, \mathbb{E}_{\boldsymbol{\beta}\sim p(\boldsymbol{\beta})}\, \mathbb{E}_{\boldsymbol{x}\sim p_{\boldsymbol{\beta}}(\boldsymbol{x}_0)} \Big(\big\|\mathbf{G}(\mathbf{u}^{\boldsymbol{g}}, \boldsymbol{x}_0) - \mathbf{F}(\mathbf{u}^{\boldsymbol{g}}, \boldsymbol{x}_0)\big\| + \big\|\mathbf{G}(\mathbf{u}^{\boldsymbol{f}}, \boldsymbol{x}_0) - \mathbf{F}(\mathbf{u}^{\boldsymbol{f}}, \boldsymbol{x}_0)\big\| \Big) \tag{15}$$

Proof. See Appendix. A. □

Lemma 1 gives an upper bound for the task performance gap $\mathcal{L}(\boldsymbol{g})$. Notably, this upper bound is the *prediction error* between the reduced-order model $\boldsymbol{g}()$ and full-order dynamics $\boldsymbol{f}()$ at their MPC solutions. Specifically, the first term on the right side of (15) is the model prediction error on the dataset

$$\mathcal{D}^{\boldsymbol{g}} = \Big\{ \mathbf{u}^{\boldsymbol{g}}(\boldsymbol{x}_0, \boldsymbol{\beta}) \,|\, \boldsymbol{x}_0 \sim p(\boldsymbol{x}_0),\ \boldsymbol{\beta} \sim p(\boldsymbol{\beta}) \Big\} \tag{16}$$

generated by the reduced-order $\boldsymbol{g}$-MPC. The second term on the right side of (15) is the model prediction error on

$$\mathcal{D}^{\boldsymbol{f}} = \Big\{ \mathbf{u}^{\boldsymbol{f}}(\boldsymbol{x}_0, \boldsymbol{\beta}) \,|\, \boldsymbol{x}_0 \sim p(\boldsymbol{x}_0),\ \boldsymbol{\beta} \sim p(\boldsymbol{\beta}) \Big\} \tag{17}$$

generated by the full model $\boldsymbol{f}$-MPC. (15) says that as long as the reduced-order model $\boldsymbol{g}()$ captures the full-order dynamics $\boldsymbol{f}()$ **at the MPC data** $\mathcal{D}^{\boldsymbol{g}}$ and $\mathcal{D}^{\boldsymbol{f}}$, not necessarily at other parts of data regime (task-irrelevant data), $\boldsymbol{g}$-MPC can replace $\boldsymbol{f}$-MPC for the same task performance. Thus, Lemma 1 justifies the learning of a task-driven reduced-order model.

Although the $\boldsymbol{f}$-MPC policy data $\mathcal{D}^{\boldsymbol{f}}$ in (17) is not directly verifiable when $\boldsymbol{f}()$ is unknown, the next lemma will show the $\boldsymbol{g}$-MPC data $\mathcal{D}^{\boldsymbol{g}}$ is related to $\mathcal{D}^{\boldsymbol{f}}$, which thus can be verified indirectly.

Lemma 2. *Suppose $\nabla_{\mathbf{x}} J_{\boldsymbol{\beta}}(\mathbf{u}, \mathbf{x})$ is L_1-Lipschitz continuous, $\nabla_{\mathbf{u}} J_{\boldsymbol{\beta}}(\mathbf{u}, \mathbf{x})$ is L_2-Lipschitz continuous, and $\|\nabla_{\mathbf{x}} J_{\boldsymbol{\beta}}(\mathbf{u}, \mathbf{x})\| \leq M_1$ for any $(\mathbf{u}, \mathbf{x}, \boldsymbol{\beta})$; $\|\nabla_{\mathbf{u}} \mathbf{G}(\mathbf{u}, \boldsymbol{x}_0)\| \leq M_g$ for any $(\mathbf{u}, \boldsymbol{x}_0)$. Then, the solutions $\mathcal{D}^{\boldsymbol{g}}$ in (16) generated by $\boldsymbol{g}$-MPC is also an ϵ-accuracy stationary solution for $\boldsymbol{f}$-MPC, i.e.,*

$$\|\nabla_{\mathbf{u}} J_{\boldsymbol{\beta}}\big(\mathbf{u}^{\boldsymbol{g}}, \mathbf{F}(\mathbf{u}^{\boldsymbol{g}}, \boldsymbol{x}_0)\big)\| \leq \epsilon \tag{18}$$

for any $\boldsymbol{x}_0 \sim p(\boldsymbol{x}_0)$, $\boldsymbol{\beta} \sim p(\boldsymbol{\beta})$ with

$$\epsilon = M_1 \|\nabla_{\mathbf{u}} \mathbf{F}(\mathbf{u}^{\boldsymbol{g}}, \boldsymbol{x}_0) - \nabla_{\mathbf{u}} \mathbf{G}(\mathbf{u}^{\boldsymbol{g}}, \boldsymbol{x}_0)\| + (L_2 + M_g L_1)\|\mathbf{F}(\mathbf{u}^{\boldsymbol{g}}, \boldsymbol{x}_0) - \mathbf{G}(\mathbf{u}^{\boldsymbol{g}}, \boldsymbol{x}_0)\| \tag{19}$$

Proof. See Appendix B. □

Lemma 2 suggests the $\boldsymbol{g}$-MPC data $\mathcal{D}^{\boldsymbol{g}}$ in (16) can become $\boldsymbol{f}$-MPC data $\mathcal{D}^{\boldsymbol{f}}$ in (17), if the reduced-order model $\boldsymbol{g}()$ fits well to the true $\boldsymbol{f}()$ in both zeroth and first orders, i.e.,

$$\|\mathbf{F}(\mathbf{u}^{\boldsymbol{g}}, \boldsymbol{x}_0) - \mathbf{G}(\mathbf{u}^{\boldsymbol{g}}, \boldsymbol{x}_0)\| \rightarrow 0 \tag{20}$$

$$\|\nabla_{\mathbf{u}} \mathbf{F}(\mathbf{u}^{\boldsymbol{g}}, \boldsymbol{x}_0) - \nabla_{\mathbf{u}} \mathbf{G}(\mathbf{u}^{\boldsymbol{g}}, \boldsymbol{x}_0)\| \rightarrow 0 \tag{21}$$

Thus, one can indirectly verify (17) by additionally looking at the first-order model precision error $\|\nabla_{\mathbf{u}} \mathbf{F}(\mathbf{u}^{\boldsymbol{g}}, \boldsymbol{x}_0) - \nabla_{\mathbf{u}} \mathbf{G}(\mathbf{u}^{\boldsymbol{g}}, \boldsymbol{x}_0)\|$, where $\nabla_{\mathbf{u}} \mathbf{F}(\mathbf{u}^{\boldsymbol{g}}, \boldsymbol{x}_0)$ can be estimated numerically via mesh grid of $\mathcal{D}^{\boldsymbol{g}}$.

Jointly looking at Lemma 1 and Lemma 2, one can conclude that if we can find a reduced-order LCS $\boldsymbol{g}()$ such that it fits $\boldsymbol{f}()$ well in both zeroth and first-order prediction on the $\boldsymbol{g}$-MPC data $\mathcal{D}^{\boldsymbol{g}}$, as in (20) and (21), respectively, such $\boldsymbol{g}()$ is minimizing the upper bound (15), thus eventually minimizing the task performance gap $\mathcal{L}(\boldsymbol{g})$ itself. Those theoretical insights will guide us to develop algorithms in the next section.

V. Practical Algorithm

The technical analysis in the previous section says that to minimize the upper bound (15) of the task performance gap, one might fit a reduced-order model $\boldsymbol{g}()$ to full-order dynamics $\boldsymbol{f}()$ well in both zeroth and first-order prediction using the $\boldsymbol{g}$-MPC policy data $\mathcal{D}^{\boldsymbol{g}}$. We take this as inspiration, though we note that, for efficiency, we will minimize zeroth-order error and will not check first-order criteria. Now, we develop the task-driven hybrid-model reduction algorithm. Throughout the following paper, $\boldsymbol{g}$-MPC will be implemented in a closed-loop (receding) fashion, i.e., the only first action of the MPC solution is applied to the robot system $\boldsymbol{f}()$.

The building blocks of the task-driven hybrid model reduction algorithm are in Fig. 1. The learning process is iterative, and each iteration includes the following three components.

- **Trust-region LCS model predictive controller:** The latest LCS $\boldsymbol{g}()$ is used in the MPC controller. Compared to (11), we additionally introduce a trust region on the control inputs in the reduced-order LCS MPC. This trust region may also be adapted according to the latest Rollout Buffer with details given in Section V-B.
- **Rollout Buffer:** denoted as $\boldsymbol{\mathcal{D}}_{\text{buffer}}=\big\{(\boldsymbol{x}^{\boldsymbol{f}}_k, \boldsymbol{u}^{\boldsymbol{g}\text{-MPC}}_k, \boldsymbol{x}^{\boldsymbol{f}}_{k+1})\big\}$, stores the current and history rollout data from running the trust-region $\boldsymbol{g}$-MPC controller on the robot (full-order dynamics). The buffer can permit a maximum buffer size.
- **Learning reduce-order LCS:** This is to train reduced-order LCS $\boldsymbol{g}()$ using the data of the latest Rollout Buffer $\boldsymbol{\mathcal{D}}_{\text{buffer}}$. Details of the training process is given in Section V-A.

A. Learning Reduced-Order LCS

We use the method of our recent work [41] to learn the LCS $\boldsymbol{g}()$ from Rollout Buffer data $\boldsymbol{\mathcal{D}}_{\text{buffer}} = \big\{(\boldsymbol{x}^{\boldsymbol{f}}_k, \boldsymbol{u}^{\boldsymbol{g}\text{-MPC}}_k, \boldsymbol{x}^{\boldsymbol{f}}_{k+1})\big\}$. This method enables efficient learning of a PWA model with

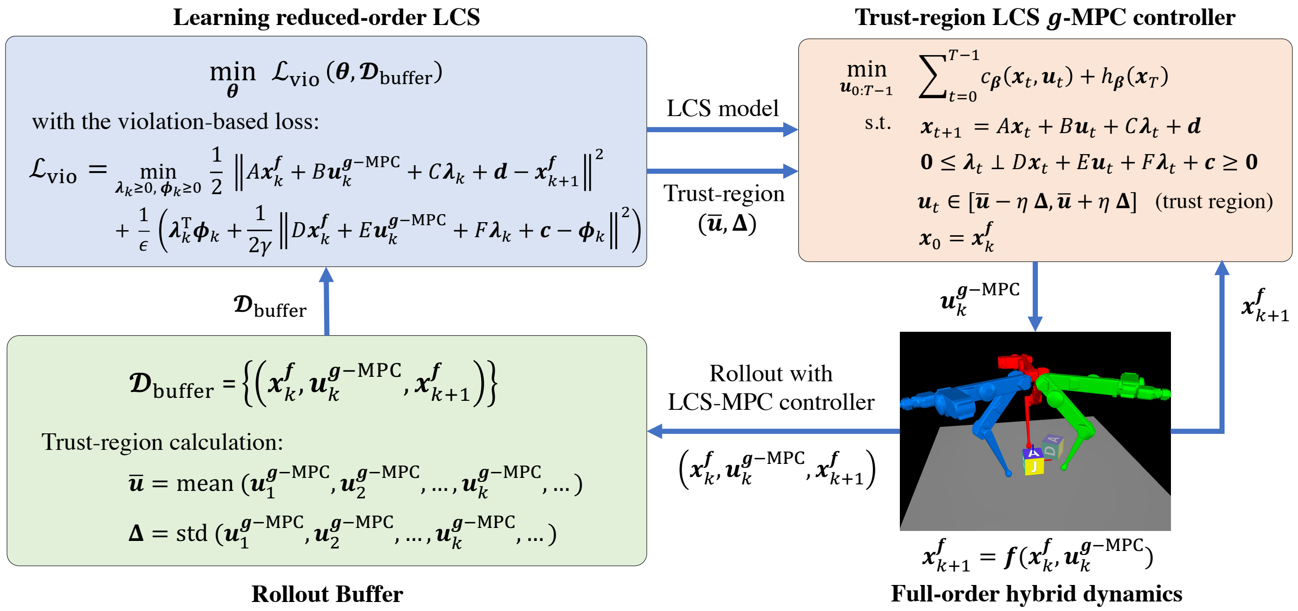

Learning reduced-order LCS
$\min_{\boldsymbol{\theta}} \ \mathcal{L}_{\text{vio}}(\boldsymbol{\theta}, \boldsymbol{\mathcal{D}}_{\text{buffer}})$
with the violation-based loss:
$\mathcal{L}_{\text{vio}} = \min_{\boldsymbol{\lambda}_k \geq 0, \boldsymbol{\phi}_k \geq 0} \frac{1}{2}\left\| A\boldsymbol{x}^{\boldsymbol{f}}_k + B\boldsymbol{u}^{\boldsymbol{g}-\text{MPC}}_k + C\boldsymbol{\lambda}_k + \boldsymbol{d} - \boldsymbol{x}^{\boldsymbol{f}}_{k+1} \right\|^2 + \frac{1}{\epsilon}\left(\boldsymbol{\lambda}_k^{\mathsf{T}}\boldsymbol{\phi}_k + \frac{1}{2\gamma}\left\| D\boldsymbol{x}^{\boldsymbol{f}}_k + E\boldsymbol{u}^{\boldsymbol{g}-\text{MPC}}_k + F\boldsymbol{\lambda}_k + \boldsymbol{c} - \boldsymbol{\phi}_k \right\|^2\right)$
LCS model
Trust-region
$(\bar{\boldsymbol{u}}, \boldsymbol{\Delta})$
Trust-region LCS $\boldsymbol{g}$-MPC controller
$\min_{\boldsymbol{u}_{0:T-1}} \sum_{t=0}^{T-1} c_{\boldsymbol{\beta}}(\boldsymbol{x}_t, \boldsymbol{u}_t) + h_{\boldsymbol{\beta}}(\boldsymbol{x}_T)$
s.t. $\boldsymbol{x}_{t+1} = A\boldsymbol{x}_t + B\boldsymbol{u}_t + C\boldsymbol{\lambda}_t + \boldsymbol{d}$
$\boldsymbol{0} \leq \boldsymbol{\lambda}_t \perp D\boldsymbol{x}_t + E\boldsymbol{u}_t + F\boldsymbol{\lambda}_t + \boldsymbol{c} \geq \boldsymbol{0}$
$\boldsymbol{u}_t \in [\bar{\boldsymbol{u}} - \eta\,\boldsymbol{\Delta}, \bar{\boldsymbol{u}} + \eta\,\boldsymbol{\Delta}]$ (trust region)
$\boldsymbol{x}_0 = \boldsymbol{x}^{\boldsymbol{f}}_k$
$\boldsymbol{\mathcal{D}}_{\text{buffer}}$
$\boldsymbol{u}^{\boldsymbol{g}-\text{MPC}}_k$
$\boldsymbol{x}^{\boldsymbol{f}}_{k+1}$
$\boldsymbol{\mathcal{D}}_{\text{buffer}} = \left\{\left(\boldsymbol{x}^{\boldsymbol{f}}_k, \boldsymbol{u}^{\boldsymbol{g}-\text{MPC}}_k, \boldsymbol{x}^{\boldsymbol{f}}_{k+1}\right)\right\}$
Trust-region calculation:
$\bar{\boldsymbol{u}} = \text{mean}\,(\boldsymbol{u}^{\boldsymbol{g}-\text{MPC}}_1, \boldsymbol{u}^{\boldsymbol{g}-\text{MPC}}_2, \ldots, \boldsymbol{u}^{\boldsymbol{g}-\text{MPC}}_k, \ldots)$
$\boldsymbol{\Delta} = \text{std}\,(\boldsymbol{u}^{\boldsymbol{g}-\text{MPC}}_1, \boldsymbol{u}^{\boldsymbol{g}-\text{MPC}}_2, \ldots, \boldsymbol{u}^{\boldsymbol{g}-\text{MPC}}_k, \ldots)$
Rollout with
LCS-MPC controller
$\left(\boldsymbol{x}^{\boldsymbol{f}}_k, \boldsymbol{u}^{\boldsymbol{g}-\text{MPC}}_k, \boldsymbol{x}^{\boldsymbol{f}}_{k+1}\right)$
$\boldsymbol{x}^{\boldsymbol{f}}_{k+1} = \boldsymbol{f}(\boldsymbol{x}^{\boldsymbol{f}}_k, \boldsymbol{u}^{\boldsymbol{g}-\text{MPC}}_k)$
Rollout Buffer
Full-order hybrid dynamics

Fig. 1: Components of task-driven hybrid model reduction algorithm. There are three main components: learning reduced-order LCS, Trust-region LCS model predictive controller, and Rollout Buffer, each of which is detailed in the text.

up to thousands of hybrid modes and effectively handles the stiff dynamics that arises from contact. For self-containment, the method is described below.

By learning a LCS in (8), we mean to learn all its matrix parameters, denoted as

$$\boldsymbol{\theta} := \{A, B, C, \boldsymbol{d}, D, E, F, \boldsymbol{c}\}. \tag{22}$$

In [41], we presented a new learning method, which learns $\boldsymbol{\theta}$ by minimizing the following violation-based loss

$$\mathcal{L}_{\text{vio}}(\boldsymbol{\theta}, \boldsymbol{\mathcal{D}}_{\text{buffer}}) = \sum_k \mathcal{L}_{\text{vio}}\left(\boldsymbol{\theta}, (\boldsymbol{x}^{\boldsymbol{f}}_k, \boldsymbol{u}^{\boldsymbol{g}\text{-MPC}}_k, \boldsymbol{x}^{\boldsymbol{f}}_{k+1})\right) \tag{23}$$

with

$$\begin{aligned}
&\mathcal{L}_{\text{vio}}\left(\boldsymbol{\theta}, (\boldsymbol{x}^{\boldsymbol{f}}_k, \boldsymbol{u}^{\boldsymbol{g}\text{-MPC}}_k, \boldsymbol{x}^{\boldsymbol{f}}_{k+1})\right) := \\
&\min_{\boldsymbol{\lambda}_k \geq \boldsymbol{0},\ \boldsymbol{\phi}_k \geq \boldsymbol{0}} \ \frac{1}{2}\| A\boldsymbol{x}^{\boldsymbol{f}}_k + B\boldsymbol{u}^{\boldsymbol{g}\text{-MPC}}_k + C\boldsymbol{\lambda}_k + \boldsymbol{d} - \boldsymbol{x}^{\boldsymbol{f}}_{k+1}\|^2 + \\
&\frac{1}{\epsilon}\left(\boldsymbol{\lambda}_k^{\mathsf{T}}\phi_k + \frac{1}{2\gamma}\| D\boldsymbol{x}^{\boldsymbol{f}}_k + E\boldsymbol{u}^{\boldsymbol{g}\text{-MPC}}_k + F\boldsymbol{\lambda}_k + \boldsymbol{c} - \phi_k\|^2\right).
\end{aligned}$$

In the above loss $\mathcal{L}_{\text{vio}}\left(\boldsymbol{\theta}, (\boldsymbol{x}^{\boldsymbol{f}}_k, \boldsymbol{u}^{\boldsymbol{g}\text{-MPC}}_k, \boldsymbol{x}^{\boldsymbol{f}}_{k+1})\right)$, the first and second terms are the violation of the affine dynamics and complementarity equations by a buffer data point $(\boldsymbol{x}^{\boldsymbol{f}}_k, \boldsymbol{u}^{\boldsymbol{g}\text{-MPC}}_k, \boldsymbol{x}^{\boldsymbol{f}}_{k+1})$, respectively. $\epsilon > 0$, which empirically takes its value from the range $(10^{-3}, 1)$, is a hyperparameter that balances the violation of these two terms. Here, $\boldsymbol{\phi} \in \mathbb{R}^r$ is an introduced slack variable for the complementarity equation, and $\gamma > 0$ can be any value as long as satisfying $\gamma \leq \sigma_{\min}(F^{\mathsf{T}} + F)$ (i.e., the smallest singular value of the matrix $(F^{\mathsf{T}} + F)$). Such a choice of γ ensures the strong convexity of the quadratic objective $\mathcal{L}_{\text{vio}}\left(\boldsymbol{\theta}, (\boldsymbol{x}^{\boldsymbol{f}}_k, \boldsymbol{u}^{\boldsymbol{g}\text{-MPC}}_k, \boldsymbol{x}^{\boldsymbol{f}}_{k+1})\right)$ in the variable $(\boldsymbol{\lambda}_k, \boldsymbol{\phi}_k)$.

As theoretically shown in [41], the above LCS learning loss $\mathcal{L}_{\text{vio}}(\boldsymbol{\theta}, \boldsymbol{\mathcal{D}}_{\text{buffer}})$ has the following properties. First, it can be proved that the inner optimization over $(\boldsymbol{\lambda}_k, \boldsymbol{\phi}_k)$ is a convex quadratic program, thus can be efficiently solved in batch using state-of-the-art solvers, e.g., OSQP [42]. Second, the gradient of the violation-based loss $\mathcal{L}_{\text{vio}}(\boldsymbol{\theta}, \boldsymbol{\mathcal{D}}_{\text{buffer}})$ with respect to all matrices in $\boldsymbol{\theta}$ can be analytically obtained using the Envelope Theorem [43] (without differentiating through the solution to the inner optimization). Third, by adding both the affine dynamics violation and complementarity violation with a balance weight ϵ, $\mathcal{L}_{\text{vio}}(\boldsymbol{\theta}, \boldsymbol{\mathcal{D}}_{\text{buffer}})$ attains a better conditioned loss landscape, enabling simultaneous identification of stiff and multi-modal dynamics.

B. *Trust-Region LCS Model Predictive Controller*

With the reduced-order LCS $\boldsymbol{g}()$, one can establish the following trust-region reduced-order LCS-based MPC:

$$\begin{aligned}
\min_{\boldsymbol{u}_{0:T-1}} \quad & \sum_{t=0}^{T-1} c_{\boldsymbol{\beta}}(\boldsymbol{x}_t, \boldsymbol{u}_t) + h_{\boldsymbol{\beta}}(\boldsymbol{x}_T) \qquad \boldsymbol{\beta} \sim p(\boldsymbol{\beta}) \\
\text{subject to} \quad & \boldsymbol{u}_t \in [\bar{\boldsymbol{u}} - \boldsymbol{\Delta}, \bar{\boldsymbol{u}} + \boldsymbol{\Delta}], \\
& \boldsymbol{x}_{t+1} = A\boldsymbol{x}_t + B\boldsymbol{u}_t + C\boldsymbol{\lambda}_t + \boldsymbol{d}, \\
& \boldsymbol{0} \leq \boldsymbol{\lambda}_t \perp D\boldsymbol{x}_t + E\boldsymbol{u}_t + F\boldsymbol{\lambda}_t + \boldsymbol{c} \geq \boldsymbol{0}, \\
& \boldsymbol{x}_0 = \boldsymbol{x}^{\boldsymbol{f}}_k.
\end{aligned} \tag{24}$$

Compared to the early $\boldsymbol{g}$-MPC in (11), the difference here is that we have enforced a trust region constraint $\bar{\boldsymbol{u}} - \boldsymbol{\Delta} \leq \boldsymbol{u}_t \leq \bar{\boldsymbol{u}} + \boldsymbol{\Delta}$ on the control input $\boldsymbol{u}_t$, $t = 0, 1, \ldots, T-1$. This is due to the following reasons. As shown in Fig. 1, since $\boldsymbol{g}()$ is trained on the current buffer data $\boldsymbol{\mathcal{D}}_{\text{buffer}}$, we expect $\boldsymbol{g}()$ is likely valid only on the region covered by $\boldsymbol{\mathcal{D}}_{\text{buffer}}$, which we refer to as the *trust region*. Thus, we constrain $\boldsymbol{g}$-MPC in (24) to this trust region, prohibiting the controller from attempting to exploit model error and generating undesired controls.

The center $\bar{\boldsymbol{u}}$ and size $\boldsymbol{\Delta}$ of the trust region may be updated along with the rollout buffer $\boldsymbol{\mathcal{D}}_{\text{buffer}}$ during each iteration. In our algorithm, at i-th iteration, we simply set the trust region center $\bar{\boldsymbol{u}}_i$ as the mean of all control input data in the current Rollout Buffer $\boldsymbol{\mathcal{D}}_{\text{buffer},i}$, i.e.,

$$\bar{\boldsymbol{u}}_i = \text{mean}\big(\{\boldsymbol{u}_1^{\boldsymbol{g}\text{-MPC}}, ..., \boldsymbol{u}_k^{\boldsymbol{g}\text{-MPC}}, ...\}\big), \qquad \boldsymbol{u}_k^{\boldsymbol{g}\text{-MPC}} \in \boldsymbol{\mathcal{D}}_{\text{buffer},i} \tag{25}$$

and the trust region size $\boldsymbol{\Delta}_i$ is set according to the standard deviation of all input data in $\boldsymbol{\mathcal{D}}_{\text{buffer},i}$:

$$\boldsymbol{\Delta}_i = \eta_i \ \text{std}\big(\{\boldsymbol{u}_1^{\boldsymbol{g}\text{-MPC}}, ..., \boldsymbol{u}_k^{\boldsymbol{g}\text{-MPC}}, ...\}\big), \qquad \boldsymbol{u}_k^{\boldsymbol{g}\text{-MPC}} \in \boldsymbol{\mathcal{D}}_{\text{buffer},i} \tag{26}$$

Here, $\eta_i > 0$ a hyperparameter of the trust region at the i-th iteration, and mean() and std() are applied dimension-wise. It is also possible that $\bar{\boldsymbol{u}}_i$ and $\boldsymbol{\Delta}_i$ are set using other rules, e.g., following the classic trust-region optimization [44].

To solve the LCS MPC in (24), we adopt the direct method of trajectory optimization [27]. Specifically, the optimization simultaneously searches over the trajectories $\boldsymbol{x}_{0:T}$, $\boldsymbol{u}_{0:T-1}$, $\boldsymbol{\lambda}_{0:T-1}$ by treating the LCS and trust region as the separate constraints imposed at each time step. We solve such nonlinear optimization using CasADi [28] interface packed with IPOPT solver [45]. In our later applications, as the reduced-order LCS in (24) has a relatively small number of hybrid modes, e.g., $\dim \boldsymbol{\lambda} \leq 5$ and a small MPC horizon $T = 5$, we can solve (24) with a real-time MPC performance (e.g., MPC running frequency can reach 50Hz).

We summarize the algorithm of task-driven hybrid model reduction in Algorithm 1. Here, subscript i denotes the learning iteration. At initialization, the Rollout Buffer $\boldsymbol{\mathcal{D}}_{\text{buffer},0}$ can be filled with data collected from running random policies on the robot $\boldsymbol{f}()$.

Algorithm 1: Task-driven hybrid model reduction

```
Initialization: Initial reduced-order LCS model $\boldsymbol{g}_{\boldsymbol{\theta}_0}$;
                Initial Buffer $\boldsymbol{\mathcal{D}}_{\text{buffer},0}$ (by random policy);
                Trust region parameter schedule $\{\eta_i\}$
for $i = 0, 1, 2, \cdots$ do
    /* Reduced-order model update                    */
    Train reduced-order LCS $\boldsymbol{g}_{\boldsymbol{\theta}_i}$ with the data from current
      Rollout Buffer $\boldsymbol{\mathcal{D}}_{\text{buffer},i}$: $\boldsymbol{g}_{\boldsymbol{\theta}_{i+1}} \leftarrow \boldsymbol{g}_{\boldsymbol{\theta}_i}$ [Section V-A]

    /* Set the trust region                          */
    Set the trust region from the current Rollout Buffer
      $\boldsymbol{\mathcal{D}}_{\text{buffer},i}$: $[\bar{\boldsymbol{u}}_i - \boldsymbol{\Delta}_i, \bar{\boldsymbol{u}}_i + \boldsymbol{\Delta}_i]$ [see (25) and (26)] ;

    /* MPC rollout and update Buffer                 */
    With the current LCS $\boldsymbol{g}_{\boldsymbol{\theta}_{i+1}}$ and current trust region
      $[\bar{\boldsymbol{u}}_i - \boldsymbol{\Delta}_i, \bar{\boldsymbol{u}}_i + \boldsymbol{\Delta}_i]$, run the trust-region LCS MPC
      policy in (24) on the robot, collect new rollout data
      $\{(\boldsymbol{x}_k^{\boldsymbol{f}}, \boldsymbol{u}_k^{\boldsymbol{g}\text{-MPC}}, \boldsymbol{x}_{k+1}^{\boldsymbol{f}})\}$ and add it to Rollout Buffer:
      $\boldsymbol{\mathcal{D}}_{\text{buffer},i+1} \leftarrow \boldsymbol{\mathcal{D}}_{\text{buffer},i} \cup \{(\boldsymbol{x}_k^{\boldsymbol{f}}, \boldsymbol{u}_k^{\boldsymbol{g}\text{-MPC}}, \boldsymbol{x}_{k+1}^{\boldsymbol{f}})\}$;
end
```

VI. Model Reduction for Hybrid Control Systems

In this section, we will use the proposed method to solve model reduction of synthetic hybrid systems of varying dimension. Examples are written in Python, available at https://github.com/wanxinjin/Task-Driven-Hybrid-Reduction.

A. Problem Setting

Consider the MPC of a general PWA system [19]

$$\begin{aligned}
\min_{\substack{\boldsymbol{u}_{0:T-1}\\ \boldsymbol{x}_{0:T}}} \quad & J = \sum_{t=0}^{T-1} c(\boldsymbol{x}_t, \boldsymbol{u}_t) + h(\boldsymbol{x}_T)\\
\text{s.t.} \quad & \boldsymbol{f}(): \begin{cases} \boldsymbol{x}_{t+1} = A_j \boldsymbol{x}_t + B_j \boldsymbol{u}_t + \boldsymbol{c}_j, \ \ (\boldsymbol{x}_t, \boldsymbol{u}_t) \in \mathcal{P}_j, \\ \mathcal{P}_j = \{(\boldsymbol{x}, \boldsymbol{u}) \,|\, D_j \boldsymbol{x} + E_j \boldsymbol{u} + \boldsymbol{h}_j \leq \boldsymbol{0}\}, \\ j \in \{1, 2, \ldots, I\}, \end{cases}\\
& \boldsymbol{x}_0 \ \text{given}.
\end{aligned} \tag{27}$$

Here, $\mathcal{P}_j$, $j \in \{1, 2, \ldots, I\}$, is the j-th partition of the state-input space, with dynamics $\boldsymbol{x}_{t+1} = A_j \boldsymbol{x}_t + B_j \boldsymbol{u}_t + \boldsymbol{c}_j$. The total number of hybrid modes of the above PWA system is I. Solving (27) is generally treated as a mixed-integer program with I^T possible mode sequences. This exponential scaling quickly becomes computationally intractable as I and T grow.

In the following, we aim to find a reduced-order LCS model $\boldsymbol{g}()$ which maintains a small budget of hybrid modes, such that running $\boldsymbol{g}$-MPC can achieve similar performance as running the full-order $\boldsymbol{f}$-MPC in (27). Here, the reduced-order LCS $\boldsymbol{g}()$ in (8) has the same dimensions of $\boldsymbol{x}$ and $\boldsymbol{u}$ as $\boldsymbol{f}()$, but we set $\dim \boldsymbol{\lambda}$ such that its maximum number of hybrid modes of $\boldsymbol{g}()$ is far less than $\boldsymbol{f}()$'s, i.e., $2^{\dim \boldsymbol{\lambda}} \ll I$.

B. Experiment Settings

We consider the task of stabilizing the hybrid system to a stationary state (zeros), and thus set the cost function J in (27) as a quadratic cost function:

$$J = \sum_{t=0}^{T-1} \big(\boldsymbol{x}_t^{\mathsf{T}} Q \boldsymbol{x}_t + \boldsymbol{u}_t^{\mathsf{T}} R \boldsymbol{u}_t\big) + \boldsymbol{x}_T^{\mathsf{T}} Q_T \boldsymbol{x}_T, \tag{28}$$

with all weight matrices being identities. We run both $\boldsymbol{f}$-MPC and $\boldsymbol{g}$-MPC policies on full-order dynamics $\boldsymbol{f}()$ in a closed-loop (receding) fashion, though noting that $\boldsymbol{f}$-MPC cannot be solved in real-time for our more complex examples. The initial state $\boldsymbol{x}_0$ of the full-order dynamics $\boldsymbol{f}()$ is subject to a uniform distribution $\boldsymbol{x}_0 \sim U[-4, 4]$.

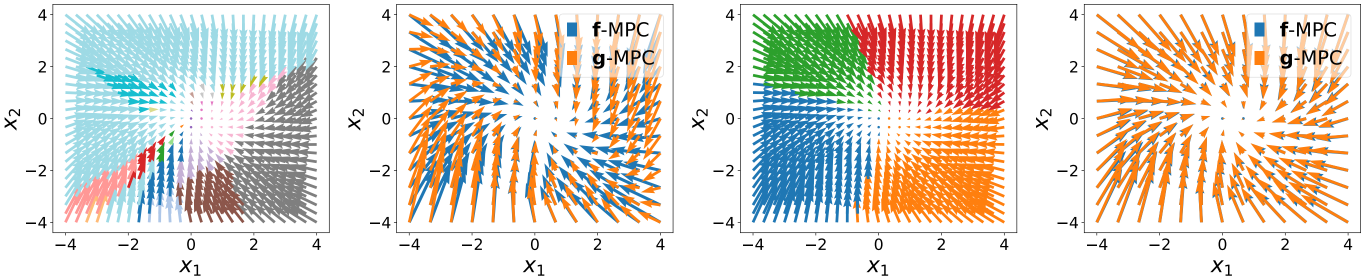
In Algorithm 1, the hyperparameters are listed in Table I. An ablation study about how the hyperparameters influence the performances will be given later in Section VI-D. For the hyperparameter setting in learning LCS, please refer to our previous paper [41].

TABLE I: Algorithm hyperparameters for model reduction of synthetic hybrid control systems.

Parameter[1]	**Symbol**	**Value**
MPC horizon	T	5
Rollout horizon	H	$15 \sim 20$
# of new rollouts added to buffer per iter.	R_{new}	5
Maximum buffer size	R_{buffer}	50 rollouts
Trust region hyperparameter	η_i	20, $\forall i$
Initial guess $\boldsymbol{\theta}$ in (22) for $\boldsymbol{g}()$	$\boldsymbol{\theta}_0$	$U[-0.5, 0.5]$[2]

[1] Other settings not listed here will be stated in text.

[2] $U[-0.5, 0.5]$ means uniform distribution in range $[-0.5, 0.5]$.



(a) Phase portrait for $\boldsymbol{f}$-MPC (b) $\boldsymbol{f}$-MPC vs. $\boldsymbol{g}$-MPC at Iter. 0 (c) Phase for $\boldsymbol{g}$-MPC at Iter. 24 (d) $\boldsymbol{f}$-MPC vs. $\boldsymbol{g}$-MPC at Iter. 24

Fig. 2: Phase portraits of the MPC-controlled full-order dynamics $\boldsymbol{x}_{t+1} = \boldsymbol{f}\big(\boldsymbol{x}_t, \text{MPC}(\boldsymbol{x}_t)\big)$, where the controller $\boldsymbol{u}_t = \text{MPC}(\boldsymbol{x}_t)$ can be either full-order $\boldsymbol{f}$-MPC or reduced-order $\boldsymbol{g}$-MPC. (a) is the phase portrait for the $\boldsymbol{f}$-MPC controller, where different colors indicate different hybrid modes (42 modes here) in $\boldsymbol{f}()$; (b) is the phase comparison between using $\boldsymbol{f}$-MPC and $\boldsymbol{g}$-MPC controllers at learning iteration 0; (c) is the phase portrait for $\boldsymbol{g}$-MPC controller at learning iteration 24, where different colors indicates different hybrid modes (4 modes here) in $\boldsymbol{g}()$; and (f) is the phase comparison between the $\boldsymbol{f}$-MPC controller and $\boldsymbol{g}$-MPC controller at learning iteration 24.

C. Results and Analysis

1) Illustration of Learning Progress: We randomly generate full-order dynamics $\boldsymbol{f}()$ in (27). Specifically, all matrices $(A_j, B_j, \boldsymbol{c}_j, D_j, E_j, \boldsymbol{h}_j)$, $i = 1, 2, \dots, I$, are sampled from uniform distributions, with dimension $\boldsymbol{x} \in \mathbb{R}^2$ and $\boldsymbol{u} \in \mathbb{R}$, and mode count $I \approx 120$ for random sampling of $\boldsymbol{x}_0 \sim U[-4, 4]$ and $\boldsymbol{u} \sim U[-10, 10]$. In the reduced-order LCS $\boldsymbol{g}()$ in (8), we take $\dim \boldsymbol{\lambda} = 2$, meaning that the maximum number of modes in $\boldsymbol{g}()$ is 4, far fewer than I of the full-order dynamics.

We plot the learning progress (iteration) for the task-driven reduced-order model $\boldsymbol{g}()$ in Fig. 2. Here, we show the phase portraits of the MPC-controlled full-order dynamics:

$$\boldsymbol{x}_{t+1} = \boldsymbol{f}\big(\boldsymbol{x}_t, \boldsymbol{u}_t\big) = \boldsymbol{f}\big(\boldsymbol{x}_t, \text{MPC}(\boldsymbol{x}_t)\big) \tag{29}$$

where the MPC controller $\boldsymbol{u}_t = \text{MPC}(\boldsymbol{x}_t)$ can be either the full-order $\boldsymbol{f}$-MPC (27) or the learned reduced-order $\boldsymbol{g}$-MPC. Specifically, Fig. 2a shows the phase portrait of $\boldsymbol{f}$-MPC controller, where different colors show different hybrid modes in $\boldsymbol{f}()$. Fig. 2b shows the phase portrait comparisons between $\boldsymbol{f}$-MPC controller (blue) and $\boldsymbol{g}$-MPC controller (orange) before learning. Fig. 2c shows the phase portrait for $\boldsymbol{g}$-MPC controller after learning, where different modes in $\boldsymbol{g}$-MPC are shown in different colors. Fig. 2d compares the phase plot between $\boldsymbol{f}$-MPC (blue) and $\boldsymbol{g}$-MPC (orange) controllers after learning.

Although the full-order dynamics $\boldsymbol{f}()$ has around $I = 120$ modes for random data $\boldsymbol{x} \sim U[-4, 4]$ and $\boldsymbol{u} \sim U[-10, 10]$, Fig. 2a shows 42 hybrid modes in $\boldsymbol{f}()$ with $\boldsymbol{f}$-MPC controller. One can notice that some modes correspond to a small portion of the state space, e.g., orange and green (near origin), and thus, most of $\boldsymbol{f}$'s task-relevant motion (flows) will not enter into or quickly pass those modes. This makes those modes less important for the task of minimizing (28). On the other hand, some other modes account for a large portion of the state space, such as cyan and gray. Most of $\boldsymbol{f}$'s motion will enter into or stay in those modes, making them dominant for the minimizing the task cost (28). In Fig. 2c, after learning, the reduced-order model $\boldsymbol{g}()$ has only 4 hybrid modes (recall $\dim \boldsymbol{\lambda} = 2$), which successfully capture the important modes in Fig. 2a. Comparing the phase portrait of the full-order $\boldsymbol{f}$-MPC controller and that of the reduced-order $\boldsymbol{g}$-MPC controller in Fig. 2d, we see a similar control performance. Thus, one can conclude that the proposed method learns a task-driven reduced-order model for the hybrid system.

2) High Dimensional Examples: In this session, we quantitatively analyze task-driven hybrid model reduction. For easy comparison, we represent the full-order dynamics $\boldsymbol{f}()$ in (27) also in LCS representation. All matrices in $\boldsymbol{f}()$ are drawn from uniform distribution. We use $\boldsymbol{\Lambda}$ to denote the complementarity variable of $\boldsymbol{f}()$. In the reduced-order LCS $\boldsymbol{g}()$, we vary $\dim \boldsymbol{\lambda}$ to show the effect of varying degrees of mode reduction.

In Table II, we consider different full-order $\boldsymbol{f}()$, listed in the second column, and different hybrid mode reduction, listed in the third column. From the fourth to ninth columns, we use the following metrics to report the learning performance.

- **Random Policy, number of modes in $\boldsymbol{f}$:** This is the total number of the hybrid modes that are active in the full-order dynamics $\boldsymbol{f}()$, when one runs a random policy with input $\boldsymbol{u}^{\text{rand}} \sim U[-10, 10]$ and initial condition $\boldsymbol{x}_0 \sim U[-4, 4]$. This metric indicates all possible modes experienced by the full-order system in a uniformly sampled state-input space.
- **Random Policy, ME$(\boldsymbol{g})(\%)$:** This is the relative prediction error of the learned reduced-order LCS model $\boldsymbol{g}()$ evaluated on the above random policy data, defined as

$$\frac{\|\boldsymbol{g}(\boldsymbol{x}, \boldsymbol{u}^{\text{rand}}) - \boldsymbol{f}(\boldsymbol{x}, \boldsymbol{u}^{\text{rand}})\|^2}{\|\boldsymbol{f}(\boldsymbol{x}, \boldsymbol{u}^{\text{rand}})\|^2 + 10^{-6}} \times 100\%. \tag{30}$$

- **$\boldsymbol{g}$-MPC Policy, number of modes in $\boldsymbol{f}$:** This is the total number of the hybrid modes that are active in the full-order dynamics $\boldsymbol{f}()$ when one runs the learned reduced-order $\boldsymbol{g}$-MPC controller on it with initial condition $\boldsymbol{x}_0 \sim U[-4, 4]$ This metric indicates all possible modes experienced by the full-order system in the task-relevant state-input space.
- **$\boldsymbol{g}$-MPC Policy, ME$(\boldsymbol{g})(\%)$:** This is the relative prediction error of the learned reduced-order LCS $\boldsymbol{g}()$ evaluated at the above $\boldsymbol{g}$-MPC policy data, defined as

$$\frac{\|\boldsymbol{g}(\boldsymbol{x}, \boldsymbol{u}^{\boldsymbol{g}\text{-MPC}}) - \boldsymbol{f}(\boldsymbol{x}, \boldsymbol{u}^{\boldsymbol{g}\text{-MPC}})\|^2}{\|\boldsymbol{f}(\boldsymbol{x}, \boldsymbol{u}^{\boldsymbol{g}\text{-MPC}})\|^2 + 10^{-6}} \times 100\%. \tag{31}$$

As the proposed algorithm chooses to minimize the model error, which is related to the performance gap as in Lemma 1. Thus it is meaningful to include this metric to show the model error of the learned $\boldsymbol{g}()$ on the task-relevant data.

TABLE II: Task-driven model reduction for hybrid control systems

Case	System dimension	Mode reduction $\dim \boldsymbol{\Lambda} \rightarrow \dim \boldsymbol{\lambda}$	**Random Policy**		**$\boldsymbol{g}$-MPC Policy**			$\mathcal{L}(\boldsymbol{g})$ (%)
			# of modes in $\boldsymbol{f}$	ME($\boldsymbol{g}$) (%)	# of modes in $\boldsymbol{f}$	ME($\boldsymbol{g}$) (%)	# of modes in $\boldsymbol{g}$	
1	$\dim \boldsymbol{x} = 6$ $\dim \boldsymbol{u} = 2$	$\dim \boldsymbol{\Lambda} = 8$ $\rightarrow \dim \boldsymbol{\lambda} = 3$	187.3 ± 14.0	33.0% ± 13.9%	18.4 ± 2.8	0.5% ± 0.2%	6.2 ± 1.2	0.1% ± 0.1%
2	$\dim \boldsymbol{x} = 10$ $\dim \boldsymbol{u} = 3$	$\dim \boldsymbol{\Lambda} = 12$ $\rightarrow \dim \boldsymbol{\lambda} = 3$	1090.0 ± 133.2	29.8% ± 13.0%	29.9 ± 2.5	1.0% ±0.1%	6.7 ± 1.0	0.5% ±0.2%
3	$\dim \boldsymbol{x} = 20$ $\dim \boldsymbol{u} = 3$	$\dim \boldsymbol{\Lambda} = 15$ $\rightarrow \dim \boldsymbol{\lambda} = 1$	2686.2 ± 197.3	16.8% ± 5.7%	50.0 ± 4.7	2.1% ±0.3%	2.0 ± 0.0	1.1% ±0.5%
4	$\dim \boldsymbol{x} = 20$ $\dim \boldsymbol{u} = 3$	$\dim \boldsymbol{\Lambda} = 15$ $\rightarrow \dim \boldsymbol{\lambda} = 2$	2869.2 ± 165.0	17.5% ± 6.5%	52.3 ± 4.5	1.9% ±0.2%	3.7 ± 0.4	1.1% ±0.4%
5	$\dim \boldsymbol{x} = 20$ $\dim \boldsymbol{u} = 3$	$\dim \boldsymbol{\Lambda} = 15$ $\rightarrow \dim \boldsymbol{\lambda} = 3$	2855.7 ± 193.2	16.6% ± 4.6%	50.1 ± 3.9	1.9% ± 0.3%	7.1 ± 0.7	1.0% ± 0.4%
6	$\dim \boldsymbol{x} = 20$ $\dim \boldsymbol{u} = 3$	$\dim \boldsymbol{\Lambda} = 15$ $\rightarrow \dim \boldsymbol{\lambda} = 5$	2839.9 ± 172.9	16.3% ± 4.6%	54.3 ± 4.8	1.8% ± 0.2%	16.2 ± 3.3	0.9% ± 0.2%
7	$\dim \boldsymbol{x} = 30$ $\dim \boldsymbol{u} = 3$	$\dim \boldsymbol{\Lambda} = 15$ $\rightarrow \dim \boldsymbol{\lambda} = 3$	3232.6 ± 219.1	11.6% ± 4.7%	70.7 ± 7.3	2.3% ± 0.6%	7.5 ± 0.6	2.0% ± 0.7%

[*] Results for each case are based on 10 trials, and each trial uses a different randomly-generated full-order dynamics $\boldsymbol{f}()$. The results are reported using mean and standard derivation over all ten trials. See detailed explanations about those quantities in text.

- **$\boldsymbol{g}$-MPC Policy, number of modes in $\boldsymbol{g}$:** This is the total number of hybrid modes that are active inside the $\boldsymbol{g}$-MPC controller, which is run on the full-order dynamics $\boldsymbol{f}()$ with initial system condition $\boldsymbol{x}_0 \sim U[-4, 4]$.
- **Relative task performance gap $\mathcal{L}(\boldsymbol{g})(\%)$:** This is the relative task performance gap $\mathcal{L}(\boldsymbol{g})(\%)$ is between $\boldsymbol{g}$-MPC controller and $\boldsymbol{f}$-MPC controller:

$$\mathcal{L}(\boldsymbol{g}) = \frac{J(\boldsymbol{g}\text{-MPC}) - J(\boldsymbol{f}\text{-MPC})}{J(\boldsymbol{f}\text{-MPC})} \times 100\%, \qquad (32)$$

where $J(\boldsymbol{g}\text{-MPC})$ is the cost of a rollout by running $\boldsymbol{g}$-MPC controller on the full-order dynamics $\boldsymbol{f}()$, namely,

$$J(\boldsymbol{g}\text{-MPC}) = \mathbb{E}_{\boldsymbol{\beta} \sim p(\boldsymbol{\beta})} \mathbb{E}_{p_{\boldsymbol{\beta}}(\boldsymbol{x}_0)} \sum_{t=0}^{H} c_{\boldsymbol{\beta}}(\boldsymbol{x}_t^{\boldsymbol{f}}, \boldsymbol{u}_t^{\boldsymbol{g}\text{-MPC}}), \qquad (33)$$

with $\{\boldsymbol{x}_{0:H}^{\boldsymbol{f}}, \boldsymbol{u}_{0:H}^{\boldsymbol{g}\text{-MPC}}\}$ being a rollout of running $\boldsymbol{g}$-MPC controller on $\boldsymbol{f}()$. The similar definition applies to $J(\boldsymbol{f}\text{-MPC})$. Recall the original learning loss (13). $\mathcal{L}(\boldsymbol{g})(\%)$ can directly indicate the performance of the learned reduced-order model $\boldsymbol{g}()$ for the given task distribution $p(\boldsymbol{\beta})$.

The results in Table II clearly show a reliable performance of the proposed method. For example, in Case 5, the full-order $\boldsymbol{f}()$ has $\boldsymbol{x} \in \mathbb{R}^{20}$ and $\boldsymbol{u} \in \mathbb{R}^{3}$ has $\boldsymbol{\Lambda} \in \mathbb{R}^{15}$. With a random policy run on $\boldsymbol{f}()$, the number of active hybrid modes in $\boldsymbol{f}()$ is around 2.8k. The proposed algorithm learns a task-driven reduced-order LCS $\boldsymbol{g}()$ only with around 7 modes ($\dim \boldsymbol{\lambda} = 3$). The resulting reduced-order $\boldsymbol{g}$-MPC controller running on the full-order $\boldsymbol{f}()$ only has around 1% performance loss relative to running the full-order $\boldsymbol{f}$-MPC controller. The relative prediction error of the learned reduced-order LCS $\boldsymbol{g}()$ is less than 2% on the on-policy ($\boldsymbol{g}$-MPC) data, while is 16.6 % on the random policy data. Also, when run with $\boldsymbol{g}$-MPC controller, full-order $\boldsymbol{f}()$ has around 50 active modes. Table II also shows that the proposed algorithm can handle high-dimensional system, such as $\boldsymbol{x} \in \mathbb{R}^{30}$. Additionally, corresponding to Case 7 in Table II, a single instance of $\boldsymbol{f}()$'s rollout with the full-order $\boldsymbol{f}$-MPC controller and with the reduced-order $\boldsymbol{g}$-MPC controller are compared in Fig. 3. Based on the results in Table II, we have the following conclusions.

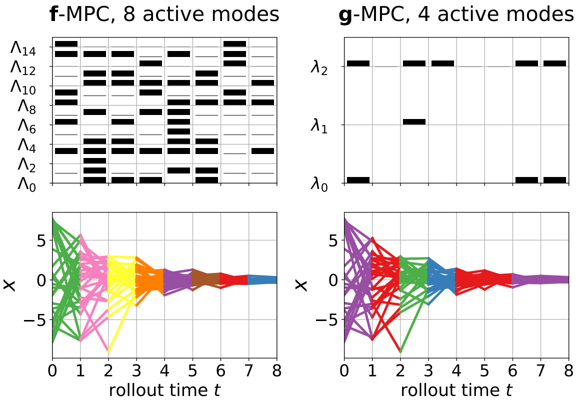
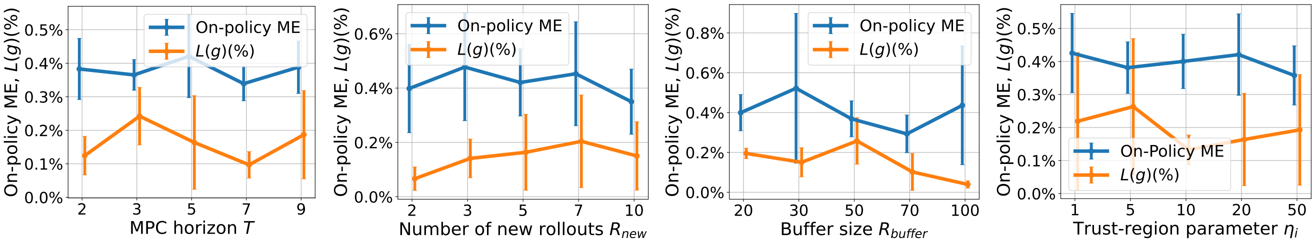


Fig. 3: An example rollout of $\boldsymbol{f}()$ with full-order $\boldsymbol{f}$-MPC controller or reduced-order $\boldsymbol{g}$-MPC controller, corresponding to Case 7 in Table II. Specifically, the left column is a single rollout of running $\boldsymbol{f}$-MPC controller on $\boldsymbol{f}()$, and the right running $\boldsymbol{g}$-MPC controller on $\boldsymbol{f}()$, both under the same initial condition. The upper row shows the activation of $\boldsymbol{\Lambda}$ or $\boldsymbol{\lambda}$ over time (black brick means nonzero and blank means zero). The bottom row shows the state trajectory $\boldsymbol{x}_{t+1} = \boldsymbol{f}\big(\boldsymbol{x}_t, \text{MPC}(\boldsymbol{x}_t)\big)$ over time, with each color representing a different hybrid mode. Note that since each panel only shows a single instance of rollout (from a fixed $\boldsymbol{x}_0$), there are not many active hybrid modes of $\boldsymbol{f}$ involved in a single trajectory.

(i) Jointly looking at the number of modes in $\boldsymbol{f}$ with random policy (fourth column), the number of modes inside $\boldsymbol{g}$-MPC policy (eighth column), and the relative performance gap (last column), one can clearly see that the proposed algorithm can find a reduced-order model with *multiple orders of magnitude fewer hybrid modes* than the full-order $\boldsymbol{f}()$, and it can result in a similar MPC control performance as using full-order MPC policy, with a performance loss less than $2\% - 3\%$.



(a)

(b)

(c)

(d)

Fig. 4: Ablation study about the effect of hyperparameter values on the algorithm performance. We here use the performance metrics **On-policy ME** in (31) and relative task performance gap $\mathcal{L}(\boldsymbol{g})(\%)$ in (32) to report the algorithm performance. The experimenting system's dimensions and other settings follow Case 1 in Table II. Each result is reported based on ten trials, and each trial uses a different randomly-generated full-order LCS $\boldsymbol{f}()$, as stated in the previous session. The error bars represent the standard deviation across all trials.

(ii) Comparison between the mode counts in $\boldsymbol{f}$ with random policy (fourth column) and that with $\boldsymbol{g}$-MPC policy (sixth column) can confirm the motivating hypothesis of this paper: a much fewer hybrid modes are actually necessary to achieve the task (here, the task is to minimize the given cost function in (28)), and the vast majority of the hybrid modes in $\boldsymbol{f}$ will remain untouched throughout the control process.

(iii) Notably, comparing the relative model error of the learned reduced-order LCS $\boldsymbol{g}()$ on random policy data (fifth column) and only on the $\boldsymbol{g}$-MPC policy data (seventh column), one can conclude that the learned reduced-order LCS $\boldsymbol{g}()$ attains higher validity on the task-relevant data. This sufficiently shows the success of our task-driven hybrid model reduction.

All the above results and analysis clearly confirm that the effectiveness and efficiency of the proposed task-driven hybrid model reduction method. Also, attention needs to be paid to Cases 3-6. Here, under the same other conditions, we used an increasingly complex reduced-order LCS $\boldsymbol{g}$ from $\dim \boldsymbol{\lambda} = 1$ to $\dim \boldsymbol{\lambda} = 5$. The results show that increasing the hybrid mode budget in $\boldsymbol{g}()$ can lead to a performance improvement, although small, in the model accuracy (seventh column) and the relative task performance gap (last column).

D. Effect of Hyperparameter Settings

We conduct the ablation study to investigate the effect of hyperparameter settings in Table I on the algorithm performance. We still use the **On-Policy ME**$(\boldsymbol{g})(\%)$ in (31) and the relative task performance gap $\mathcal{L}(\boldsymbol{g})(\%)$ in (32) to report the results. The dimensions of the full-order and reduced-order models and other settings follows Case 1 in Table II. The results are in Fig. 4. The results show that the learning performance is quite robust against a large range of hyperparameter values in Table I. Fig. 4c suggests that using a larger buffer size would slightly lower the final task performance gap, although not much improving the accuracy of the reduced-order model. Fig. 4d indicates that the choice of the trust region parameter η_i does not significantly influence the learning performance. Overall, Fig. 4 suggests that setting algorithm hyperparameters is not difficult in practice.

VII. Three-Finger Dexterous Manipulation

In this section, we will apply the proposed method to solve the three-fingered robotic hand manipulation [46]. The Python codes to reproduce all the following results are available at https://github.com/wanxinjin/Task-Driven-Hybrid-Reduction.

A. Three-Finger Robotic Hand Manipulation

As illustrated in Fig. 5, the three-finger robotic hand manipulation system includes three 3-DoF robotic fingers and a cube with a table. The goal is to find a control policy for the three fingers to moves the cube to given target poses. The entire simulation environment is based on MuJoCo physics engine [13]. This paper considers two specific tasks.

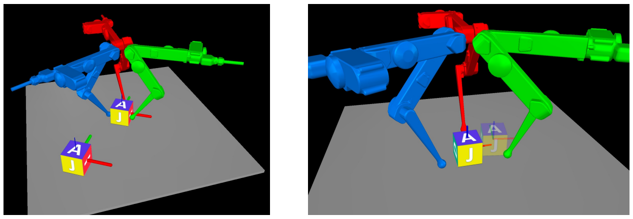


Fig. 5: Three-finger dexterous manipulation tasks. Left: the three robotic fingers need to turn a cube to a random target orientation, given by a reference in the left corner. Right: the three fingers need to move the cube to a target pose with random position and orientation, given by the shaded reference. The simulation environment uses MuJoCo physics engine [13].

Cube Turning Task: As shown in the left panel of Fig. 5, the cube has one degree of freedom (DoF) relative to the table: it can only rotate around a fixed vertical axis on the table. There is friction between the cube and table and also damping in the joint of cube rotation. Three fingers need to turn the cube to any random target orientation α^{goal}, sampled from a uniform distribution:

$$\boldsymbol{\beta} = \alpha^{\text{goal}} \sim U[-1.5,\ 1.5] \quad \text{(radius)} \tag{34}$$

For visualization, the target is shown in the bottom left corner.

Cube Moving Task: As shown in the right panel of Fig. 5, the cube has 6 DoFs relative to the table: it is a free object on the table. The three fingers need to move the cube to align it to any random target pose $\boldsymbol{\beta} = (\boldsymbol{p}^{\text{goal}}, \alpha^{\text{goal}})$ on the table, where $\boldsymbol{p}^{\text{goal}} \in \mathbb{R}^2$ (center of mass) is the cube's target xy-position and α^{goal} is the cube's target orientation angle on the table, both sampled from uniform distribution

$$\begin{aligned} \boldsymbol{p}^{\text{goal}} \sim U[-\boldsymbol{p}_{\max},\ \boldsymbol{p}_{\max}], \quad \boldsymbol{p}_{\max} = [0.06,\ 0.06]^{\mathsf{T}}(\text{m}), \\ \alpha^{\text{goal}} \sim U[-0.5,\ 0.5] \quad \text{(rad)}. \end{aligned} \tag{35}$$

The challenge of the above three-finger manipulation tasks lies in that the system contains a large number of potential contact interactions that need to reason about. For example, (i) the contact interaction between each finger and the cube has three modes: separate, stick, and slip; (ii) each fingertip needs to reason which of cube faces to contact with; (iii) the contact interaction between the table and cube also contains at least three modes. Thus, the full-order dynamics, although unknown, contains an estimated *thousands* of hybrid modes. Further, it will become even more challenging if one aims to perform real-time closed-loop control on the three-finger manipulation system to achieve the given tasks.

In this following, we will focus on solving the above three-finger manipulation tasks without any prior knowledge about the three-finger manipulation system, e.g., geometry, physical properties, etc. We will apply the proposed method to learn a task-driven reduced-order LCS for real-time closed-loop MPC on the three-finger manipulation to accomplish the above tasks. Note that different from our application to synthetic hybrid systems in previous section, we here do not have a true hybrid model $\boldsymbol{f}()$ (and $\boldsymbol{f}$-MPC) for ground truth comparison.

B. Experiment Settings

Before proceeding, we here clarify the state and input spaces in the reduced-order LCS model $\boldsymbol{g}()$ in (8) and define the cost functions for the above two manipulation tasks.

1) Reduced-Order LCS Model: We select the state space of the three-finger manipulation system as

$$\boldsymbol{x} = \begin{bmatrix}\boldsymbol{p}_{\text{cube}}, & \alpha_{\text{cube}}, & \boldsymbol{p}_{\text{fingertip}}\end{bmatrix}^{\mathsf{T}} \in \mathbb{R}^9, \tag{36}$$

where $\boldsymbol{p}_{\text{fingertip}} \in \mathbb{R}^6$ is the xy positions of the three fingertips, $\boldsymbol{p}_{\text{cube}} \in \mathbb{R}^2$ is the xy position (of center of mass) of the object, and α_{cube} is the planar orientation angle of the cube. The input space of the three-finger manipulation system is

$$\boldsymbol{u} = \Delta \boldsymbol{p}_{\text{fingertip}} \in \mathbb{R}^6, \tag{37}$$

which includes the incremental position of each fingertip. We use operational space control (OSC) [47] in the lower level to map from $\boldsymbol{u}$ to the joint torque of each finger, also the OSC controller regularizes the z (vertical) position of each fingertip to be constant. The OSC control frequency is 10 Hz. In the reduced-order LCS model $\boldsymbol{g}()$ in (8), we set

$$\dim \boldsymbol{\lambda} = 5 \tag{38}$$

for both manipulation tasks. This means that the reduced-order model can maximally represent $2^5 = 32$ hybrid modes, which are far fewer than the estimated thousands of modes in the full-order dynamics of the three-finger manipulation system. The selection of $\dim \boldsymbol{\lambda}$ will be discussed later in Section VII-E.

2) Cost Functions: For given manipulation tasks, we can simply define a continuous cost function as the sum of three terms: (i) the distance of the object to the goal, (ii) control effort, and (iii) the distance of the actuated fingers to the object (which is part of the state variable) to encourage the contact. Thus, we define the following quadratic cost function $J_{\boldsymbol{\beta}}$

$$J_{\boldsymbol{\beta}} = \sum\nolimits_{t=0}^{T-1} c_{\boldsymbol{\beta}}(\boldsymbol{x}_t, \boldsymbol{u}_t) + h_{\boldsymbol{\beta}}(\boldsymbol{x}_T), \quad \boldsymbol{\beta} \sim p(\boldsymbol{\beta}) \tag{39}$$

where $p(\boldsymbol{\beta})$ is (34) for the Cube Turning task and (35) for the Cube Moving task, and

$$\begin{aligned} c_{\boldsymbol{\beta}} =& w_1^c \|\boldsymbol{p}_{\text{fingertip}} - \boldsymbol{p}_{\text{cube}}\|^2 + w_2^c \|\boldsymbol{p}_{\text{cube}} - \boldsymbol{p}^{\text{goal}}\|^2 \\ &+ w_3^c (\alpha_{\text{cube}} - \alpha^{\text{goal}})^2 + 0.01\|\boldsymbol{u}\|^2, \\ h_{\boldsymbol{\beta}} =& w_1^h \|\boldsymbol{p}_{\text{fingertip}} - \boldsymbol{p}_{\text{cube}}\|^2 + w_2^h \|\boldsymbol{p}_{\text{cube}} - \boldsymbol{p}^{\text{goal}}\|^2 \\ &+ w_3^h (\alpha_{\text{cube}} - \alpha^{\text{goal}})^2. \end{aligned} \tag{40}$$

Here, the cost term $\|\boldsymbol{p}_{\text{fingertip}} - \boldsymbol{p}_{\text{cube}}\|^2$ penalizes the distance between the fingertips and center of the cube, i.e. encouraging contact between fingertips and cube; $\|\boldsymbol{p}_{\text{cube}} - \boldsymbol{p}^{\text{goal}}\|^2$ and $(\alpha_{\text{cube}} - \alpha^{\text{goal}})^2$ are the squared distance to the target position or orientation, respectively; and $0.01\|\boldsymbol{u}\|^2$ penalizes the control cost. $\boldsymbol{w}^c = [w_1^c, w_2^c, w_3^c]^{\mathsf{T}}$ and $\boldsymbol{w}^h = [w_1^h, w_2^h, w_3^h]^{\mathsf{T}}$ are the cost weights, whose values will be given later. An extended discussion of different choices of the cost weights will be given in later Section VII-E.

The other hyperparameters of Algorithm 1 are listed in the following table, which largely follows the ones in Table I for the previous synthetic system examples (recall that the discussion of the hyperparameter settings is in Section VI-D).

TABLE III: Hyperparameters for three-finger manipulation

Parameter[1]	Symbol	Value
MPC horizon	T	5
Rollout horizon	H	20
# of new rollouts added to buffer per iter.	R_{new}	5
Maximum buffer size	R_{buffer}	200 rollouts
Trust region parameter	η_i	1.0 $\forall i$
Initial guess $\boldsymbol{\theta}$ in (22) for $\boldsymbol{g}()$	$\boldsymbol{\theta}_0$	$U[-0.5, 0.5]$

[1] Other settings not listed here will be stated in text.

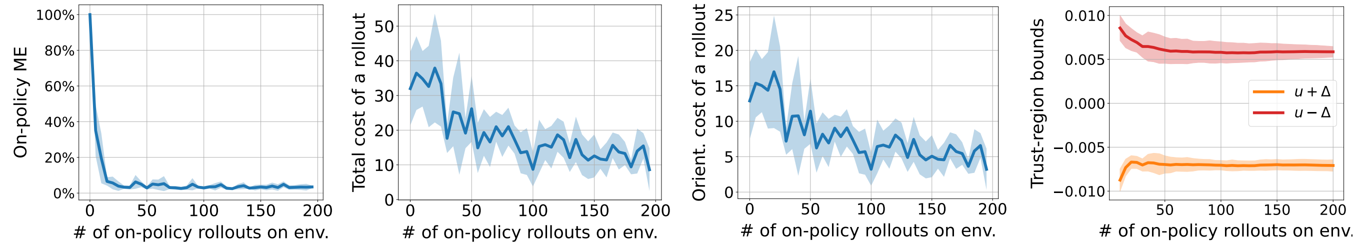
C. Cube Turning Task

This session presents the results and analysis for solving the Cube Turning manipulation task. In this task, we set cost function weights in (40) as

$$\begin{aligned} \boldsymbol{w}^c &= \begin{bmatrix}10.0 & 0.0 & 2.0\end{bmatrix}^{\mathsf{T}}, \\ \boldsymbol{w}^h &= \begin{bmatrix}2.0 & 0.0 & 10.0\end{bmatrix}^{\mathsf{T}}. \end{aligned} \tag{41}$$

The above weight values are not deliberately picked. In fact, the performance is not sensitive to the choice of $\boldsymbol{w}^c$ and $\boldsymbol{w}^h$. As will be discussed in Section VII-E, the similar learning and task performance permits a wide selection of weight values.

1) Results: The key learning curves are shown in Fig. 6, where each curve is the average of five random-seed trials and the shaded area indicates the standard deviation. In all panels, x-axis shows the total number of on-policy ($\boldsymbol{g}$-MPC controller) rollouts of the environment, which is proportional to the learning iteration (i.e., each iteration collects 5 new on-policy rollouts). Specifically, Fig. 6a shows the relative prediction error of the reduced-order LCS $\boldsymbol{g}()$ evaluated on the on-policy rollout data, defined in (31), where $\boldsymbol{f}(\boldsymbol{x}, \boldsymbol{u}^{\boldsymbol{g}\text{-MPC}})$ is a direct observation of the next state of the environment. Fig. 6b shows the total cost of a rollout from running $\boldsymbol{g}$-MPC controller on the environment, defined in (33). Fig. 6c shows



(a) (b) (c) (d)

Fig. 6: Learning curves of the three-finger manipulation system for the Cube Turning task. Each curve is the average of five random seeds, and its shaded area shows the standard deviation. All results here are shown on an on-policy rollout basis, and each on-policy rollout is a result of running the reduced-order $\boldsymbol{g}$-MPC controller on the three-finger system (environment), i.e., the (unknown) full-order $\boldsymbol{f}()$. Detailed explanations are given in text.

the orientation cost of a rollout from running $\boldsymbol{g}$-MPC controller in the environment, defined as

$$E_{\boldsymbol{\beta}\sim p(\boldsymbol{\beta})}\,\mathbb{E}_{\boldsymbol{x}\sim p_{\boldsymbol{\beta}}(\boldsymbol{x}_0)}\sum\nolimits_{t=0}^{H}(\alpha_{\text{cube},t}-\alpha^{\text{goal}})^2. \tag{42}$$

Fig. 6b and Fig. 6c show the very similar pattern because the orientation cost term $(\alpha_{\text{cube}}-\alpha^{\text{goal}})^2$ dominates in (40) relative to other cost terms in scale. Fig. 6d shows the trust region upper bound $\bar{\boldsymbol{u}}+\boldsymbol{\Delta}$ and lower bound $\bar{\boldsymbol{u}}-\boldsymbol{\Delta}$ for the first component in the control input vector.

Some quantitative results that are not have shown in Fig. 6 are given in Table IV. In the second row of Table IV, the cube's terminal orientation error $|\alpha_{\text{cube},H}-\alpha^{\text{goal}}|$ is calculated at the end (at time step H) of a rollout. In the last row of Table IV, we report the robustness of the $\boldsymbol{g}$-MPC controller against external disturbance torques added to the cube during $\boldsymbol{g}$-MPC policy rollout. Here, we apply a 3D external disturbance torque, sampled from $\boldsymbol{\tau}_{\text{disturb}}\sim U[-\tau^{\text{mag}}_{\text{disturb}},\tau^{\text{mag}}_{\text{disturb}}]$, during each time interval (0.1s) of rollout steps. We increase the disturbance magnitude $\tau^{\text{mag}}_{\text{disturb}}$ until the resulting $\boldsymbol{g}$-MPC rollout has an average cube terminal orientation error $|\alpha_{\text{cube},H}-\alpha^{\text{goal}}|\geq 0.3$ (rad). We report the result by calculating the maximum angular acceleration of disturbance: $\frac{\tau^{\text{mag}}_{\text{disturb}}}{I_{\text{cube}}}$, with I_{cube} the cube inertia.

TABLE IV: Performance of learned $\boldsymbol{g}$-MPC for Cube Turning

Results	Value
Total number of hybrid modes in $\boldsymbol{g}$-MPC	(around) 14
Cube terminal orientation error $\lvert\alpha_{\text{cube},H}-\alpha^{\text{goal}}\rvert$	0.06 ± 0.02 (rad)
Terminal error (relative) $\frac{(\alpha_{\text{cube},H}-\alpha^{\text{goal}})^2}{(\alpha^{\text{goal}})^2}$	3.4% ± 2.3%
Total training time of the algorithm[1]	4.1 ± 0.1 mins
Total # of environment samples in training	4k
Running frequency of reduced-order $\boldsymbol{g}$-MPC[1]	>50 Hz
% of stick-slip-separate modes in rollouts	(approx.) $>70\%$[2]
Maximum $\frac{\tau^{\text{mag}}_{\text{disturb}}}{I_{\text{cube}}}$ until $\lvert\alpha_{\text{cube},H}-\alpha^{\text{goal}}\rvert\geq 0.3$	5000 rad/s^2

[1] The experiments are tested on MacBook Pro with M1 Pro chip.
[2] This is approximately calculated by observing the environment rollouts with the learned reduced-order $\boldsymbol{g}$-MPC.

Fig. 6 and Table IV show the efficiency of the proposed method to successfully solve the three-finger dexterous manipulation for the Cube Turning task. Particularly, we have the following conclusions.

(i) The proposed algorithm learns a reduced-order model to solve the three-finger manipulation of Cube Turning task without any prior knowledge within just 5 minutes of wall-clock time, including real-time closed-loop control on the manipulation system. It only collects around 4k (200 rollouts × 20 steps/rollout) data points from the environment.

(ii) The learned task-driven reduced-order LCS $\boldsymbol{g}()$ leads to a closed-loop MPC controller on the three-finger manipulation system, achieving a high accuracy: the cube terminal orientation error $|\alpha_{\text{cube},H}-\alpha^{\text{goal}}|<0.08$ (rad) and the relative orientation error $\frac{(\alpha_{\text{cube},H}-\alpha^{\text{goal}})^2}{(\alpha^{\text{goal}})^2}<5\%$.

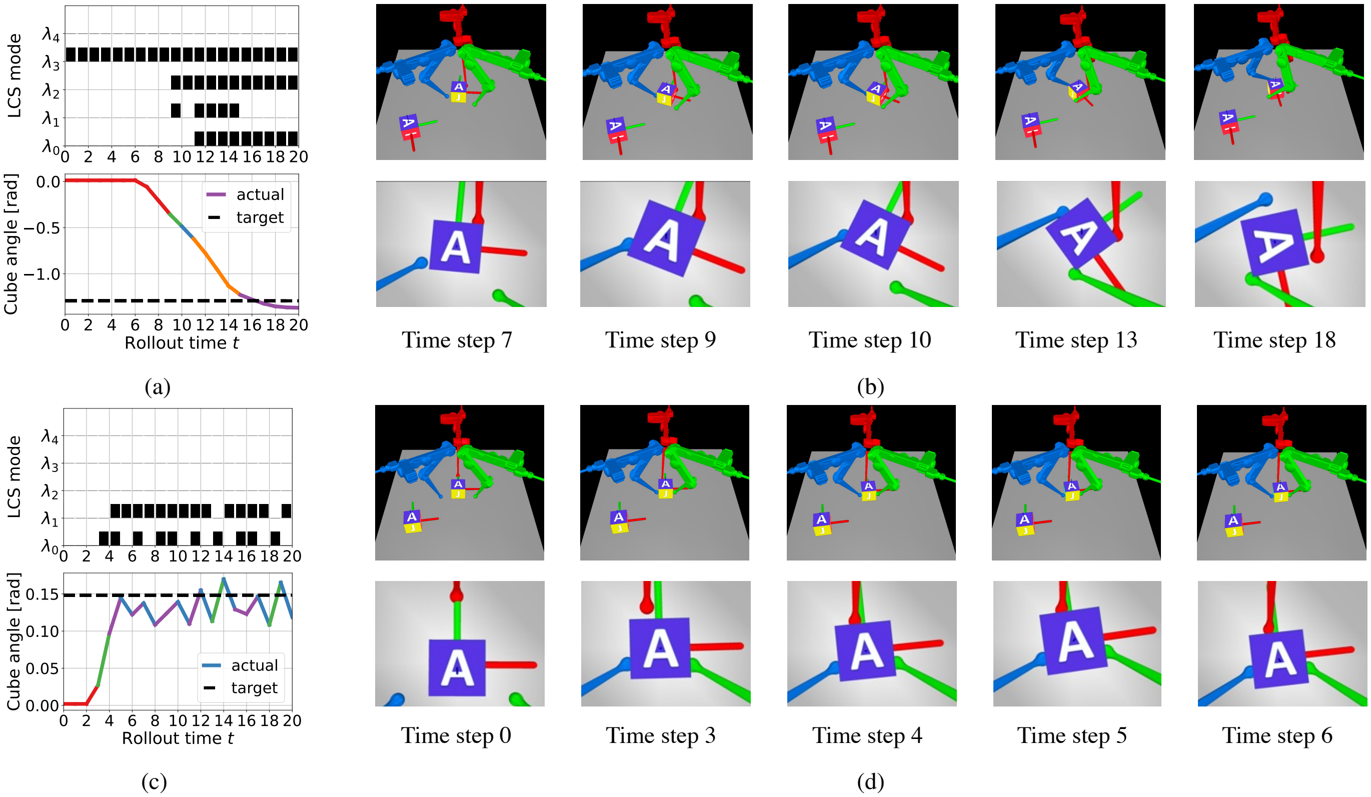
(iii) The learned LCS $\boldsymbol{g}()$ results in a $\boldsymbol{g}$-MPC, which enables real-time closed-loop control on the three-finger manipulation system to achieve the task. The running frequency of $\boldsymbol{g}$-MPC is more than 50Hz.

(iv) The learned reduced-order LCS $\boldsymbol{g}()$ maximally contains 32 modes, and around 14 of them are active. Those 14 hybrid modes enables rich contact interactions, including `separate`, `stick`, and `slip` between the fingertips and the cube, and `stick`, `CCW rotational slip`, and `CW rotational slip` between the cube and the table, happening at different time steps. More than 70% of rollouts contains the sequence of `stick-slip-separate` modes. More detailed explanations will be given in the next session.

(v) The reduced-order $\boldsymbol{g}$-MPC controller shows high robustness against large external torque disturbances. The robustness is a natural benefit of the closed-loop implementation of $\boldsymbol{g}$-MPC controller. Such a high robust performance could also partially due to the high stiffness gain in our lower-level OSC.

2) Analysis of Reduced-Order Hybrid Modes: In this session, we will detail how the learned task-driven reduced-order LCS $\boldsymbol{g}()$ enables the three-finger system to reason about the contact decision in the Cube Tuning task. In each row of Fig. 7, given a random target orientation, we show the rollout trajectories (left) and key-time-step snapshots (right) of the three-finger manipulation environment by running the learned $\boldsymbol{g}$-MPC. The rollout horizon $H=20$.

Specifically, in Fig. 7a and 7b, the target orientation is $\alpha^{\text{goal}}=-1.29$ (rad), shown by a reference at the lower left corner. The upper panel in Fig. 7a shows the mode activation of each dimension of $\boldsymbol{\lambda}$ in $\boldsymbol{g}$-MPC. Here, the black bricks mean $\boldsymbol{\lambda}>0$, and blank means $\boldsymbol{\lambda}=0$. For exposition simplicity, we use $\texttt{sign}(\boldsymbol{\lambda})$ to denote the mode activation. For example, at



(a)

Time step 7 Time step 9 Time step 10 Time step 13 Time step 18

(b)

(c)

Time step 0 Time step 3 Time step 4 Time step 5 Time step 6

(d)

Fig. 7: Two rollouts of running the learned reduced-order $\boldsymbol{g}$-MPC controller on the three-finger system. (a) and (b): the target orientation is $\alpha^{\text{goal}} = -1.29$ (rad). (c) and (d): $\alpha^{\text{goal}} = -0.148$. The upper panel of (a) or (c) shows the mode activation $\texttt{sign}(\boldsymbol{\lambda})$ in $\boldsymbol{g}()$ over rollout time. Here, black bricks show $\boldsymbol{\lambda} > 0$ and blank $\boldsymbol{\lambda} = 0$. The bottom panel of (a) or (c) shows the trajectory of cube angle $\alpha_{\text{cube},t}$, where different mode activation are indicated by different colors. (b) or (d) shows the key-time-step snapshots of the simulator, corresponding to (a) or (c), respectively. Here, the upper panels show the environment snapshots, and lower panels show the zoom-in details. Analysis are given in text and Tables V and VI. Note that in (b) and (d) we have attached a green-red body coordinate frame to the cube only for visualization purposes (i.e., the coordinate frame does not affect the physical contact interaction).

TABLE V: Empirical correspondence between LCS mode activation in Fig. 7a and physical contact interaction in Fig.7b.

Time t	**Mode activation in $\boldsymbol{g}()$**	**Interaction between fingertips and cube**[1]	**Interaction between cube and table**[1]
$t = 0, 1, .., 8$	$\texttt{sign}(\boldsymbol{\lambda}) = [0, 0, 0, 1, 0]^{\mathsf{T}}$	(`R, G, B separate`) or (`G, B separate` and `R touching`)	`Cube stick to table` or `CW rotational slip`
$t = 9$	$\texttt{sign}(\boldsymbol{\lambda}) = [0, 1, 1, 1, 0]^{\mathsf{T}}$	`G separate` and `R, B touching`	`CW rotational slip`
$t = 10$	$\texttt{sign}(\boldsymbol{\lambda}) = [0, 0, 1, 1, 0]^{\mathsf{T}}$	`R right slip` and `G separate` and `B stick`	`CW rotational slip`
$t = 11, ...14$	$\texttt{sign}(\boldsymbol{\lambda}) = [1, 1, 1, 1, 0]^{\mathsf{T}}$	`R, G, B touching`	`CW rotational slip`
$t = 15, ..., 20$	$\texttt{sign}(\boldsymbol{\lambda}) = [1, 0, 1, 1, 0]^{\mathsf{T}}$	`G touching` and `R, B separate`	`CW rotational slip`

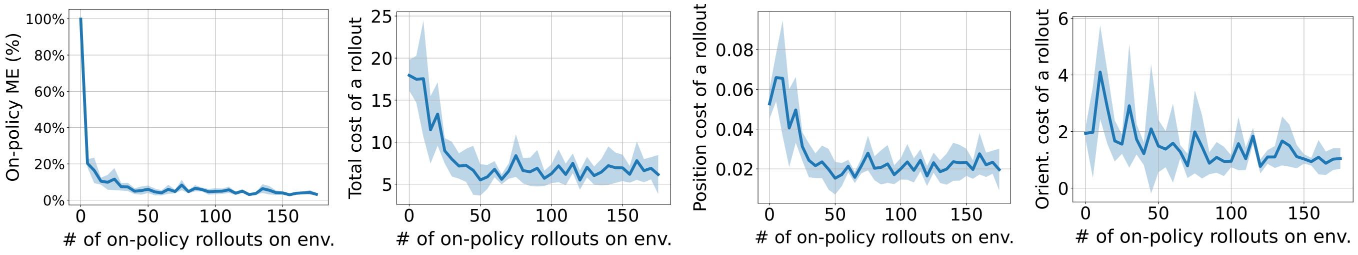
[1] ‘R’:‘red finger’, ‘B’:‘red finger’, ‘G’:‘green finger’, ‘CW’:‘clockwise’, ‘CCW’:‘counter-clockwise’.

time step $t = 9$ is $\texttt{sign}(\boldsymbol{\lambda}_9) = [0, 1, 1, 1, 0]^{\mathsf{T}}$. The bottom panel in Fig. 7a shows the cube's orientation angle trajectory, where the segments of different colors show different mode activation. The upper row in Fig. 7b shows the snapshots of the simulator at some key time steps of the same rollout, and the lower row gives the zoom-in details. Physically, the red fingertip begins touching the cube at time step 7, pushes the cube to rotate clockwise during steps 7-14 (during this period it also slips on the surface of the cube), and then separates from the cube at time step 15. The blue fingertip begins touching the cube at time step 9, then pushes the cube to rotation closewise, and finally separates from the cube at step 15. The green fingertip begins touching the cube at time step 11, and continue pushing the cube until the end of the rollout. By connecting Fig. 7a and 7b, we can observe the empirical correspondence between $\boldsymbol{g}()$'s mode activation in Fig. 7a and physical interaction in Fig.7b, listed in Table V.

In Fig. 7c and 7d, the target orientation is $\alpha^{\text{goal}} = 0.148$ (rad). Similar to the above description, Table VI gives the empirical correspondence between the mode activation in Fig. 7c and physical contact interaction in Fig. 7d. Table VI shows a more interesting connection between the mode activation in

TABLE VI: Empirical correspondence between LCS mode activation in Fig. 7c and physical contact interaction in Fig.7d.

Time t	**Mode activation in** $\boldsymbol{g}()$	**Interaction between fingertips and cube**[1]	**Interaction between cube and table**[1]
$t = 0, 1, 2$	$\texttt{sign}(\boldsymbol{\lambda}) = [0, 0, 0, 0, 0]^\mathsf{T}$	`R, G, B separate`	`Cube stick to table` or `CCW rotation slip`
$t = 3, 13, 18$	$\texttt{sign}(\boldsymbol{\lambda}) = [1, 0, 0, 0, 0]^\mathsf{T}$	(`R separate` and `G, B touching`) or `R, G, B touching`	`CCW rotational slip (large)`
$t = 4, 6, 8, 9, 11, 15, 16$	$\texttt{sign}(\boldsymbol{\lambda}) = [1, 1, 0, 0, 0]^\mathsf{T}$	`R, G, B touching`	`CCW rotational slip (small)`
$t = 5, 7, 10, 12, 14, 17, 19$	$\texttt{sign}(\boldsymbol{\lambda}) = [0, 1, 0, 0, 0]^\mathsf{T}$	`R, G, B touching`	`CW rotational slip`



(a) (b) (c) (d)

Fig. 8: Learning curves of the three-finger manipulation system for the Cube Moving task. Each curve is the average of five random seeds, and its shaded area shows the standard deviation. All results here are shown on an on-policy rollout basis, and each on-policy rollout is a result of running the reduced-order $\boldsymbol{g}$-MPC controller on the three-finger system (environment), i.e., the (unknown) full-order $\boldsymbol{f}()$. Detailed explanations are given in text.

$\boldsymbol{g}()$ and the physical contact. From the bottom panel of Fig. 7c, we see the three fingers turns the cube to the target at time step 5, and from that on, it slightly shakes the cube around the target. This makes the contact interaction between the cube and table also change alternatively, i.e., between the `CW rotational slip` and `CCW rotational slip`. This alternative physical interactions have been captured in the upper panel of Fig. 7c, where the mode activation from time step 5 also changes alternatively. This clearly shows the connections between mode in the reduced-order LCS $\boldsymbol{g}()$ and physical contacts. For example, the mode $\texttt{sign}(\boldsymbol{\lambda}_t) = [0, 1, 0, 0, 0]^\mathsf{T}$ is for the cube `CW rotational slip` (blue segments), while $\texttt{sign}(\boldsymbol{\lambda}_t) = [1, 1, 0, 0, 0]^\mathsf{T}$ and $\texttt{sign}(\boldsymbol{\lambda}_t) = [1, 0, 0, 0, 0]^\mathsf{T}$ are for `CCW rotational slip` (green and purple segments).

From Tables V and VI, we have the following comments.

(i) The learned LCS $\boldsymbol{g}()$ can approximately capture the hybrid nature of the physical system. Since we limit the maximum mode count in $\boldsymbol{g}()$ to 32, some physical contact interactions share the same mode activation of $\texttt{sign}(\boldsymbol{\lambda})$. For example, in Table V, the mode $\texttt{sign}(\boldsymbol{\lambda}) = [0, 0, 0, 1, 0]^\mathsf{T}$ captures two interactions between the cube and fingertips: (`R, G, B separate`) and (`G, B separate` and `R touching`).

(ii) Note that Tables V and VI come from empirical observation. For contact interactions that are very similar in human eyes, there could exist unnoticeable physical differences, leading to different modes in $\boldsymbol{g}()$. For example, in Table VI, `CCW rotational slip (large)` corresponds to $\texttt{sign}(\boldsymbol{\lambda}) = [1, 0, 0, 0, 0]^\mathsf{T}$, while `CCW rotational slip (small)` to $\texttt{sign}(\boldsymbol{\lambda}) = [1, 1, 0, 0, 0]^\mathsf{T}$. There is no tight connection from the mode of $\boldsymbol{g}()$ to physical phenomena.

D. Cube Moving Task

This session presents the results and analysis of using the proposed method to solve the Cube Moving task. In this task, we set weights in (40) as

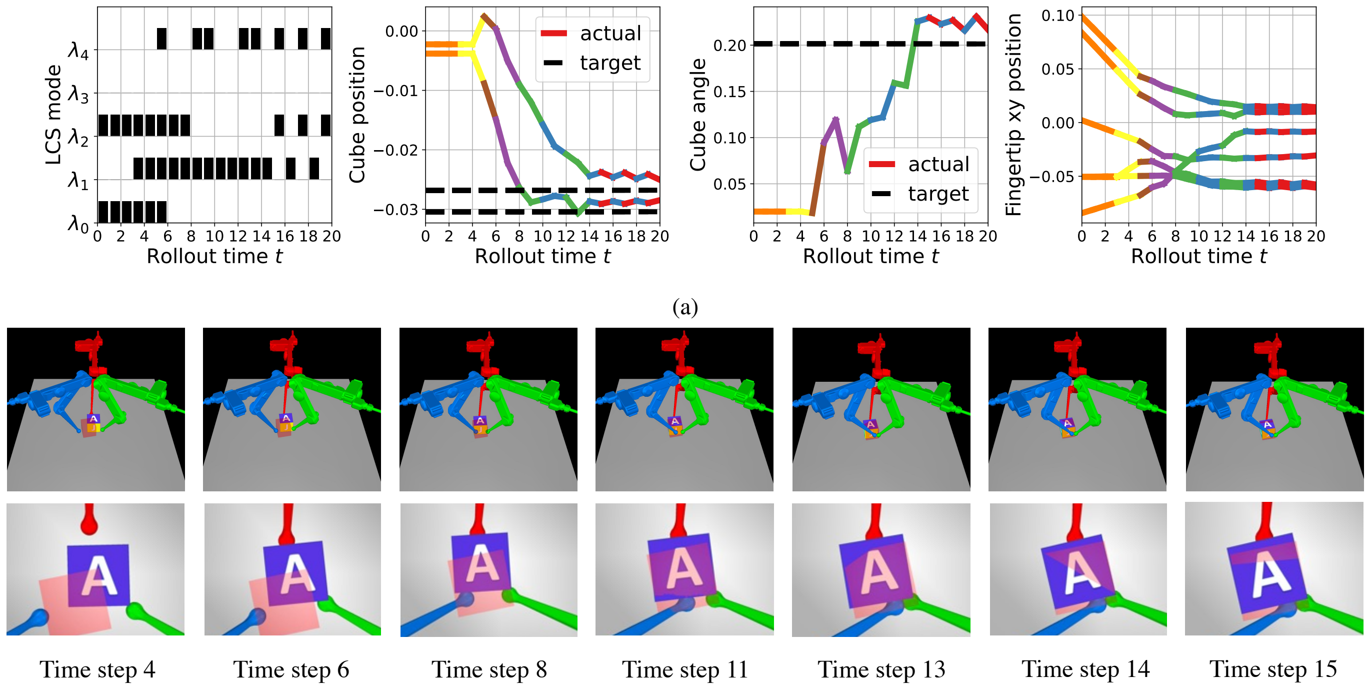
$$\begin{aligned}\boldsymbol{w}^c &= \begin{bmatrix}12.0 & 200.0 & 0.2\end{bmatrix}^\mathsf{T}, \\ \boldsymbol{w}^h &= \begin{bmatrix}6.0 & 200.0 & 1.0\end{bmatrix}^\mathsf{T}.\end{aligned} \tag{43}$$

The above weight values are not deliberately picked. In fact, the performance is not sensitive to the choice of $\boldsymbol{w}^c$ and $\boldsymbol{w}^h$. As will be discussed in Section VII-E, the similar learning and task performance permits a wide selection of weight values.

TABLE VII: Performance of learned $\boldsymbol{g}$-MPC for Cube Moving

Results	Value (mean+std)
Total number of hybrid modes in $\boldsymbol{g}$-MPC	15
Terminal position error $\lVert\boldsymbol{p}_{\text{cube},H} - \boldsymbol{p}^{\text{goal}}\rVert$	0.0076 ± 0.0010 (m)
Terminal position error (rel.) $\frac{\lVert\boldsymbol{p}_{\text{cube},H} - \boldsymbol{p}^{\text{goal}}\rVert^2}{\lVert\boldsymbol{p}^{\text{goal}}\rVert^2}$	3.8% ± 1.3%
Terminal orientation error $\lvert\alpha_{\text{cube},H} - \alpha^{\text{goal}}\rvert$	0.065 ± 0.021 (rad)
Total training time	4.6 ± 0.4 (mins)
Total # of environment samples in training	≤4k
Running frequency of $\boldsymbol{g}$-MPC controller	> 30 Hz
Max. $\frac{\mathrm{w}^{\text{mag}}_{\text{disturb}}}{m_{\text{cube}}}$ until $\lVert\boldsymbol{p}_{\text{cube},H} - \boldsymbol{p}^{\text{goal}}\rVert \geq 0.01$	0.9 m/s^2

1) Results: Similar to the previous session, we present the learning curves of the three-finger manipulation system in Fig. 8, and list the key results in Table VII. Each result is the average of five random seeds. Specifically, Fig. 8a shows the relative model error of the learned LCS $\boldsymbol{g}()$ evaluated on the $\boldsymbol{g}$-MPC policy data, defined in (31). Fig. 8b shows the total



(a)

Time step 4 Time step 6 Time step 8 Time step 11 Time step 13 Time step 14 Time step 15

(b)

Fig. 9: One rollout by running the learned reduced-order $\boldsymbol{g}$-MPC on the three-finger manipulation system for the Cube Moving task. (a) shows the mode activation in $\boldsymbol{g}()$ (first panel), the trajectory of the cube xy position (second panel), the trajectory of the cube angle (third panel), and the trajectories of all three fingertip xy positions (fourth panel), over the duration of the rollout. Here, in the model activation panel, black brick indicate $\lambda_i > 0$ and blank $\lambda_i = 0$. In the trajectory panels, the segments are colored differently, corresponding to different mode activation. (b) shows the snapshots of the environment at key time steps of the $\boldsymbol{g}$-MPC rollout. Here, the upper row of (b) shows the whole environment, and the lower row show the zoom-in details. Explanations and analysis are given in Section VII-D2 and Table VIII.

cost of a rollout with the $\boldsymbol{g}$-MPC controller, defined in (33). Fig. 8c shows the position cost of a rollout with the $\boldsymbol{g}$-MPC controller, defined as

$$E_{\boldsymbol{\beta}\sim p(\boldsymbol{\beta})}\,\mathbb{E}_{\boldsymbol{x}\sim p_{\boldsymbol{\beta}}(\boldsymbol{x}_0)}\sum\nolimits_{t=0}^{H}\|\boldsymbol{p}_{\text{cube},t}-\boldsymbol{p}^{\text{goal}}\|^2. \tag{44}$$

Fig. 8d shows the orientation cost (42) of a rollout with the $\boldsymbol{g}$-MPC controller. Table VII lists some key quantitative results. Here, the cube's terminal position error $\|\boldsymbol{p}_{\text{cube},H}-\boldsymbol{p}^{\text{goal}}\|$ and terminal orientation error $|\alpha_{\text{cube},H}-\alpha^{\text{goal}}|$ are calculated at the end (time step H) of a rollout. In the last row of Table VII, we test the robustness of the closed-loop $\boldsymbol{g}$-MPC controller against the external disturbance forces added to the cube during its motion. Here, we apply an external disturbance wrench (3D torque and 3D forces), sampled from $U[-\mathrm{w}^{\text{mag}}_{\text{disturb}}, \mathrm{w}^{\text{mag}}_{\text{disturb}}]$, during each time interval (0.1s) of the rollout steps. We increase the disturbance magnitude $\mathrm{w}^{\text{mag}}_{\text{disturb}}$ until the resulting $\boldsymbol{g}$-MPC rollout has an average cube terminal position error $\|\boldsymbol{p}_{\text{cube},H}-\boldsymbol{p}^{\text{goal}}\| \geq 0.01$ (m). We report the result using $\mathrm{w}^{\text{mag}}_{\text{disturb}}/m_{\text{cube}}$, with m_{cube} the cube mass. Based on the results in Fig. 8 and Table VII, we have the following conclusions.

(i) Without any prior knowledge, the proposed method learns a reduced-order LCS and successfully solves the Cube Moving manipulation task within 5 minutes of wall-clock time. The reduced-order model $\boldsymbol{g}()$ leads to a real-time $\boldsymbol{g}$-MPC controller with running frequency > 30 Hz. The method requires collecting less than 4k data points (175 rollouts × 20 steps/rollout) from the environment.

(ii) The learned reduced-order LCS $\boldsymbol{g}()$ only permits 32 modes (among them 15 are used for the task), which is much fewer than the estimated number of hybrid modes in full-order dynamics, which could be thousands.

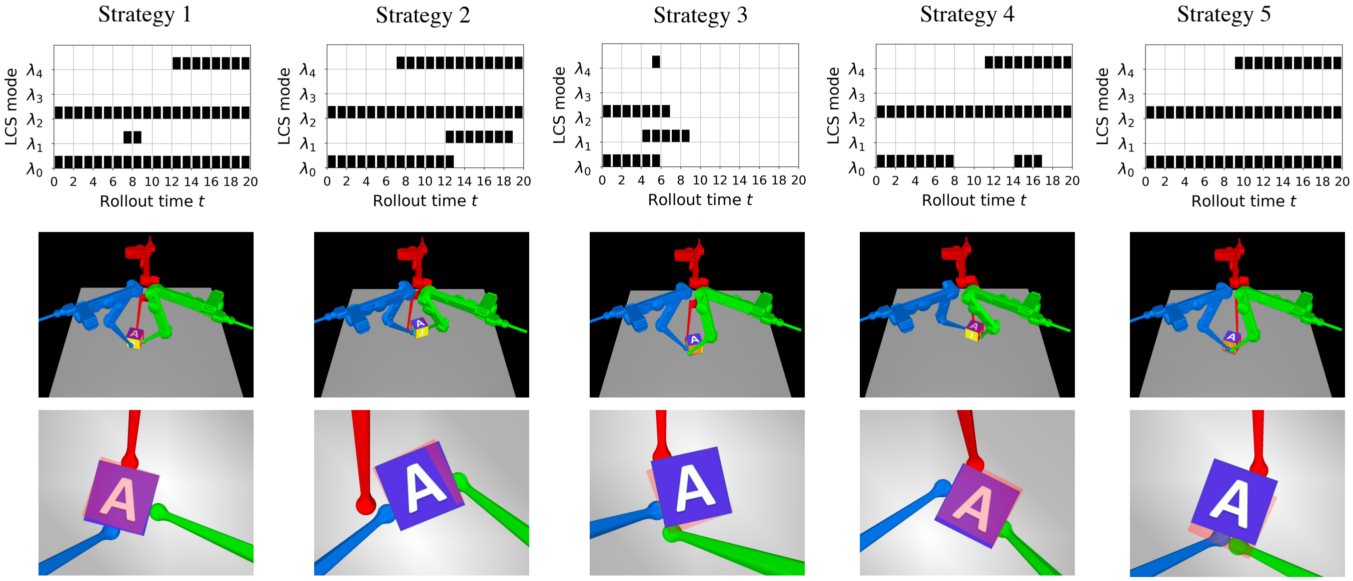
(iii) The reduced-order $\boldsymbol{g}$-MPC controller shows robustness against large external wrench disturbances. This is an advantage of using the closed-loop $\boldsymbol{g}$-MPC controller.

2) Analysis of Reduced-Order Hybrid Modes: Fig. 9 shows one rollout of running the learned $\boldsymbol{g}$-MPC on the three-finger manipulation system (the environment). Fig. 9a plots the trajectory of mode activation in $\boldsymbol{g}()$ (first panel), the trajectory of the cube position (second panel), the trajectory of the cube orientation angle (third panel), and the trajectories of xy position of three fingertips (fourth panel), over the duration of rollout. All trajectories are colored differently for different mode activation in $\boldsymbol{g}()$. Fig. 9b shows the snapshots of the environment at some key time steps of the rollout. As done in the previous task, Table VIII lists the empirical connection between mode activation in $\boldsymbol{g}$-MPC in Fig. 9a and physical contact interaction in Fig. 9b.

Results in Fig. 9 and Table VIII show rich contact interactions in the Cube Moving task. Different hybrid mode of $\boldsymbol{g}()$ can approximately capture different contact interactions during rollout. Those interactions include `separate`, `stick`, and `slip` between fingertips and cube, `stick`, `CCW rotational slip`, and `CW rotational slip` between cube and table, and `B contacting G` between

TABLE VIII: Approximate correspondence between LCS mode activation in Fig. 9a and contact interaction in Fig.9b.

Duration t	Mode activation in $\boldsymbol{g}()$	Interaction between fingertips and cube	Interaction between cube and table
$t = 0, 1, 2$	$\texttt{sign}(\boldsymbol{\lambda}) = [1, 0, 1, 0, 0]^\mathsf{T}$	R, G, B separate	stick to table
$t = 3, 4$	$\texttt{sign}(\boldsymbol{\lambda}) = [1, 1, 1, 0, 0]^\mathsf{T}$	R, B separate and G separate	translational slip
$t = 5, 6, 7$	$\texttt{sign}(\boldsymbol{\lambda}) = [1, 1, 1, 0, 1]^\mathsf{T}$ or $[0, 1, 1, 0, 0]^\mathsf{T}$	R, G stick & B separate	translational & rotational slip
$t = 8, 9$	$\texttt{sign}(\boldsymbol{\lambda}) = [0, 1, 0, 0, 1]^\mathsf{T}$	R, G, B stick	translational & rotational slip
$t = 10, 11$	$\texttt{sign}(\boldsymbol{\lambda}) = [0, 1, 0, 0, 0]^\mathsf{T}$	R, G stick and B right slip	translational & rotational slip
$t = 12, 13$	$\texttt{sign}(\boldsymbol{\lambda}) = [0, 1, 0, 0, 1]^\mathsf{T}$	R stick and B contacting G	translational & rotational slip
$t = 14, 16, 18$	$\texttt{sign}(\boldsymbol{\lambda}) = [0, 1, 0, 0, 0]^\mathsf{T}$	R, B, G stick and B contacting G	CCW rotational slip
$t = 15, 17, 19$	$\texttt{sign}(\boldsymbol{\lambda}) = [0, 0, 1, 0, 1]^\mathsf{T}$	R, B, G stick and B contacting G	CW rotational slip



R **slip** on the **back** face
G **stick** on the **right** face
B **slip** on the **front** face

R **separate** from **back** face
G **stick** on the **right** face
B **stick** on the **left** face

R **stick** on **back** face
G **slip** on the **front** face
B **slip** on the **left** face

R **stick** on **back** face
G **slip** on the **right** face
B **slip** on the **left** face

R **stick** on **back** face
G **stick** on the **right** face
B **slip** on the **front** face

Fig. 10: Different contact strategies generated by the same learned reduced-order LCS $\boldsymbol{g}()$ in its $\boldsymbol{g}$-MPC policy rollout given different targets. The first row shows the mode activation of $\boldsymbol{g}$ over the rollout; the second row shows the snapshot of the environment at the end of the rollout; and the third row shows the zoom-in details of contact interactions. The bottom title in each column describes the main interactions for that strategy. Analysis is given in Section VII-D3.

fingertips, happening at different time steps. Notably, in Fig. 9a, for $t \geq 14$, the three fingers start slightly shaking the cube, as shown by the cube position and angle trajectories, and active mode in $\boldsymbol{g}()$ also changes alternatively, as shown in the mode activation panel. Those observations indicate the modes of the learned reduced-order LCS are able to approximately capture the rich contact interactions in the manipulation system.

3) Generation of Different Manipulation Strategies: Notably, for different target poses (sampled from (35)), we observe that the learned $\boldsymbol{g}$-MPC produces different manipulation strategies to move the cube. We show this in Fig. 10.

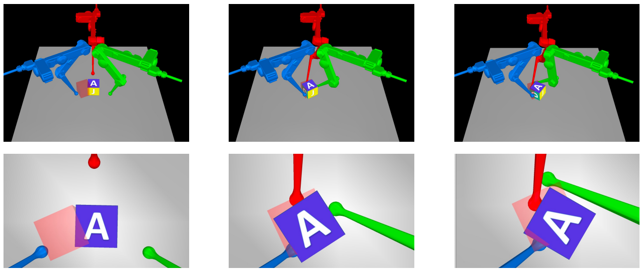
Different columns in Fig. 10 shows different manipulation strategies given different targets. Note that all those strategies are generated from the same learned reduced-order LCS $\boldsymbol{g}()$ in its $\boldsymbol{g}$-MPC policy rollout, given different targets. The first row in Fig. 10 shows the mode activation in $\boldsymbol{g}()$ during the rollout with $\boldsymbol{g}$-MPC controller; the second row shows the snapshot of the environment at the end of rollout; and the third row shows the zoom-in details of the contact interaction. The bottom title in each column describes the main physical interactions for the rollout. From Fig. 10, we have the following comments.

(i) Fig. 10 clearly shows the learned reduced-order LCS $\boldsymbol{g}()$ enables generating different strategies for different targets. Particularly, $\boldsymbol{g}()$ enables the three fingertips to choose different faces (`right`, `left`, `front`, and `back`) of the cube with different contact interactions (`separate`, `stick`, and `slip`). For example, the blue fingertip chooses the `front` face in Strategies 1 and 5 while the `left` face in the others. The red fingertip is `slip` in Strategy 1, `separate` in Strategy 2, and `stick` in others.

(ii) Jointly looking at the mode activation of $\boldsymbol{g}()$ in the first row of Fig. 10, we observe that the mode activation at the beginning of all rollouts, e.g., $t \leq 4$, are quite similar since the cube during this period is still, and all fingertips are separate from the cube. After $t > 4$, different modes in $\boldsymbol{g}$-MPC begin to activate, leading to different manipulation strategies.

(iii) Recall that all strategies shown in Fig. 10 are generated by the same learned reduced-order LCS $\boldsymbol{g}()$. The results clearly show the effectiveness of the learned reduced-order $\boldsymbol{g}$-MPC to capture and make use of its task-relevant hybrid modes to produce different strategies for the manipulation task.

4) Occasional Reorientation Failure: We report some failure cases in the above cube moving manipulation task. Fig. 11 shows one example of reorientation failure. Here, some snapshots at key time steps of the $\boldsymbol{g}$-MPC rollout are shown. At time step 10 (middle column), the three fingertips had successfully moved and turned the cube to the target pose. However, at the subsequent time steps, the green fingertip continues to slide along the cube surface, leading to the misalignment of the cube orientation (third column).



Time step 0 Time step 11 Time step 19

Fig. 11: An example of reorientation failure in a rollout.

We observed that reorientation failure are target dependent, meaning that reorientation failures happen more frequently at some targets than at others. Also, changing the random seed for target distribution (35) also changes the failure target locations. This makes us believe that the reorientation failure could be caused by insufficient target sampling at such regions. In fact, the whole training process sampled from fewer than 200 target poses from (35). Some regions of the target space could be less sampled than other regions, leading to the learned $\boldsymbol{g}()$ not well representing those regions. We expect that those failures could be reduced by decreasing the target space. In fact, in our previous Cube Turning task, since the target space is one-dimensional, we have not experienced the reorientation failures in that task.

E. Discussion

We conclude this section with some additional performance evaluations of the proposed method, and note some limitations (and future work) of the proposed method.

1) LCS with Different Hybrid Modes: In the above three-finger manipulation tasks, we have used LCS models $\boldsymbol{g}()$ in (8) with a fixed $\dim \boldsymbol{\lambda} = 5$, which allows the representation of 32 modes. Results in Table IV and Table VII show that the learned reduced-order LCS has not used up all of those modes. Therefore, a natural question is whether it is possible to learn a LCS with fewer hybrid modes. To show this, we learn a reduced-order LCS $\boldsymbol{g}()$ with different $\dim \boldsymbol{\lambda}$ for the Cube Turning task, under the same other settings as in Section VII-C. We show the results in Fig. 12. Here, for each $\dim \boldsymbol{\lambda}$ case, we run the experiments with five random seeds, and the mean and variance are computed across different runs.

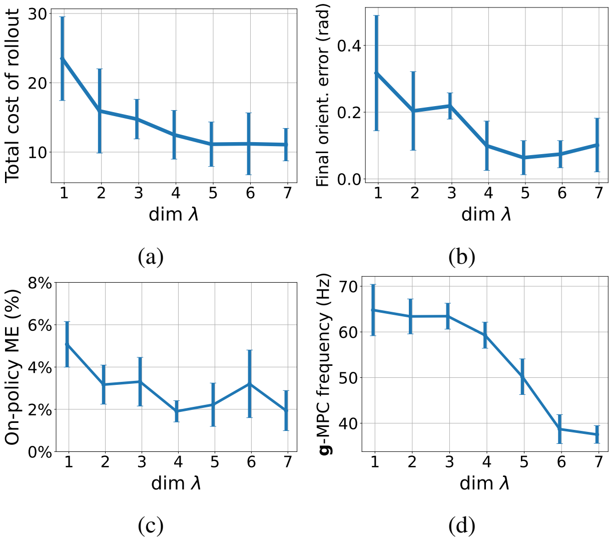
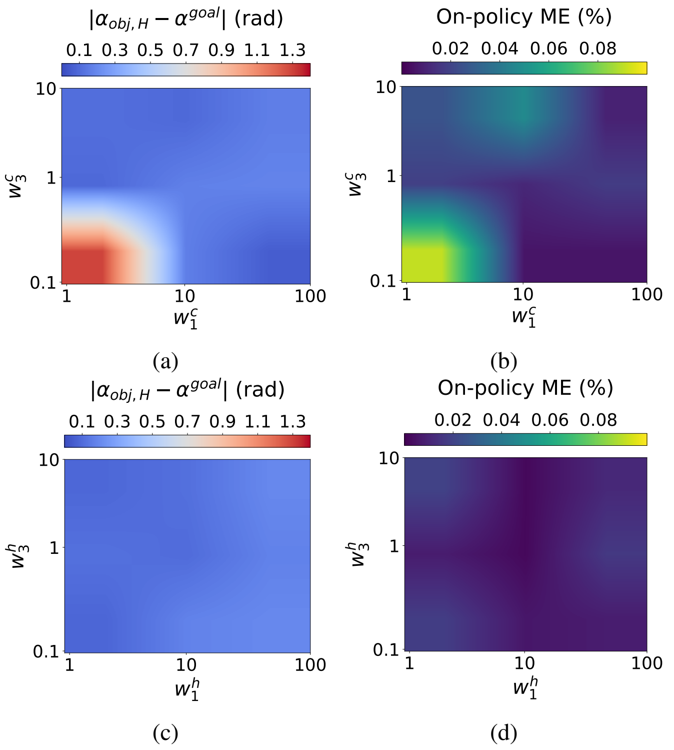


Fig. 12: Testing performance of the learned LCS $\boldsymbol{g}()$ with different $\dim \boldsymbol{\lambda}$ for the Cube Turning task. The metrics in (a)-(c) follow the same definitions as the ones in Section VII-C, and (d) is running frequency of $\boldsymbol{g}$-MPC. For each $\dim \boldsymbol{\lambda}$ case, the mean and variance are calculated across five runs with different random seeds. The other settings follow the ones in Section VII-C.

Fig. 12 shows that a LCS of fewer modes, e.g., $\dim \boldsymbol{\lambda} \leq 3$, leads to degraded performance. The previous Table IV has showed that successful manipulation needs around 14 hybrid modes in $\boldsymbol{g}()$. Thus, the successful manipulation on average requires at least $\dim \boldsymbol{\lambda} \geq 4$. Fig. 12 confirms this by showing improved performances if $\dim \boldsymbol{\lambda} \geq 4$. Fig. 12 also shows that further increasing of $\dim \boldsymbol{\lambda}$, say $\dim \boldsymbol{\lambda} = 6$, will not help the performance too much. Also, increasing $\dim \boldsymbol{\lambda}$ will slow the speed of the closed-loop $\boldsymbol{g}$-MPC controller in Fig. 12d. In practice, the choice of $\dim \boldsymbol{\lambda}$ depends on tasks and systems, and one typically needs to find a $\dim \boldsymbol{\lambda}$ (via trial and error) by balancing its performance and computational complexity.

2) Choice of Cost Weights: In this session, we investigate how the choice of cost weights $\boldsymbol{w}^c = [w_1^c, w_2^c, w_3^c]^\mathsf{T}$ and $\boldsymbol{w}^h = [w_1^h, w_2^h, w_3^h]^\mathsf{T}$ in (40) affects the algorithm performance. We apply the proposed method to learn a reduced LCS of $\dim \boldsymbol{\lambda} = 5$ for the Cube Turning task, with the same other conditions as in Section VII-C, except varying the values of (w_1^c, w_3^c) and (w_1^h, w_3^h) (note $w_2^c = w_2^h = 0$ in the Cube Turning task). The results are shown in Fig. 13.

Compared to the previous performance (see Fig. 6 and Table IV) in Section VII.C, Fig. 13a and Fig. 13b show that a similarly good performance permits a wide selection of running cost weights $\boldsymbol{w}^c$, e.g., $w_3^c = 0.5$ and $w_1^c \in (10, 100)$. Typically, to achieve a good algorithm performance, Fig. 13a and Fig. 13b suggest that the selection of running cost weights $\boldsymbol{w}^c$ should make the corresponding cost terms, here, w_1^c for the distance between fingertip and cube and w_3^c for the orientation cost, roughly have a similar scale. The results in Fig. 13c and Fig. 13d also say that the allowable choice of final cost weights



(a)

(b)

(c)

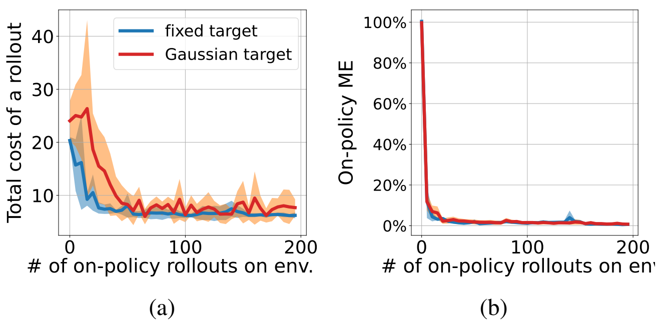
(d)

Fig. 13: Performance of the proposed method with different choices of cost weights $\boldsymbol{w}^c$ and $\boldsymbol{w}^h$ in (40) for the Cube Turning task. (a) and (b) report the performance using different $\boldsymbol{w}^c$ while fixing $\boldsymbol{w}^h = [2, 0, 10]^{\mathsf{T}}$, and (c) and (d) using different $\boldsymbol{w}^h$ while fixing $\boldsymbol{w}^c = [10, 0, 2]^{\mathsf{T}}$. The metric in (a) and (c) is the cube's terminal orientation error $|\alpha_{\text{cube},H} - \alpha^{\text{goal}}|$, as used in Table IV, and the metric in (b) and (d) uses the on-policy model error, defined in (31).

$\boldsymbol{w}^h$ is more flexible, and different $\boldsymbol{w}^h$s have limited influence on the algorithm performance. Thus, we can conclude that the proposed algorithm is not sensitive to the choice of $\boldsymbol{w}^c$ and $\boldsymbol{w}^h$ in (40). In practical implementation, it is always not difficult to find the cost weights to produce good performance, and one empirical principle to select cost weights is to make each cost term have a similar scale.

3) Choice of Task Distribution: We briefly discuss the choice of task parameter $\boldsymbol{\beta}$, which we have used to define a distribution of tasks $p(\boldsymbol{\beta})$. In the examples above, we have considered uniformly distributed targets (e.g., $p(\boldsymbol{\beta})$ in (34) and (35)). We note, however, that this choice is somewhat arbitrary, and for completeness we demonstrate similar performance across other distributions. The experiment settings follow Section VII-C, except using different targets distribution $p(\boldsymbol{\beta})$.

Fig. 14, shows results from two other target distributions for the Cube Turning task: δ-distribution corresponding to a single fixed target, $p(\boldsymbol{\beta} = 0.6) = 1$, shown in the blue lines, and a Gaussian distributed target, $p(\boldsymbol{\beta}) = \mathcal{N}(0.6, 0.2)$, shown in the red lines. The learning curves in Fig.14a and Fig. 14b indicate that for each task distribution, a reduced-order LCS model is successfully learned. Also, we notice that compared to the single target task, multiple targets bring more variance. In conclusion, given a different task distribution, the proposed method always finds a reduced order model to minimize the task performance gap in expectation over the task distribution, as in (13). In practice, we expect $p(\boldsymbol{\beta})$ to be chosen either by the practitioner, to best represent the tasks the robot must accomplish, or to be independently identified (e.g., via the output from some higher-level planner).



(a)

(b)

Fig. 14: Performance of the proposed method under different target distribution in the Cube Turning task. We use two target distributions: δ-distribution for a single fixed target, i.e., $p(\boldsymbol{\beta} = 0.6) = 1$, in the blue lines, and a Gaussian distributed target, $p(\boldsymbol{\beta}) = \mathcal{N}(0.6, 0.2)$, in the red lines. The metric follows the ones used in Fig. 6.

4) Limitation of PWA Models: In the paper, we use PWA models (8) as the reduced-order hybrid representation. As these PWA models are inherently based on linearization, they do not naturally apply to all manipulation tasks, for example large rotations (e.g. full 360-degree rotation of the cube). Such large rotations involve significant non-linearity that local PWA models cannot capture well, unless one adds more 'pieces' in PWA to approximate it [48]. However, using more 'pieces' to approximate smooth non-linearity is not the interest of this paper; instead, we focus on using pieces to capture the hybrid structure (i.e., mode boundaries) of a non-linear hybrid system. But this limitation motivates a future direction to extend LCS representation for non-linear complementarity models.

VIII. Conclusions And Future Work

This paper proposes the method of task-driven hybrid model reduction for multi-contact dexterous manipulation. Building upon our prior work of learning linear complementarity systems, we propose learning a reduced-order hybrid model with a limited number of task-relevant hybrid modes, such that it enables real-time closed-loop MPC control and is sufficient to achieve high performance on hybrid systems like multi-finger dexterous manipulation. We have shown that learning a reduced-order hybrid model attains a provably upper-bounded closed-loop performance. We have demonstrated the proposed method in reducing the mode count of synthetic hybrid control systems by multiple orders of magnitude while achieving task performance loss of less than 5%. We apply the proposed method to solve three-finger robotic hand manipulation for object reorientation. Without any prior knowledge, the proposed method achieves state-of-the-art closed-loop performance in less than five minutes of data collection and model learning. The future work includes building the hardware and testing it on the hardware robotic manipulation system, as well as extension into nonlinear reduced-order hybrid models.

Acknowledgements

Toyota Research Institute funded and supported this work.

Appendix A
Proof of Lemma 1

Proof.

$$
\begin{aligned}
& J_{\boldsymbol{\beta}}\big(\mathbf{u}^{\boldsymbol{g}},\mathbf{F}(\mathbf{u}^{\boldsymbol{g}},\boldsymbol{x}_0)\big)-J_{\boldsymbol{\beta}}\big(\mathbf{u}^{\boldsymbol{f}},\mathbf{F}(\mathbf{u}^{\boldsymbol{f}},\boldsymbol{x}_0)\big)\\
=\ & \underbrace{J_{\boldsymbol{\beta}}\big(\mathbf{u}^{\boldsymbol{g}},\mathbf{F}(\mathbf{u}^{\boldsymbol{g}},\boldsymbol{x}_0)\big)-J_{\boldsymbol{\beta}}\big(\mathbf{u}^{\boldsymbol{g}},\mathbf{G}(\mathbf{u}^{\boldsymbol{g}},\boldsymbol{x}_0)\big)}_{\text{Term I}}\\
+\ & \underbrace{J_{\boldsymbol{\beta}}\big(\mathbf{u}^{\boldsymbol{g}},\mathbf{G}(\mathbf{u}^{\boldsymbol{g}},\boldsymbol{x}_0)\big)-J_{\boldsymbol{\beta}}\big(\mathbf{u}^{\boldsymbol{f}},\mathbf{G}(\mathbf{u}^{\boldsymbol{f}},\boldsymbol{x}_0)\big)}_{\text{Term II}}\\
+\ & \underbrace{J_{\boldsymbol{\beta}}\big(\mathbf{u}^{\boldsymbol{f}},\mathbf{G}(\mathbf{u}^{\boldsymbol{f}},\boldsymbol{x}_0)\big)-J_{\boldsymbol{\beta}}\big(\mathbf{u}^{\boldsymbol{f}},\mathbf{F}(\mathbf{u}^{\boldsymbol{f}},\boldsymbol{x}_0)\big)}_{\text{Term III}},
\end{aligned}
\tag{45}
$$

Here, Term I and Term III can follow the Lipschitz continuity:

$$
\begin{aligned}
\text{Term I} &\le M\|\mathbf{G}(\mathbf{u}^{\boldsymbol{g}},\boldsymbol{x}_0)-\mathbf{F}(\mathbf{u}^{\boldsymbol{g}},\boldsymbol{x}_0)\|\\
\text{Term III} &\le M\|\mathbf{G}(\mathbf{u}^{\boldsymbol{f}},\boldsymbol{x}_0)-\mathbf{F}(\mathbf{u}^{\boldsymbol{f}},\boldsymbol{x}_0)\|.
\end{aligned}
\tag{46}
$$

Term II trivially satisfies

$$
J_{\boldsymbol{\beta}}\big(\mathbf{u}^{\boldsymbol{g}},\mathbf{G}(\mathbf{u}^{\boldsymbol{g}},\boldsymbol{x}_0)\big)-J_{\boldsymbol{\beta}}\big(\mathbf{u}^{\boldsymbol{f}},\mathbf{G}(\mathbf{u}^{\boldsymbol{f}},\boldsymbol{x}_0)\big)\le 0 \tag{47}
$$

because $\mathbf{u}^{\boldsymbol{g}}$ is the minimum by the definition of $\boldsymbol{g}$-MPC (11). Putting (46) and (47) together, one has

$$
\begin{aligned}
& J_{\boldsymbol{\beta}}\big(\mathbf{u}^{\boldsymbol{g}},\mathbf{F}(\mathbf{u}^{\boldsymbol{g}},\boldsymbol{x}_0)\big)-J_{\boldsymbol{\beta}}\big(\mathbf{u}^{\boldsymbol{f}},\mathbf{F}(\mathbf{u}^{\boldsymbol{f}},\boldsymbol{x}_0)\big)\le\\
& M\Big(\|\mathbf{G}(\mathbf{u}^{\boldsymbol{g}},\boldsymbol{x}_0)-\mathbf{F}(\mathbf{u}^{\boldsymbol{g}},\boldsymbol{x}_0)\|+\|\mathbf{G}(\mathbf{u}^{\boldsymbol{f}},\boldsymbol{x}_0)-\mathbf{F}(\mathbf{u}^{\boldsymbol{f}},\boldsymbol{x}_0)\|\Big).
\end{aligned}
$$

The above still holds with expectation of both sides over $p(\boldsymbol{x}_0)$ and $p(\boldsymbol{\beta})$, yielding (15). This completes the proof. □

Remark. $\mathbb{E}_{p(\boldsymbol{\beta})}\,\mathbb{E}_{p_{\boldsymbol{\beta}}(\boldsymbol{x}_0)}\Big\|\mathbf{G}(\mathbf{u}^{\boldsymbol{f}},\boldsymbol{x}_0)-\mathbf{F}(\mathbf{u}^{\boldsymbol{f}},\boldsymbol{x}_0)\Big\|$ *in (15) is called the domain adaption in reinforcement learning [49]. Specifically, if* $\boldsymbol{g}()$ *is trained only with* $\boldsymbol{g}$*-MPC data* $\mathcal{D}^{\boldsymbol{g}}$*, the* domain adaption *term captures the model error when the* $\boldsymbol{g}()$ *is evaluated with* $\boldsymbol{f}$*-MPC data* $\mathcal{D}^{\boldsymbol{f}}$*. This domain adaption term is inevitable if want a learned proxy model* $\boldsymbol{g}()$ *trained on its generated data to capture the true dynamics* $\boldsymbol{f}()$.

Appendix B
Proof of Lemma 2

Proof. Given any $\boldsymbol{x}_0\sim p(\boldsymbol{x}_0)$ and $\boldsymbol{\beta}\sim p(\boldsymbol{\beta})$, as $\mathbf{u}^{\boldsymbol{g}}(\boldsymbol{x}_0,\boldsymbol{\beta})$ is a solution to (11), it satisfies the first-order condition

$$
\nabla_{\mathbf{u}}J_{\boldsymbol{\beta}}\big(\mathbf{u}^{\boldsymbol{g}},\mathbf{G}(\mathbf{u}^{\boldsymbol{g}},\boldsymbol{x}_0)\big)=\left(\frac{\partial J_{\boldsymbol{\beta}}}{\partial\mathbf{u}^{\boldsymbol{g}}}+\frac{\partial J_{\boldsymbol{\beta}}}{\partial\mathbf{G}(\mathbf{u}^{\boldsymbol{g}})}\frac{\partial\mathbf{G}}{\partial\mathbf{u}^{\boldsymbol{g}}}\right)^{\mathsf{T}}=\mathbf{0}, \tag{48}
$$

where $\frac{\partial J_{\boldsymbol{\beta}}}{\partial\mathbf{u}^{\boldsymbol{g}}}$ denotes the partial gradient of $J_{\boldsymbol{\beta}}\big(\mathbf{u},\mathbf{G}(\mathbf{u},\boldsymbol{x}_0)\big)$ with respect to $\mathbf{u}$ evaluated at $\mathbf{u}^{\boldsymbol{g}}(\boldsymbol{x}_0,\boldsymbol{\beta})$, and similar notations applies to $\frac{\partial J_{\boldsymbol{\beta}}}{\partial\mathbf{G}(\mathbf{u}^{\boldsymbol{g}})}$ and $\frac{\partial\mathbf{G}}{\partial\mathbf{u}^{\boldsymbol{g}}}$ and below. Using (48), one has

$$
\begin{aligned}
&\Big\|\nabla_{\mathbf{u}}J_{\boldsymbol{\beta}}\big(\mathbf{u}^{\boldsymbol{g}},\mathbf{F}(\mathbf{u}^{\boldsymbol{g}},\boldsymbol{x}_0)\big)\Big\|\\
=&\Big\|\nabla_{\mathbf{u}}J_{\boldsymbol{\beta}}\big(\mathbf{u}^{\boldsymbol{g}},\mathbf{F}(\mathbf{u}^{\boldsymbol{g}},\boldsymbol{x}_0)\big)-\nabla_{\mathbf{u}}J_{\boldsymbol{\beta}}\big(\mathbf{u}^{\boldsymbol{g}},\mathbf{G}(\mathbf{u}^{\boldsymbol{g}},\boldsymbol{x}_0)\big)\Big\|\\
=&\left\|\frac{\partial J_{\boldsymbol{\beta}}(\mathbf{u},\mathbf{F})}{\partial\mathbf{u}^{\boldsymbol{g}}}+\frac{\partial J_{\boldsymbol{\beta}}}{\partial\mathbf{F}(\mathbf{u}^{\boldsymbol{g}})}\frac{\partial\mathbf{F}}{\partial\mathbf{u}^{\boldsymbol{g}}}-\frac{\partial J_{\boldsymbol{\beta}}(\mathbf{u},\mathbf{G})}{\partial\mathbf{u}^{\boldsymbol{g}}}-\frac{\partial J_{\boldsymbol{\beta}}}{\partial\mathbf{G}(\mathbf{u}^{\boldsymbol{g}})}\frac{\partial\mathbf{G}}{\partial\mathbf{u}^{\boldsymbol{g}}}\right\|\\
\le&\left\|\frac{\partial J_{\boldsymbol{\beta}}(\mathbf{u},\mathbf{F})}{\partial\mathbf{u}^{\boldsymbol{g}}}-\frac{\partial J_{\boldsymbol{\beta}}(\mathbf{u},\mathbf{G})}{\partial\mathbf{u}^{\boldsymbol{g}}}\right\|+\left\|\frac{\partial J_{\boldsymbol{\beta}}}{\partial\mathbf{F}(\mathbf{u}^{\boldsymbol{g}})}\frac{\partial\mathbf{F}}{\partial\mathbf{u}^{\boldsymbol{g}}}-\frac{\partial J_{\boldsymbol{\beta}}}{\partial\mathbf{G}(\mathbf{u}^{\boldsymbol{g}})}\frac{\partial\mathbf{G}}{\partial\mathbf{u}^{\boldsymbol{g}}}\right\|,
\end{aligned}
\tag{49}
$$

where last inequality is due to Cauchy–Schwarz inequality, and for clarity, we drop the dependence on $\boldsymbol{x}_0$ momentarily. The first term in (49) has

$$
\left\|\frac{\partial J_{\boldsymbol{\beta}}(\mathbf{u},\mathbf{F})}{\partial\mathbf{u}^{\boldsymbol{g}}}-\frac{\partial J_{\boldsymbol{\beta}}(\mathbf{u},\mathbf{G})}{\partial\mathbf{u}^{\boldsymbol{g}}}\right\|\le L_2\Big\|\mathbf{F}(\mathbf{u}^{\boldsymbol{g}})-\mathbf{G}(\mathbf{u}^{\boldsymbol{g}})\Big\|. \tag{50}
$$

because of the L_2-Lipschitz continuity of $\nabla_{\mathbf{u}}J_{\boldsymbol{\beta}}(\mathbf{u},\mathbf{x})$ assumed in Lemma 2. The second term of (49) has

$$
\begin{aligned}
&\left\|\frac{\partial J_{\boldsymbol{\beta}}}{\partial\mathbf{F}(\mathbf{u}^{\boldsymbol{g}})}\frac{\partial\mathbf{F}}{\partial\mathbf{u}^{\boldsymbol{g}}}-\frac{\partial J_{\boldsymbol{\beta}}}{\partial\mathbf{G}(\mathbf{u}^{\boldsymbol{g}})}\frac{\partial\mathbf{G}}{\partial\mathbf{u}^{\boldsymbol{g}}}\right\|\\
=&\left\|\frac{\partial J_{\boldsymbol{\beta}}}{\partial\mathbf{F}(\mathbf{u}^{\boldsymbol{g}})}\frac{\partial\mathbf{F}}{\partial\mathbf{u}^{\boldsymbol{g}}}-\frac{\partial J_{\boldsymbol{\beta}}}{\partial\mathbf{F}(\mathbf{u}^{\boldsymbol{g}})}\frac{\partial\mathbf{G}}{\partial\mathbf{u}^{\boldsymbol{g}}}+\frac{\partial J_{\boldsymbol{\beta}}}{\partial\mathbf{F}(\mathbf{u}^{\boldsymbol{g}})}\frac{\partial\mathbf{G}}{\partial\mathbf{u}^{\boldsymbol{g}}}-\frac{\partial J_{\boldsymbol{\beta}}}{\partial\mathbf{G}(\mathbf{u}^{\boldsymbol{g}})}\frac{\partial\mathbf{G}}{\partial\mathbf{u}^{\boldsymbol{g}}}\right\|\\
\le&\left\|\frac{\partial J_{\boldsymbol{\beta}}}{\partial\mathbf{F}(\mathbf{u}^{\boldsymbol{g}})}\frac{\partial\mathbf{F}}{\partial\mathbf{u}^{\boldsymbol{g}}}-\frac{\partial J_{\boldsymbol{\beta}}}{\partial\mathbf{F}(\mathbf{u}^{\boldsymbol{g}})}\frac{\partial\mathbf{G}}{\partial\mathbf{u}^{\boldsymbol{g}}}\right\|+\left\|\frac{\partial J_{\boldsymbol{\beta}}}{\partial\mathbf{F}(\mathbf{u}^{\boldsymbol{g}})}\frac{\partial\mathbf{G}}{\partial\mathbf{u}^{\boldsymbol{g}}}-\frac{\partial J_{\boldsymbol{\beta}}}{\partial\mathbf{G}(\mathbf{u}^{\boldsymbol{g}})}\frac{\partial\mathbf{G}}{\partial\mathbf{u}^{\boldsymbol{g}}}\right\|\\
\le&M_1\left\|\frac{\partial\mathbf{F}}{\partial\mathbf{u}^{\boldsymbol{g}}}-\frac{\partial\mathbf{G}}{\partial\mathbf{u}^{\boldsymbol{g}}}\right\|+M_gL_1\Big\|\mathbf{F}(\mathbf{u}^{\boldsymbol{g}})-\mathbf{G}(\mathbf{u}^{\boldsymbol{g}})\Big\|,
\end{aligned}
\tag{51}
$$

where the last inequality is due to the bound $\|\nabla_{\mathbf{x}}J_{\boldsymbol{\beta}}(\mathbf{u},\mathbf{x})\|\le M_1$, L_1-Lipschitz continuity of $\nabla_{\mathbf{x}}J_{\boldsymbol{\beta}}(\mathbf{u},\mathbf{x})$, and the bound $\|\nabla_{\mathbf{u}}G(\mathbf{u})\|\le M_g$ given in Lemma 2.

Combining (49)-(51) and replacing $\Big\|\frac{\partial\mathbf{F}}{\partial\mathbf{u}^{\boldsymbol{g}}}-\frac{\partial\mathbf{G}}{\partial\mathbf{u}^{\boldsymbol{g}}}\Big\|$ compactly with $\big\|\nabla_{\mathbf{u}}\mathbf{F}(\mathbf{u})-\nabla_{\mathbf{u}}\mathbf{G}(\mathbf{u})\big\|$ we have (18) and (19) in Lemma 2. This completes the proof. □

References

[1] M. Parmar, M. Halm, and M. Posa, “Fundamental challenges in deep learning for stiff contact dynamics,” in *IEEE/RSJ International Conference on Intelligent Robots and Systems*. IEEE, 2021, pp. 5181–5188.

[2] B. Bianchini, M. Halm, N. Matni, and M. Posa, “Generalization bounded implicit learning of nearly discontinuous functions,” in *Learning for Dynamics and Control Conference*. PMLR, 2022, pp. 1112–1124.

[3] S. Pfrommer, M. Halm, and M. Posa, “Contactnets: Learning discontinuous contact dynamics with smooth, implicit representations,” in *Conference on Robot Learning*, 2020.

[4] W. Jin, T. D. Murphey, D. Kulić, N. Ezer, and S. Mou, “Learning from sparse demonstrations,” *IEEE Transactions on Robotics*, 2022.

[5] Y.-M. Chen and M. Posa, “Optimal reduced-order modeling of bipedal locomotion,” in *IEEE International Conference on Robotics and Automation*. IEEE, 2020, pp. 8753–8760.

[6] A. Aydinoglu and M. Posa, “Real-time multi-contact model predictive control via admm,” in *IEEE International Conference on Robotics and Automation*, 2022, pp. 3414–3421.

[7] S. L. Cleac’h, T. Howell, M. Schwager, and Z. Manchester, “Fast contact-implicit model-predictive control,” *arXiv preprint arXiv:2107.05616*, 2021.

[8] F. de Avila Belbute-Peres, K. Smith, K. Allen, J. Tenenbaum, and J. Z. Kolter, “End-to-end differentiable physics for learning and control,” *Advances in neural information processing systems*, vol. 31, 2018.

[9] M. Geilinger, D. Hahn, J. Zehnder, M. Bächer, B. Thomaszewski, and S. Coros, “Add: Analytically differentiable dynamics for multi-body systems with frictional contact,” *ACM Transactions on Graphics*, vol. 39, no. 6, pp. 1–15, 2020.

[10] E. Heiden, D. Millard, E. Coumans, Y. Sheng, and G. S. Sukhatme, “Neuralsim: Augmenting differentiable simulators with neural networks,” in *IEEE International Conference on Robotics and Automation*. IEEE, 2021, pp. 9474–9481.

[11] T. A. Howell, S. L. Cleac’h, J. Z. Kolter, M. Schwager, and Z. Manchester, “Dojo: A differentiable simulator for robotics,” *arXiv preprint arXiv:2203.00806*, 2022.

[12] H. J. Suh, M. Simchowitz, K. Zhang, and R. Tedrake, “Do differentiable simulators give better policy gradients?” in *International Conference on Machine Learning*. PMLR, 2022, pp. 20 668–20 696.

[13] E. Todorov, T. Erez, and Y. Tassa, “Mujoco: A physics engine for model-based control,” in *IEEE/RSJ International Conference on Intelligent Robots and Systems*, 2012, pp. 5026–5033.

[14] D. Stewart and J. C. Trinkle, “An implicit time-stepping scheme for rigid body dynamics with coulomb friction,” in *IEEE International Conference on Robotics and Automation*, vol. 1, 2000, pp. 162–169.

[15] E. Coumans *et al.*, “Bullet physics library,” *Open source: bulletphysics. org*, vol. 15, no. 49, p. 5, 2013.

[16] R. Tedrake and the Drake Development Team, “Drake: Model-based design and verification for robotics,” 2019. [Online]. Available: https://drake.mit.edu

[17] W. P. Heemels, B. De Schutter, and A. Bemporad, “Equivalence of hybrid dynamical models,” *Automatica*, vol. 37, no. 7, pp. 1085–1091, 2001.

[18] A. Bemporad, F. Borrelli, and M. Morari, “Piecewise linear optimal controllers for hybrid systems,” in *American Control Conference*, vol. 2. IEEE, 2000, pp. 1190–1194.

[19] T. Marcucci and R. Tedrake, “Mixed-integer formulations for optimal control of piecewise-affine systems,” in *ACM International Conference on Hybrid Systems: Computation and Control*, 2019, pp. 230–239.

[20] F. Lauer, “On the complexity of piecewise affine system identification,” *Automatica*, vol. 62, pp. 148–153, 2015.

[21] G. Ferrari-Trecate, M. Muselli, D. Liberati, and M. Morari, “A clustering technique for the identification of piecewise affine systems,” *Automatica*, vol. 39, no. 2, pp. 205–217, 2003.

[22] A. Bemporad, “A piecewise linear regression and classification algorithm with application to learning and model predictive control of hybrid systems,” *IEEE Transactions on Automatic Control*, 2022.

[23] J.-P. Sleiman, F. Farshidian, M. V. Minniti, and M. Hutter, “A unified mpc framework for whole-body dynamic locomotion and manipulation,” *IEEE Robotics and Automation Letters*, vol. 6, no. 3, pp. 4688–4695, 2021.

[24] C. Mastalli, R. Budhiraja, W. Merkt, G. Saurel, B. Hammoud, M. Naveau, J. Carpentier, L. Righetti, S. Vijayakumar, and N. Mansard, “Crocoddyl: An efficient and versatile framework for multi-contact optimal control,” in *IEEE International Conference on Robotics and Automation*. IEEE, 2020, pp. 2536–2542.

[25] A. W. Winkler, C. D. Bellicoso, M. Hutter, and J. Buchli, “Gait and trajectory optimization for legged systems through phase-based end-effector parameterization,” *IEEE Robotics and Automation Letters*, vol. 3, no. 3, pp. 1560–1567, 2018.

[26] F. R. Hogan and A. Rodriguez, “Reactive planar non-prehensile manipulation with hybrid model predictive control,” *The International Journal of Robotics Research*, vol. 39, no. 7, pp. 755–773, 2020.

[27] M. Posa, C. Cantu, and R. Tedrake, “A direct method for trajectory optimization of rigid bodies through contact,” *The International Journal of Robotics Research*, vol. 33, no. 1, pp. 69–81, 2014.

[28] J. A. Andersson, J. Gillis, G. Horn, J. B. Rawlings, and M. Diehl, “Casadi: a software framework for nonlinear optimization and optimal control,” *Mathematical Programming Computation*, vol. 11, no. 1, pp. 1–36, 2019.

[29] A. Nagabandi, K. Konolige, S. Levine, and V. Kumar, “Deep dynamics models for learning dexterous manipulation,” in *Conference on Robot Learning*. PMLR, 2020, pp. 1101–1112.

[30] A. Allshire, M. Mittal, V. Lodaya, V. Makoviychuk, D. Makoviichuk, F. Widmaier, M. Wüthrich, S. Bauer, A. Handa, and A. Garg, “Transferring dexterous manipulation from gpu simulation to a remote real-world trifinger,” *arXiv preprint arXiv:2108.09779*, 2021.

[31] O. M. Andrychowicz, B. Baker, M. Chociej, R. Jozefowicz, B. McGrew, J. Pachocki, A. Petron, M. Plappert, G. Powell, A. Ray *et al.*, “Learning dexterous in-hand manipulationnagabandi2020deep,” *The International Journal of Robotics Research*, vol. 39, no. 1, pp. 3–20, 2020.

[32] P.-T. De Boer, D. P. Kroese, S. Mannor, and R. Y. Rubinstein, “A tutorial on the cross-entropy method,” *Annals of operations research*, vol. 134, no. 1, pp. 19–67, 2005.

[33] T. Wang, X. Bao, I. Clavera, J. Hoang, Y. Wen, E. Langlois, S. Zhang, G. Zhang, P. Abbeel, and J. Ba, “Benchmarking model-based reinforcement learning,” *arXiv preprint arXiv:1907.02057*, 2019.

[34] S. Kajita and K. Tani, “Study of dynamic biped locomotion on rugged terrain-derivation and application of the linear inverted pendulum mode,” in *IEEE International Conference on Robotics and Automation*. IEEE Computer Society, 1991, pp. 1405–1406.

[35] D. E. Orin, A. Goswami, and S.-H. Lee, “Centroidal dynamics of a humanoid robot,” *Autonomous robots*, vol. 35, no. 2, pp. 161–176, 2013.

[36] A. Pandala, R. T. Fawcett, U. Rosolia, A. D. Ames, and K. A. Hamed, “Robust predictive control for quadrupedal locomotion: Learning to close the gap between reduced-and full-order models,” *IEEE Robotics and Automation Letters*, vol. 7, no. 3, pp. 6622–6629, 2022.

[37] D. E. Stewart, “Rigid-body dynamics with friction and impact,” *SIAM review*, vol. 42, no. 1, pp. 3–39, 2000.

[38] E. Todorov, “A convex, smooth and invertible contact model for trajectory optimization,” in *IEEE International Conference on Robotics and Automation*, 2011, pp. 1071–1076.

[39] M. J. Tsatsomeros, “Generating and detecting matrices with positive principal minors,” *Asian Information-Science-Life: An International Journal*, vol. 1, no. 2, pp. 115–132, 2002.

[40] R. W. Cottle, J.-S. Pang, and R. E. Stone, *The linear complementarity problem*. SIAM, 2009.

[41] W. Jin, A. Aydinoglu, M. Halm, and M. Posa, “Learning linear complementarity systems,” *Annual Learning for Dynamics and Control Conference*, 2022.

[42] B. Stellato, G. Banjac, P. Goulart, A. Bemporad, and S. Boyd, “OSQP: an operator splitting solver for quadratic programs,” *Mathematical Programming Computation*, vol. 12, no. 4, pp. 637–672, 2020. [Online]. Available: https://doi.org/10.1007/s12532-020-00179-2

[43] S. Afriat, “Theory of maxima and the method of lagrange,” *SIAM Journal on Applied Mathematics*, vol. 20, no. 3, pp. 343–357, 1971.

[44] Y.-x. Yuan, “A review of trust region algorithms for optimization,” in *International Council for Industrial and Applied Mathematics*, vol. 99, no. 1, 2000, pp. 271–282.

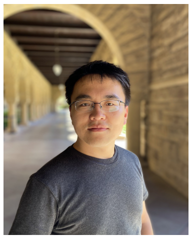
[45] A. Wächter and L. T. Biegler, “On the implementation of an interior-point filter line-search algorithm for large-scale nonlinear programming,” *Mathematical programming*, vol. 106, no. 1, pp. 25–57, 2006.

[46] M. Wüthrich, F. Widmaier, F. Grimminger, J. Akpo, S. Joshi, V. Agrawal, B. Hammoud, M. Khadiv, M. Bogdanovic, V. Berenz *et al.*, “Trifinger: An open-source robot for learning dexterity,” *arXiv preprint arXiv:2008.03596*, 2020.

[47] P. M. Wensing and D. E. Orin, “Generation of dynamic humanoid behaviors through task-space control with conic optimization,” in *IEEE International Conference on Robotics and Automation*, 2013, pp. 3103–3109.

[48] H. Dai, G. Izatt, and R. Tedrake, “Global inverse kinematics via mixed-integer convex optimization,” *The International Journal of Robotics Research*, vol. 38, no. 12-13, pp. 1420–1441, 2019.

[49] A. Rajeswaran, I. Mordatch, and V. Kumar, “A game theoretic framework for model based reinforcement learning,” in *International conference on machine learning*, 2020, pp. 7953–7963.



Wanxin Jin is an Assistant Professor in the School for Engineering of Matter, Transport, and Energy at Arizona State University in Tempe, AZ, USA. He earned his Ph.D. degree from Purdue University in West Lafayette, IN, USA, in 2021. Between 2021 and 2023, Dr. Jin held the position of Postdoctoral Researcher at the GRASP Lab, University of Pennsylvania, Philadelphia, PA, USA. At Arizona State University, Dr. Jin leads the Intelligent Robotics and Interactive Systems Lab, focusing on developing fundamental methods at the convergence of control, machine learning, and optimization, with the goal to enable robots to safely and efficiently interact with humans and physical objects.

Michael Posa received the B.S. and M.S. degrees in mechanical engineering from Stanford University, Stanford, CA, USA, in 2007 and 2008, respectively. He received the Ph.D. degree in electrical engineering and computer science from the Massachusetts Institute of Technology, Cambridge, MA, USA, in 2017. He is currently an Assistant Professor of Mechanical Engineering and Applied Mechanics with the University of Pennsylvania, Philadelphia, PA, USA, where he is a Member of the General Robotics, Automation, Sensing and Perception (GRASP) Lab. He holds secondary appointments in electrical and systems engineering and in computer and information science. He leads the Dynamic Autonomy and Intelligent Robotics Lab, University of Pennsylvania, which focuses on developing computationally tractable algorithms to enable robots to operate both dynamically and safely as they maneuver through and interact with their environments, with applications including legged locomotion and manipulation. Dr. Posa was a recipient of multiple awards, including the NSF CAREER Award, RSS Early Career Spotlight Award, and best paper awards. He is an Associate Editor for IEEE Transactions on Robotics.