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 in less than five minutes of online learning.

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 discrete set of 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 approaches struggle to train on and represent stiffness and multi-modality [1], [2], leading to learning frameworks explicitly designed to capture hybrid dynamics [3], [4]. Similar challenges exist in planning and control of contact-rich robotic 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 technical 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 into LCS and develop an LCS-based model predictive controller, 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 the 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 LCS learning [4], and the recent development of real-time LCS-based control 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 improve 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 object reorientation in simulation environment. With no prior knowledge, we achieve state-of-the-art closed-loop performance in less than five minutes of online learning.

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 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 for object reorientation. 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 without jumps: piecewise-affine (PWA) models. PWA models bring two benefits. First, they can capture 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].

Identifying PWA models is NP-hard in general [13]. Most existing methods [14], [15] 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 method [4]. We represent a PWA model as a linear complementarity system (LCS), via an implicit parameterization, and propose an implicit violation-based loss that generalizes the physics-based method in ContactNets [3]. This method does not need explicit data clustering and can handle tens of thousands of (potentially stiff) modes efficiently. Recent results have also 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, [16], [17] use the predefined sequence of mode to achieve real-time multi-contact control on legged locomotion [18] and manipulation [19]. To enable general-purpose fast multi-contact control, [6] and [7] consider 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 model predictive controller. This was first proposed in [20]. In our implementation, we utilize the state-of-the-art optimal control solver [21] for fast MPC.

3) Reinforcement Learning for Contact-Rich Manipulation: Reinforcement learning (RL) has achieved impressive results in contact-rich manipulation [22]–[24]. Some representative work includes [24], where dexterous in-hand manipulation policies are learned for object reorientation, and [23] for solving the 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 the sample inefficiency, model-based RL has been applied to robotic manipulation by first learning a dynamics model to aid policy search [22], 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 [25] have a complexity exponential to planning depth and system dimension [26].

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, 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 [27], [28] 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 [29]. 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 the 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 learn the unmodeled aspects of a reduced-order model to improve locomotion performance [29]. 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 problem is directly formulated on minimizing 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:

\[ x_{t+1} = f_i(x_t, u_t) \quad \text{with} \quad (x_t, u_t) \in \mathcal{P}_i, \]
\[ \mathcal{P}_i = \{(x, u) | \psi_i(x_t, u_t) \leq 0, \quad i \in \{1, 2, \ldots, I\} \}. \] (1)

Here, \( x_t \in \mathbb{R}^n \) and \( u_t \in \mathbb{R}^m \) are the system state and input at time step \( t = 0, 1, 2, \ldots \), respectively. The system state \( x \) and the system input \( u \) are the system state and input at time step \( t = 0, 1, 2, \ldots, T \). The system state \( x \) and the system input \( u \) are the system state and input at time step \( t = 0, 1, 2, \ldots, T \). The hybrid system is defined as a sub-level set of \( \psi_i(x_t, u_t) \). \( f_i \) is the dynamics model (vector field) in the i-th mode.

A subset of the generic hybrid systems (1) corresponds to the complementarity systems, which have been widely used to describe the multi-contact model of robot dynamics [30–32]:

\[ M(q_t)(v_{t+1} - v_t) = C(q_t,v_t) + Bu_t + \sum_{i=1}^{I} J_i(q_t)^T \Lambda_{i,t}, \] (2)

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

\[ 0 \leq \Lambda_{i,t} \perp \Phi_{i,t}(q_t, v_{t+1}, \Lambda_{i,t}) \geq 0, \quad i = 1, 2, \ldots, I. \] (3)

Here, \( (q_t, v_t) \) is the generalized coordinate and velocity of a robot system. \( u_t \) is the actual impulse with input projection matrix \( B \). \( M(q_t) \) is the inertia matrix, and \( C(q_t,v_t) \) includes all non-contact impulses resulting from gravity and gyroscopic forces. \( \Lambda_{i,t} \) is i-th contact impulse between the robot and objects or environments, and \( J_i(q_t) \) is its Jacobian matrix. The complementarity constraint (3) means either the contact impulse \( \Lambda_{i,t} \) or the value of its distance-related function, \( \Phi_{i,t}(q_t, v_{t+1}, \Lambda_{i,t}) \), is zero, but both cannot be negative, i.e., contact interaction between object and object cannot both pull or penetrate into each other. Coulomb friction can be similarly described (e.g. [32]).

Define \( x \equiv [q,v]^T \), \( \Lambda \equiv [\Lambda_1, \Lambda_1, \ldots, \Lambda_I]^T \), and \( \Phi \equiv [\Phi_1, \Phi_2, \ldots, \Phi_I]^T \). One can abstractly write the multi-contact dynamics (2) and (3) into the following general form, denoted as \( f() \):

\[ f() : \begin{cases} \Phi_t(x_{t+1}, x_t, u_t, \Lambda_t) = 0, \\ 0 \leq \Lambda_t \perp \Phi_t(x_{t+1}, x_t, u_t, \Lambda_t) \geq 0. \end{cases} \] (4)

Connecting (4) to (1), here the active or inactive constraints in \( \Phi \geq 0 \) determine the domain of hybrid modes, and \( F \) and \( \Phi \) jointly and implicitly determine the dynamics model of each hybrid mode. A significantly 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 \( f() \) contains all potential contact modes determined by the aggregated contact impulse vector \( \Lambda \) and the corresponding distance vector function \( \Phi \). Thus, we call \( f() \) the full-order dynamics. This paper focuses on learning a task-driven reduced-order model for such complementary-based hybrid systems.

B. Full-Order Model Predictive Control

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

\[ \min_{u_0, \beta} J_\beta = \sum_{t=0}^{T-1} c_\beta(x_t, u_t) + h_\beta(x_T) \] (5)

s.t. \( x_{t+1} = f(x_t, u_t), \quad \text{given} \quad x_0 \sim p_\beta(x_0), \)

where \( T \) is the MPC horizon; \( J_\beta \) is a given task cost function, parameterized by \( \beta \) sampled from a known task distribution \( \beta \sim p(\beta) \); and \( f() \) is the hybrid system in (4).

For notation simplicity, we write the system input and state trajectories compactly as \( u = \{u_0, u_1, u_2, \ldots, u_{T-1}\} \) and \( x = \{x_0, x_1, \ldots, x_{T-1}, x_T\} \), respectively. The system state trajectory given input trajectory \( u \) and initial \( x_0 \) is written as \( \mathbf{F}(u, x_0) := \{x | x_{t+1} = f(x_t, u_t), \quad \text{given} \quad x_0 \quad \text{and} \quad u \}. \) (6)

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

\[ \text{f-MPC} : \quad u^f(x_0, \beta) := \arg \min_u J_\beta(u, \mathbf{F}(u, x_0)), \]
\[ x_0 \sim p_\beta(x_0), \quad \beta \sim p(\beta). \] (7)

The full-order dynamics MPC in (7) is applied to the multi-contact robot system \( f() \) in a closed-loop (receding) fashion. At rollout time step \( k = 1, 2, 3, \ldots, x_0 \) in (7) is set to the robot’s actual state: \( x_0 = x_0^f \). After solving \( u^f(x_0^f, \beta) = \{u_0^f, u_1^f, \ldots, u_{T-1}^f\} \) from (7), only the first input \( u_0^f \) is applied to the robot for execution and drive the robot to the next state: \( x_{T+1}^f = f(x_0^f, u_0^f) \). Then, this process repeats again at the robot new state \( x_{T+1}^f \). The above MPC leads to a closed-loop control policy: mapping from robot’s current state \( x_k^f \) to its control input \( u_k^f \).

As indicated by the full-order dynamics \( f() \) in (4), solving the above MPC (7) requires reasoning over the sequence of contact impulses \( \{\Lambda_0, \Lambda_1, \ldots, \Lambda_{T-1}\} \) in addition to \( u \) and \( x \). Its combinatorial complexity is \( 2^I \) (\( I = \dim \Lambda \)). Despite recent progress, particularly on modestly sized problems [6], [8], [11], [33], 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 model \( f() \) for MPC (5) 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 \( f() \) in (5), to enable real-time control and sufficiently achieve high task performance.
C. Linear Complementarity Systems

In this paper, we aim to find a reduced-order representation for the full-order dynamics \( f() \). We consider piecewise affine (PWA) models, since they can sufficiently describe multi-modality but are 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 \( g() \),

\[
\begin{align*}
    x_{t+1} &= Ax_t + Bu_t + CA_t + d \\
    0 &\leq \lambda_t \perp Dx_t + Eu_t + F\lambda_t + c \geq 0.
\end{align*}
\] (8)

Here, the first line of (8) is the complementarity equation. \( \lambda_t \in \mathbb{R}^\prime \) is the complementarity variable, solved from the complementarity equation. \((A, B, C, d, D, E, F, c)\) are system matrix parameters with compatible dimensions. The active or inactive inequality of \( Dx_t + Eu_t + F\lambda_t + c \geq 0 \) partitions the state-input space and determines the hybrid modes of the system. Thus, the maximum number of the hybrid modes the LCS (8) can represent is \( 2^{\dim \lambda} \) (\( \dim \lambda = r \)). For any given \((x_t, u_t)\), to guarantee the existence and uniqueness of \( \lambda_t \) solved from the complementary equation (8), we impose the restriction that the symmetric part of \( F \) be positive definite, \( F^T + F \succ 0 \) [34], [35]. This is accomplished by parameterizing \( F \) as

\[
    F := GG^T + H - H^T,
\] (9)

with \( G \) and \( H \) matrices with the same dimension as \( F \).

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

\[
    \dim \lambda < \dim \Lambda.
\] (10)

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

D. Problem Formulation

In this paper, we will find a reduced-order LCS model \( g() \) in (8) for the given set of tasks \( J_\beta, \beta \sim p(\beta) \), and establish the following reduced-order \( g\)-MPC (following the notation convention used in (7)):

\[
    g\text{-MPC} : \quad u^g(x_0, \beta) := \arg \min_{u} J_\beta(u, G(u, x_0)), \quad x_0 \sim \rho(x_0), \beta \sim p(\beta),
\] (11)

such that when running the reduced-order \( g\)-MPC on the full-order robot dynamics \( f() \), one can achieve a task performance as similar to the task performance of running \( f\)-MPC on \( f() \) as possible. Here, we also compactly write the state trajectory of \( g() \) given \( u \) and \( x_0 \) as

\[
    G(u, x_0) := \{ x \mid x_{t+1} = g(x_t, u_t), \text{ given } x_0 \text{ and } u \}. \] (12)

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

\[
    \mathcal{L}(g) := \mathbb{E}_{\beta \sim p(\beta)} \mathbb{E}_{x \sim \rho(x_0)} \left[ J_\beta(u^g, F(u^g, x_0)) - J_\beta(u^f, F(u^f, x_0)) \right],
\] (13)

where the first cost \( J_\beta(u^g, F(u^g, x_0)) \) is the task performance of running reduced-order \( g\)-MPC on the robot system \( f() \), and the second cost \( J_\beta(u^f, F(u^f, x_0)) \) is the task performance of running full-order \( f\)-MPC on the robot system \( f() \). Here, \( u^g \) is the solution to the reduced-order \( g\)-MPC in (11) and \( u^f \) is the solution to the full-order \( f\)-MPC in (7). We make the following remarks on the above problem statement.

Remark 1. \( f() \) is the full-order dynamics in (4), which could be non-linear and contain a large number of hybrid modes. The reduced-order LCS model \( g() \) shares the same dimensions of states and inputs as \( f() \), but has far fewer hybrid modes by setting \( \dim \lambda \leq \dim \Lambda \). Compared to full-order \( f\)-MPC in (5), the benefits of the reduced-order \( g\)-MPC is its computational tractability due to affine structure and reduced mode count. One can take the advantage of the recent development of real-time hybrid MPC solver [6] for real-time implementation.

Remark 2. The learning criterion (13) is to minimize the performance gap between running the reduced-order \( g\)-MPC and running the full-order \( f\)-MPC. We define this via evaluating both MPC policies on the full-order dynamics \( f() \), which is the original hybrid robotic system. Therefore, a minimal task performance gap of \( g() \) means that one can confidently use the reduced-order model \( g() \) to accomplish the given set of tasks \( J_\beta, \beta \sim p(\beta) \).

Directly minimizing the task performance gap \( \mathcal{L}(g) \) requires access to and optimization with the full-order dynamics model \( f() \), because of the coupling between \( J_\beta() \) and \( f() \) 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 \( f() \).

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 \( f() \). To start, we pose a mild assumption about the Lipschitz continuity of task cost function \( J_\beta(u, x) \) for any \( \beta \sim p(\beta) \).

Assumption 1. For any task parameter sample \( \beta \sim p(\beta) \), the task cost function \( J_\beta(u, x) \) is \( M \)-Lipschitz continuous, i.e., for any \( z_1 := (u_1, x_1), z_2 := (u_2, x_2) \),

\[
    |J_r(z_1) - J_r(z_2)| \leq M \|z_1 - z_2\|
\] (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 \( \mathcal{L}(g) \) in (13):

**Lemma 1.** Suppose Assumption 1 holds. For any simplified model \( g() \), the following inequality holds:

\[
\mathcal{L}(g) \leq M \mathbb{E}_{\beta \sim p(\beta)} \mathbb{E}_{x \sim p_{\text{data}}(x_0)} \left( \left\| G(u^g, x_0) - F(u^g, x_0) \right\| + \left\| G(u^f, x_0) - F(u^f, x_0) \right\| \right)
\]

(15)

**Proof.**

\[
J_\beta(u^g, F(u^g, x_0)) - J_\beta(u^f, F(u^f, x_0))
\]

+ \[J_\beta(u^g, G(u^g, x_0)) - J_\beta(u^f, G(u^f, x_0))\]

+ \[J_\beta(u^f, G(u^f, x_0)) - J_\beta(u^f, F(u^f, x_0))\]

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

Term I \( \leq M\|G(u^g, x_0) - F(u^g, x_0)\| \)

Term III \( \leq M\|G(u^f, x_0) - F(u^f, x_0)\| \).

(16)

Term II trivially satisfies

\[
J_\beta(u^g, G(u^g, x_0)) - J_\beta(u^f, G(u^f, x_0)) \leq 0
\]

(17)

because \( u^g \) is the minimum by the definition of \( g\)-MPC (11). Putting (17) and (18) together, one has

\[
J_\beta(u^g, F(u^g, x_0)) - J_\beta(u^f, F(u^f, x_0)) \leq M \left( \|G(u^g, x_0) - F(u^g, x_0)\| + \|G(u^f, x_0) - F(u^f, x_0)\| \right).
\]

The above inequality still holds with expectation of both sides over \( x_0 \sim p(x_0) \) and \( \beta \sim p(\beta) \), yielding (15). This completes the proof.

**Remark 3.** Lemma 1 gives an upper bound for the task performance gap \( \mathcal{L}(g) \) in (13). Notably, this upper bound is the prediction error between the reduced-order model \( g() \) and full-order dynamics \( f() \) at their MPC solutions. Specifically, the first term \( \mathbb{E}_{\beta \sim p(\beta)} \mathbb{E}_{p_{\text{data}}(x_0)} \left\| G(u^g, x_0) - F(u^g, x_0) \right\| \) is the model prediction error evaluated on the dataset \( D^g = \{u^g(x_0, \beta) \mid x_0 \sim p(x_0), \beta \sim p(\beta)\} \).

(19)

generated by the reduced-order \( g\)-MPC. The second model error term \( \mathbb{E}_{\beta \sim p(\beta)} \mathbb{E}_{p_{\text{data}}(x_0)} \left\| G(u^f, x_0) - F(u^f, x_0) \right\| \) is evaluated using the dataset \( D^f = \{u^f(x_0, \beta) \mid x_0 \sim p(x_0), \beta \sim p(\beta)\} \).

(20)

Thus, one can indirectly verify the domain adaption error in (20) by additionally looking at the first-order model precision error \( \|\nabla_u F(u^g, x_0) - \nabla_u G(u^g, x_0)\| \), where \( \nabla_u F(u^g, x_0) \) can be estimated numerically via mesh grid of \( D^g \).

**Remark 4.** Lemma 1 also justifies the task-driven reduced-order model \( g() \). As long as the learned reduced-order model \( g() \) captures the full-order dynamics \( f() \) only at the MPC policy data \( D^g \) and \( D^f \), not necessarily at other parts of data regime (task-irrelevant data), \( g\)-MPC can be used to replace the full-order \( f\)-MPC with the same task performance.

**Remark 5.** \( \mathbb{E}_{p(\beta)} \mathbb{E}_{p_{\text{data}}(x_0)} \left\| G(u^f, x_0) - F(u^f, x_0) \right\| \) in (15) is called the domain adaption in reinforcement learning [36]. Specifically, if \( g() \) is trained only with the \( g\)-MPC data \( D^g \), the domain adaption term captures the model error when the trained \( g() \) is evaluated with \( f\)-MPC data \( D^f \). This domain adaption term is inevitable if we want a learned proxy model \( g() \) trained on its generated data to capture the true dynamics \( f() \) which is unknown.

Although the full-order \( f\)-MPC data \( D^f \) is not directly verifiable when \( f() \) is unknown, the next lemma will show the \( g\)-MPC data \( D^g \) is related to \( D^f \), which thus can be verified indirectly.

**Lemma 2.** Suppose \( \nabla_x J_\beta(u, x) \) is \( L_1\)-Lipschitz continuous, \( \nabla_u J_\beta(u, x) \) is \( L_2\)-Lipschitz continuous, and \( \|\nabla_x J_\beta(u, x)\| \leq M_1 \) for any \( (u, x, \beta) \) \( \|\nabla_u G(u, x_0)\| \leq M_2 \) for any \( (u, x_0) \). Then, the solutions \( D^g \) in (19) generated by \( g\)-MPC is also an \( \epsilon\)-accuracy stationary solution for \( f\)-MPC, i.e.,

\[
\|\nabla_u J_\beta(u^g, F(u^g, x_0))\| \leq \epsilon
\]

(21)

for any \( x_0 \sim p(x_0), \beta \sim p(\beta) \) with

\[
\epsilon = M_1 \|\nabla_u F(u^g, x_0) - \nabla_u G(u^g, x_0)\| + (L_2 + M_2 L_1) \|F(u^g, x_0) - G(u^g, x_0)\|
\]

(22)

**Proof.** See Appendix A.

**Remark 6.** Lemma 2 suggests the \( g\)-MPC data \( D^g \) in (19) can become \( f\)-MPC data \( D^f \) in (20), if the reduced-order model \( g() \) fits well to the true \( f() \) in both zeroth and first orders, i.e.,

\[
\|F(u^g, x_0) - G(u^g, x_0)\| \to 0
\]

(23)

\[
\|\nabla_u F(u^g, x_0) - \nabla_u G(u^g, x_0)\| \to 0
\]

(24)

Thus, one can indirectly verify the task performance gap \( \mathcal{L}(g) \) itself. Those theoretical insights will guide us to develop algorithms in the next session.

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 \( g() \) to full-order dynamics \( f() \) well in both zeroth and first-order prediction using the \( g\)-MPC policy data \( D^g \). 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, $g$-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 $f()$.

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 $g()$ is deployed 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 will be adapted according to the latest Rollout Buffer with details given in Section V-B.

- **Rollout Buffer**: denoted as $D_{buffer} = \{(x^f_k, u^g_{\text{MPC}} k, x^f_{k+1})\}$, stores the current and history rollout data from running the trust-region $g$-MPC controller on the robot (full-order dynamics). The buffer can permit a maximum buffer size.

- **Learning reduced-order LCS**: This is to train reduced-order LCS $g()$ using the data of the latest Rollout Buffer $D_{buffer}$. Details of the training process is given in Section V-A.

### A. Learning Reduced-Order LCS

We use the method in our recent work [37] to learn the LCS (8) from Rollout Buffer data $D_{buffer} = \{(x^f_k, u^g_{\text{MPC}} k, x^f_{k+1})\}$. This method enables efficient learning a PWA model with up to thousands of hybrid modes and effectively handles the stiff dynamics that arises from contact. For self-containment, the method is briefly described below.

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

$$\theta := \{A, B, C, d, D, E, G, H, c\}. \quad (25)$$

In [37], we presented a new learning method, which jointly learns $\theta$ by minimizing the following violation-based loss

$$\mathcal{L}_{\text{vio}}(\theta, D_{buffer}) :=$$

$$\min_{A_k \geq 0, \phi_k \geq 0} \frac{1}{2} \|Ax^f_k + Bu^g_{\text{MPC}} k + C\lambda_k + d - x^f_{k+1}\|^2 +$$

$$\frac{1}{\epsilon} \left(\lambda^2_k \phi_k + \frac{1}{2\gamma} \|Dx^f_k + Eu^g_{\text{MPC}} k + F\lambda_k + c - \phi_k\|^2\right) \quad (26)$$

In the above violation-based loss, the first and second terms are the violation of the dynamics and complementarity equations by the buffer data point $(x^f_k, u^g_{\text{MPC}} k, x^f_{k+1})$, respectively. Here, $\phi \in \mathbb{R}^T$ is an introduced slack variable; $\epsilon > 0$ is a hyperparameter (empirically taking from 10^0 to 10^{-3}; and $\gamma > 0$ can be any value as long as satisfying $\gamma \leq \sigma_{\text{min}}(F^T + F)$ (i.e., the smallest singular value of matrix $(F^T + F)$). See [37] for detailed explanations and experiments of those hyperparameters.

As theoretically shown in [37], the above LCS training loss $\mathcal{L}_{\text{vio}}(\theta, D_{buffer})$ has the following properties. First, the inner optimization over $(\lambda_k, \phi_k)$ is convex quadratic program, which can be efficiently solved in batch using state-of-the-art solvers such as OSQP [38]. Second, the gradient of the violation-based loss $\mathcal{L}_{\text{vio}}(\theta, D_{buffer})$ with respect to all matrices in $\theta$ can be analytically obtained using the Envelope Theorem [39]; unlike methods based on the Implicit Function Theorem (e.g. [8]), this avoids differentiating through the solution. Third, most importantly, by adding both the dynamics violation and complementarity violation with a balance weight $\epsilon$, $\mathcal{L}_{\text{vio}}(\theta, D_{buffer})$ has a better conditioned loss landscape, which enables simultaneous identification of stiff and multimodal dynamics.

### B. Trust-Region LCS Model Predictive Controller

With the reduced-order LCS $g()$, one can establish the following trust-region reduced-order LCS-based model predictive
controller:

\[
\min_{u_{0:T-1}} \sum_{t=0}^{T-1} c\beta(x_t, u_t) + h\beta(x_T) \quad \beta \sim p(\beta)
\]

subject to \( u_t \in [\bar{u} - \Delta, \bar{u} + \Delta] \)  

\[
x_{t+1} = Ax_t + Bu_t + Cl_t + d \\
0 \leq \lambda_t - Dx_t + Eu_t + Fl_t + c \geq 0.
\]

\( x_0 = x_f \)

Compared to the early \( g \)-MPC in (11), the difference here is that we have enforced a trust region constraint \( \bar{u} - \Delta \leq u_t \leq \bar{u} + \Delta \) on the control input \( u_t \), \( t = 0, 1, \ldots, T-1 \). This is due to the following reasons. As shown in Fig. 1, since the reduced-order LCS \( g() \) is trained on the current buffer data \( D_{\text{buffer}} \), we expect \( g() \) is likely valid only on the region covered by \( D_{\text{buffer}} \), which we refer to as the trust region. Thus, we constrain \( g \)-MPC in (27) to this trust region, prohibiting the controller from attempting to exploit model error and generating undesired controls.

It is important that the center \( \bar{u} \) and size \( \Delta \) of the trust region in (27) is updated along with the rollout buffer \( D_{\text{buffer}} \) during each iteration. In our algorithm, \( i \)-th iteration, we set the trust region center \( \bar{u} \) as the mean of all control input data in the current Rollout Buffer \( D_{\text{buffer},i} \), i.e.,

\[
\bar{u}_i = \text{mean}\left(\{u_{i}^{g\text{-MPC}}, \ldots, u_{k}^{g\text{-MPC}}, \ldots\}\right), \quad u_{k}^{g\text{-MPC}} \in D_{\text{buffer},i}
\]

and the trust region size \( \Delta_i \) is set according to the standard deviation of all input data in \( D_{\text{buffer},i} \):

\[
\Delta_i = \eta_i \text{std}\left(\{u_{1}^{g\text{-MPC}}, \ldots, u_{k}^{g\text{-MPC}}, \ldots\}\right), \quad u_{k}^{g\text{-MPC}} \in D_{\text{buffer},i}
\]

Here, \( \eta_i > 0 \) is a hyperparameter of the trust region at the \( i \)-th iteration, and mean() and std() operations are applied dimension-wise. It is also possible that \( \bar{u}_i \) and \( \Delta_i \) are set using more complex rules, e.g., following the classic trust-region optimization [40]. But we found that the above strategy is simple, flexible, and works well in practice.

To quickly solve the LCS MPC in (27), we apply the direct method of trajectory optimization [20]. Specifically, the optimization simultaneously searches over the trajectories \( x_{0:T}, u_{0:T-1}, \lambda_{0:T-1} \) by treating the LCS and trust region as the separate constraints imposed at each time step. We solve such nonlinear optimization using CasADi [21] differentiation interface packed with the IPOPT solver [41]. In our later applications, as the reduced-order LCS (27) has a relatively small number of hybrid modes, such as \( \text{dim} \lambda \leq 5 \) and a small MPC horizon \( T = 5 \), we can solve (27) with real-time MPC performance (e.g., MPC running frequency can reach 50Hz). For higher dimensional CLS and planning horizon, one can also use the recent real-time multi-contact MPC method [6].

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

### VI. MODEL REDUCTION FOR HYBRID CONTROL SYSTEM

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, with code available at https://github.com/wanxinjin/Task-Driven-Hybrid-Reduction.

#### A. Problem Setting

Consider the MPC of a general PWA system [33]

\[
\min_{x_{0:T-1}} \sum_{t=0}^{T-1} c(x_t, u_t) + h(x_T) \\
\text{s.t.} \quad f_j : \quad \{x, u\} | D_j x + E_j u + h_j = 0 \\
\quad j \in \{1, 2, \ldots, I\}
\]

\( x_0 \) given

Here, \( P_j, j \in \{1, 2, \ldots, I\} \), is the \( j \)-th polytope partition of the state-input space. Dynamics in \( P_j \) is \( x_{t+1} = A_j x_t + B_j u_t + c_j \). The total number of hybrid modes of the above PWA system is \( I \). Solving (30) can be generally treated as a mixed-integer program with \( I \) 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 \( g() \) which maintains a small budget of hybrid modes, such that running \( g \)-MPC can achieve similar performance as running the full-order \( f \)-MPC in (30). Here, the reduced-order LCS \( g() \) in (8) has the same dimensions of \( x \) and \( u \) as \( f() \), but we set \( \text{dim} \lambda \) such that its maximum number of hybrid modes of \( g() \) is far less than \( f()'s \), i.e., \( 2^{\text{dim} \lambda} \ll I \).

#### B. Experiment Settings

In the simulation presented below, we set the cost function \( J \) in (30) as a quadratic cost

\[
J = \sum_{t=0}^{T-1} (x_t^T Q x_t + u_t^T R u_t) + x_T^T Q_T x_T,
\]

where

\[
\text{Algorithm 1: Task-driven hybrid model reduction}
\]

**Initialization:** Initial reduced-order LCS model \( g_{\theta_0} \);  
Initial Buffer \( D_{\text{buffer},0} \) (by random policy);  
Trust region parameter schedule \( \{\eta_i\} \)

for \( i = 0, 1, 2, \ldots \) do

\[
% Reduced-order model update
\]

Train reduced-order LCS \( g_{\theta_{i+1}} \) with the data from current Rollout Buffer \( D_{\text{buffer},i} \) \( \Rightarrow \theta_{i+1} \) [Section V-A]

\[
% Compute the trust region
\]

Compute the trust region from the current Rollout Buffer \( D_{\text{buffer},i} \) \( \Rightarrow \left[ \bar{u}_i - \Delta_i, \bar{u}_i + \Delta_i \right] \) [see (28) and (29)]

\[
% MPC rollout and update Buffer
\]

With current LCS \( g_{\theta_{i+1}} \) and current trust region \( \left[ \bar{u}_i - \Delta_i, \bar{u}_i + \Delta_i \right] \), run the trust-region LCS MPC policy in (27) on the robot, collect new rollout data \( \{(x_f^i, u_k^{g\text{-MPC}}, x_{f+1}^i)\} \) and add it to Rollout Buffer:

\[
D_{\text{buffer},i+1} \leftarrow D_{\text{buffer},i} \cup \{(x_f^i, u_k^{g\text{-MPC}}, x_{f+1}^i)\};
\]
end
with all weight matrices being identities. We run both \( f \)-MPC and \( g \)-MPC policies on full-order dynamics \( f() \) in a closed-loop (receding) fashion, though noting that \( f \)-MPC cannot be solved in real-time for our more complex examples. The initial state \( x_0 \) of the full-order dynamics \( f() \) is subject to a uniform distribution \( x_0 \sim U[-4, 4] \).

In Algorithm 1, the hyperparameters are listed in Table I. A 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 [37].

<table>
<thead>
<tr>
<th>Parameter(^1)</th>
<th>Symbol</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>MPC horizon</td>
<td>( T )</td>
<td>5</td>
</tr>
<tr>
<td>Rollout horizon</td>
<td>( H )</td>
<td>15 ( \sim ) 20</td>
</tr>
<tr>
<td># of new rollouts added to buffer per iter.</td>
<td>( R_{\text{new}} )</td>
<td>5</td>
</tr>
<tr>
<td>Maximum buffer size</td>
<td>( R_{\text{buffer}} )</td>
<td>50 rollouts</td>
</tr>
<tr>
<td>Trust region hyperparameter</td>
<td>( \theta_t )</td>
<td>20, 5i</td>
</tr>
<tr>
<td>Initial guess ( \theta ) in (25) for ( g() )</td>
<td>( \theta_0 )</td>
<td>( U[-0.5, 0.5] )(^2)</td>
</tr>
</tbody>
</table>

\(^1\) Other settings not listed here will be stated in text.
\(^2\) \( U[-0.5, 0.5] \) means uniform distribution in range \([-0.5, 0.5] \).

C. Results and Analysis

1) Illustration of Learning Progress: In this session, we use a randomly generated full-order dynamics \( f() \) in (30). Specifically, all matrices \( (A_j, B_j, c_j, D_j, E_j, h_j) \), \( i = 1, 2, \ldots , I \), are sampled from uniform distributions, with dimension \( x \in \mathbb{R}^2 \) and \( u \in \mathbb{R} \), and the mode count \( I \approx 120 \) for random sampling of \( x_0 \sim U[-4, 4] \) and \( u \sim U[-10, 10] \). In the reduced-order LCS \( g() \) in (8), we take \( \dim \lambda = 2 \), meaning that the maximum number of modes in \( g() \) is 4, far fewer than \( I \) of the full-order dynamics.

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

\[
x_{i+1} = f(x_i, u_i) = f(x_i, \text{MPC}(x_i))
\]

where the MPC controller \( u_i = \text{MPC}(x_i) \) can be either the full-order \( f \)-MPC (30) or the learned reduced-order \( g \)-MPC. Specifically, Fig. 2a shows the phase portrait of running \( f \)-MPC controller, where different colors show different hybrid modes in \( f() \). Fig. 2b shows the phase portrait comparisons between \( f \)-MPC controller (blue) and \( g \)-MPC controller (orange) before learning. Fig. 2c shows the phase portrait for \( g \)-MPC controller after learning, where different modes of \( g \)-MPC are shown in different colors. Fig. 2d compares the phase plot between \( f \)-MPC controller (blue) and \( g \)-MPC controller (orange) after learning.

Although the full-order dynamics \( f() \) has around \( I = 120 \) modes for random data \( x \sim U[-4, 4] \) and \( u \sim U[-10, 10] \), Fig. 2a shows 42 hybrid modes in \( f() \) with \( f \)-MPC controller. One can notice that some modes correspond to small portion of the state space, e.g., orange and green (near origin), and thus, most of \( f \)'s task-relevant motion (arrows) will not enter into or quickly pass those modes. This makes those modes less important for the given task of minimizing (31). On the other hand, some other modes account for a large portion of the state space, such as cyan and gray. Most of \( f \)'s motion will enter into or stay in those modes, making them dominant for the minimizing the task cost (31). In Fig. 2c, after learning, the reduced-order model \( g() \) has only 4 hybrid modes (recall \( \dim \lambda = 2 \)), which successfully capture the important modes in Fig. 2a. Comparing the phase portraits between the full-order \( f \)-MPC controller and the reduced-order \( g \)-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 \( f() \) in (30) also using LCS representation, as in (8). All system matrices in \( f() \) are drawn from uniform distribution. We use \( \Lambda \) to denote the complementarity variable of \( f() \). In the reduced-order LCS model \( g() \) in (8), we vary \( \dim \lambda \) to demonstrate the effect of varying degrees of mode reduction.

In Table II, we consider different cases of full-order \( f() \), 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 metric to report the results of the learned task-driven reduced-order LCS \( g() \).
TABLE II: Task-driven model reduction for hybrid control systems

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

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

Fig. 3: An example instance of f()’s rollout with full-order f-MPC controller or reduced-order g-MPC controller, corresponding to two cases in Table II. In each case, the left column is a single rollout of running f-MPC controller, and the right running g-MPC controller. The upper row shows the activation of Λ or λ over time (black brick means nonzero and blank means zero). The bottom row shows the state trajectory x_{t+1} = f(x_t, MPC(x_t)) 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 x_0), there are not many active hybrid modes of f involved in a single trajectory.

- **Random Policy, number of modes in f**: This is the total number of the hybrid modes that are active in the full-order dynamics f(), when one runs a random policy with initial state sampling from x_0 ~ U[−4, 4] and input sampling from u ~ U[−10, 10].

- **Random Policy, ME(g)(%)**: This is the relative prediction error of the learned reduced-order LCS model g() evaluated on the above random policy data, defined as
  \[
  \frac{\|g(x, u^{\text{rand}}) - f(x, u^{\text{rand}})\|_2^2}{\|f(x, u^{\text{rand}})\|_2^2 + 10^{-6}} \times 100\%. \tag{33}
  \]

- **g-MPC Policy, number of modes in f**: This is the total number of the hybrid modes that are active in the full-order dynamics f() when one runs the learned reduced-order g-MPC controller on it with initial state sampling from x_0 ~ U[−4, 4].

- **g-MPC Policy, ME(g)(%)**: This is the relative prediction error of the learned reduced-order LCS g() evaluated at the above g-MPC policy data, defined as
  \[
  \frac{\|g(x, u^g_{\text{MPC}}) - f(x, u^g_{\text{MPC}})\|_2^2}{\|f(x, u^g_{\text{MPC}})\|_2^2 + 10^{-6}} \times 100\%. \tag{34}
  \]

- **g-MPC Policy, number of modes in g**: This is the total number of hybrid modes that are active inside the g-MPC controller, which is run on the full-order dynamics f() with initial state sampling from x_0 ~ U[−4, 4].
• $\mathcal{L}(g)(\%)$ is the relative task performance gap between $g$-MPC controller and $f$-MPC controller, defined as

$$\mathcal{L}(g) = \frac{J(g\text{-MPC}) - J(f\text{-MPC})}{J(f\text{-MPC})} \times 100\%, \quad (35)$$

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

$$J(g\text{-MPC}) = \mathbb{E}_{\beta \sim p(\beta)} \mathbb{E}_{x \sim p_\beta(x_0)} \sum_{t=0}^H c_\beta(x_t^f, u_t^{g\text{-MPC}}), \quad (36)$$

with $\{x_0^f, u_0^{g\text{-MPC}}\}$ being a rollout of running $g$-MPC controller on the full-order dynamics $f()$. The similar definition applies to $J(f\text{-MPC})$.

In each experiment case in Table II, we run the algorithm for ten trials with different random seeds, and the full dynamics $f()$ in each trial is also randomized. All results in Table II are reported using mean and standard derivation over all trials.

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

(i) Jointly looking at the number of modes in $f$ with random policy (fourth column), the number of modes in $g$-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 $f()$, and it can result in a similar MPC control performance as using full-order MPC policy, with a performance loss less than 2% - 3%.

(ii) Comparison between the mode counts in $f$ with random policy (fourth column) and that with $g$-MPC policy (sixth column) can clearly confirm the motivating hypothesis of this paper: a much fewer hybrid modes are actually necessary to achieve the task (here is to minimize the given cost function (31), and vast majority of the hybrid modes in $f$ will remain untouched throughout the control process.

(iii) Notably, comparing the relative model error of the learned reduced-order LCS $g()$ on random policy data (fifth column) and on the $g$-MPC policy data (seventh column), one can conclude that the learned reduced-order LCS $g()$ 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 on Cases 3-6. Here, under the same other conditions, we used an increasingly complex reduced-order LCS $g$ from $\dim \lambda = 1$ to $\dim \lambda = 5$. The results show that increasing the hybrid mode budget in $g()$ can lead to a performance improvement, although small, in the model accuracy (seventh column) and the relative task performance gap (last column).

Lastly, corresponding to Table II, single instances of $f()$’s rollout with full-order $f$-MPC controller and with the reduced-order $g$-MPC controller are compared in Fig. 3. The upper row in each case shows the mode activation, i.e., nonzero (black) or zero (blank) in each dimension of $\Lambda$ or $\lambda$, in $f$-MPC (left) and $g$-MPC (right) along rollout, respectively. The bottom row shows the state trajectory $x_{t+1} = f(x_t, \text{MPC}(x_t))$ for the $f$-MPC controller (left) and the $g$-MPC controller (right), and different colors representing different hybrid modes. Note that since Fig. 3 only shows a single instance of rollout (from a fixed $x_0$), there are not many active hybrid modes involved in a single trajectory.

D. Effect of Hyperparameter Settings

In this session, 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$(g)(\%)$ in (34) and the relative task performance gap $\mathcal{L}(g)(\%)$ in (35) to report the results. The dimensions of the full-order and reduced-order systems and other settings follow Case 1 in Table II. We show the results in Fig. 4.
Fig. 4 shows 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 $\eta_t$ does not significantly influence the learning performance. Overall, Fig. 4 shows 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 [42]. The Python codes are available at https://github.com/wanxinjin/Task-Driven-Hybrid-Reduction.

A. Three-Finger Dexterous Manipulation

As illustrated in Fig. 5, the three-finger dexterous manipulation system includes three 3-DoF robotic fingers and a cube with a table. The manipulation goal is to find a the control policy for the three fingers to move the cube to random target poses. The entire simulation environment uses MuJoCo physics engine [43]. In this paper, we consider two specific tasks.

Fig. 5: Three-finger dexterous manipulation tasks. Left: the three robotic fingers need to turn the cube to a random target orientation, given by a reference in the left corner. Right: the three robotic fingers need to move the cube to a random target pose with random position and orientation, given by the shaded reference. The simulation environment uses MuJoCo physics engine [43].

Manipulation task 1: Cube Turning. As shown in the left 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 table. There is friction between the cube and table and also damping in the joint of cube rotation. Thus, the cube here resembles a ‘valve’ on the table. Three fingers need to turn the cube to any random target orientation $\alpha_{\text{goal}}$, sampled from a uniform distribution:

$$\beta = \alpha_{\text{goal}} \sim U[-1.5, 1.5] \quad \text{(radius)}$$  \hfill (37)

For visualization, the target orientation is shown at the bottom left corner.

Manipulation task 2: Cube Moving. As shown in the right figure 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 $\beta = (p_{\text{goal}}, \alpha_{\text{goal}})$ on the table, where $p_{\text{goal}} \in \mathbb{R}^2$ (center of mass) is cube’s target xy-position and $\alpha_{\text{goal}}$ is the cube’s target orientation angle on the table, both sampled from uniform distribution

$$p_{\text{goal}} \sim U[-P_{\text{max}}, P_{\text{max}}], \quad P_{\text{max}} = [0.06, 0.06]^T (\text{m}),$$

$$\alpha_{\text{goal}} \sim U[-0.5, 0.5] \quad \text{(rad)}.$$  \hfill (38)

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 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, and use it 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 $f()$ (and $f$-MPC) for ground truth comparison.

B. Experiment Settings

Before proceeding, we here clarify the state and input spaces in the reduced-order LCS model $g()$ 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

$$x = [p_{\text{cube}}, \alpha_{\text{cube}}, p_{\text{fingertip}}]^T \in \mathbb{R}^9,$$  \hfill (39)

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

$$u = \Delta p_{\text{fingertip}} \in \mathbb{R}^6,$$  \hfill (40)

which includes the incremental position of each fingertip. We use operational space control (OSC) [44] in the lower level to map from $u$ 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 $g()$ in (8), we set

$$\dim \lambda = 5$$  \hfill (41)

for both manipulation tasks. This means that the reduced-order model can maximally represent $2^5 = 32$ hybrid modes which is far fewer than the estimated thousands of modes in the full-order dynamics of the three-finger manipulation system.
2) Cost Functions: For both manipulation tasks, we define the following cost function $J_β$ 

$$J_β = \sum_{t=0}^{T-1} c_β(x_t, u_t) + h_β(x_T), \quad β \sim p(β) \quad (42)$$

where $p(β)$ is (37) for the cube turning task and (38) for the cube moving task, and 

$$c_β = \|P_{\text{fingertip}} - P_{\text{cube}}\|^2 + \|P_{\text{cube}} - P_{\text{goal}}\|^2 + \alpha_{\text{cube}} - \alpha_{\text{goal}} )^2 + 0.01 \|u\|^2,$$

$$h_β = \|P_{\text{fingertip}} - P_{\text{cube}}\|^2 + \|P_{\text{cube}} - P_{\text{goal}}\|^2 + \alpha_{\text{cube}} - \alpha_{\text{goal}} )^2 . \quad (43)$$

Here, cost term $\|P_{\text{fingertip}} - P_{\text{cube}}\|^2$ penalizes the distance between fingertips and center of the cube, i.e. encouraging contact between fingertips and cube; $\|P_{\text{cube}} - P_{\text{goal}}\|^2$ and $(\alpha_{\text{cube}} - \alpha_{\text{goal}} )^2$ are the squared distance to the target position or orientation, respectively; and 0.01 $\|u\|^2$ penalize the control cost. $w^c = [w_1^c, w_2^c, w_3^c]^T$ and $w^h = [w_1^h, w_2^h, w_3^h]^T$ are the cost weights, whose values will be given later.

The other hyperparameters of Algorithm 1 are listed in the following table, which largely follows ones in Table I in the first application (recall the discussion of the hyperparameter settings are in Section VI-D).

TABLE III: Hyperparameters for three-finger manipulation

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Symbol</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>MPC horizon</td>
<td>$T$</td>
<td>5</td>
</tr>
<tr>
<td>Rollout horizon</td>
<td>$H$</td>
<td>20</td>
</tr>
<tr>
<td># of new rollouts added to buffer per iter.</td>
<td>$R_{\text{new}}$</td>
<td>5</td>
</tr>
<tr>
<td>Maximum buffer size</td>
<td>$R_{\text{buffer}}$</td>
<td>200 rollouts</td>
</tr>
<tr>
<td>Trust region parameter</td>
<td>$\eta_\ell$</td>
<td>1.0</td>
</tr>
<tr>
<td>Initial guess $\theta$ in (25) for $g(\cdot)$</td>
<td>$\theta_0$</td>
<td>$U[-0.5, 0.5]$</td>
</tr>
</tbody>
</table>

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 ($g$-MPC controller) rollouts on the environment, which is proportional to the learning iteration (each iteration collects 5 new on-policy rollouts). Specifically, Fig. 6a shows the relative prediction error of the reduced-order LCS model $g()$ evaluated on the on-policy rollout data, defined in (34), where $f(x, u^g_{\text{-MPC}})$ is the direct observation of the next state of the environment. Fig. 6b shows the total cost of a rollout from running $g$-MPC controller on the environment, defined in (36). Fig. 6c shows the orientation cost of a rollout from running $g$-MPC controller on the environment, defined as 

$$E_{β \sim p(β)} E_{x \sim p(x|α)} \sum_{t=0}^{H} (α_{\text{cube}}, t - α_{\text{goal}} )^2 . \quad (45)$$

Fig. 6b and Fig. 6c show the very similar pattern because the orientation cost term $(α_{\text{cube}} - α_{\text{goal}} )^2$ dominates in (43) relative to other cost terms in scale. Fig. 6d shows the trust region upper bound $\bar{u} + Δ$ and lower bound $\bar{u} - Δ$ for the first dimension of the control input.

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

$$\frac{\tau_{\text{mag}}^{\text{disturb}}}{I_{\text{cube}}} \quad \text{with } I_{\text{cube} \text{ the inertia of cube}}. \quad (46)$$

Fig. 6 and Table IV show the efficiency of using 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 with-
Fig. 7: Two rollouts of running the reduced-order \( g \)-MPC controller on the three-finger system. (a) and (b): the target orientation is \( \alpha_{\text{goal}} = -1.29 \) (rad). (c) and (d): the target orientation is \( \alpha_{\text{goal}} = -0.148 \). The upper panel of (a) or (c) shows the mode activation \( \text{sign}(\lambda) \) in \( g() \) over rollout time. Here, black bricks show \( \lambda > 0 \) and blank \( \lambda = 0 \). 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 environment, corresponding to (a) or (c), respectively. Here, the upper panels show the environment snapshots, and lower panels show the zoom-in interaction details. Explanation and analysis are given in text and Tables V and VI.

TABLE IV: Results for the manipulation task of Cube Turning.

<table>
<thead>
<tr>
<th>Results</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Total number of hybrid modes in ( g )-MPC</td>
<td>(around) 14</td>
</tr>
<tr>
<td>Cube terminal orientation error (</td>
<td>\alpha_{\text{cube},H} - \alpha_{\text{goal}}</td>
</tr>
<tr>
<td>Terminal error (relative) ( \frac{</td>
<td>\alpha_{\text{cube},H} - \alpha_{\text{goal}}</td>
</tr>
<tr>
<td>Total training time of the algorithm(^1)</td>
<td>4.1 ± 0.1 mins</td>
</tr>
<tr>
<td>Running frequency of reduced-order ( g )-MPC(^1)</td>
<td>&gt;50 Hz</td>
</tr>
<tr>
<td>% of stick-slip-separate modes in rollouts (approx.)</td>
<td>&gt;70%(^2)</td>
</tr>
<tr>
<td>Maximum ( \frac{\text{tau}}{\text{dist}} ) until (</td>
<td>\alpha_{\text{cube},H} - \alpha_{\text{goal}}</td>
</tr>
</tbody>
</table>

\(^1\) The experiments are tested on MacBook Pro with M1 Pro chip.
\(^2\) This is approximated calculated by observing the simulation rollouts by running the learned reduced-order \( g \)-MPC.

out any prior knowledge within just 5 minutes of wall-clock time, including real-time closed-loop control experiments.

(ii) The learned task-driven reduced-order LCS \( g() \) 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 \( \frac{|\alpha_{\text{cube},H} - \alpha_{\text{goal}}|}{|\alpha_{\text{goal}}|} < 5\% \).

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

(iv) The learned reduced-order LCS \( g() \) 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 \( g \)-MPC controller shows high robustness against large external torque disturbances. The robustness is a natural benefit of the closed-loop implementation of \( g \)-MPC controller. Such a high robust performance could also partially because of the high stiffness gain in our lower-level OCS control.

2) Analysis of Reduced-Order Hybrid Modes: In this session, we will detail how the learned task-driven reduced-order LCS model \( g() \) enables the three-finger system to reason about the contact decision in the cube tuning task. Given two random target orientations, Fig. 7 shows the rollout trajectories (left) of running the learned reduced-order \( g \)-MPC on the three-finger
TABLE V: Empirical correspondence between LCS mode activation in Fig. 7a and physical contact interaction in Fig. 7b.

<table>
<thead>
<tr>
<th>Time $t$</th>
<th>Mode activation in $g()$</th>
<th>Interaction between fingertips and cube$^1$</th>
<th>Interaction between cube and table$^1$</th>
</tr>
</thead>
<tbody>
<tr>
<td>$t = 0, 1, ..., 8$</td>
<td>$\text{sign}(\lambda) = [0, 0, 0, 1, 0]^T$</td>
<td>(R, G, B separate) or (G, B separate and R touching)</td>
<td>Cube stick to table or CW rotational slip</td>
</tr>
<tr>
<td>$t = 9$</td>
<td>$\text{sign}(\lambda) = [0, 1, 1, 1, 0]^T$</td>
<td>G separate and R, B touching</td>
<td>CW rotational slip</td>
</tr>
<tr>
<td>$t = 10$</td>
<td>$\text{sign}(\lambda) = [0, 0, 1, 1, 0]^T$</td>
<td>R right slip and G separate and B stick</td>
<td>CW rotational slip</td>
</tr>
<tr>
<td>$t = 11, ..., 14$</td>
<td>$\text{sign}(\lambda) = [1, 1, 1, 1, 0]^T$</td>
<td>R, G, B touching</td>
<td>CW rotational slip</td>
</tr>
<tr>
<td>$t = 15, ..., 20$</td>
<td>$\text{sign}(\lambda) = [0, 1, 0, 1, 0]^T$</td>
<td>G touching and R, B separate</td>
<td>CW rotational slip</td>
</tr>
</tbody>
</table>


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

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

For the cube CW rotational slip (blue segments), while $\text{sign}(\lambda_i) = [1, 1, 0, 0, 0]^T$ and $\text{sign}(\lambda_j) = [1, 0, 0, 0, 0]^T$ are for CCW rotational slip (green and purple segments).

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

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

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

D. Manipulation Task 2: Cube Moving

This session presents the results and analysis of using the proposed method to solve Cube Moving manipulation task. In this task, we set weights in (43) as

$$w^c = [12.0 \ 200.0 \ 0.2]^T,$$

$$w^b = [6.0 \ 200.0 \ 1.0]^T.$$  \hspace{1cm} (47)

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 reduced-order LCS $g()$ evaluated on the $g$-MPC policy data, defined in (34). Fig. 8b shows the total cost of a rollout with the $g$-MPC controller, defined in (36). Fig. 8c shows the position cost of a rollout with the $g$-MPC controller, defined as

$$E_{\beta \sim p(\beta)} \mathbb{E}_{x_0 \sim p(x_0)} \sum_{t=0}^{H} \| p_{\text{cube},t} - p_{\text{goal}} \|^2.$$  (48)

(i) Without any prior knowledge, the proposed method learns a reduced-order LCS to successfully solve the Cube Moving manipulation task within 5 minutes of wall-clock time. The reduced-order model $g()$ leads to a real-time $g$-MPC controller with running frequency > 30 Hz.

(ii) The learned reduced-order LCS model $g()$ only permits 32 modes (among them 15 are active for the task), which is much fewer than the estimated number of hybrid modes in full-order dynamics (unknown), which could be thousands. In the following session, we will details how such fewer modes in the learned reduced-order LCS $g()$ can capture the rich contact interaction of the manipulation task.

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

2) Analysis of Reduced-Order Hybrid Modes: Fig. 9 shows one rollout of running the reduced-order LCS $g$-MPC on the three-finger manipulation system. Fig. 9a plots the trajectory of mode activation in $g()$ (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. 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 of $g$-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 manipulation task. Different hybrid mode of the reduced-order $g$ can approximately capture different contact interactions during the rollout. Those interactions include separate, stick, and slip between the fingertips and the cube, stick, CCW rotational slip, and CW rotational slip between the cube and the table, and contact $G$ between 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 mode in $g()$ 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.

![Graphs and tables](image.png)
3) Generation of Different Manipulation Strategies: Most notably, for different random target poses (sampled from (38)), we observe that the learned reduced-order $g$-MPC can produce different three-finger 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 $g$ in its $g$-MPC policy rollout given different targets. The first row in Fig. 10 shows the mode activation in $g$ during the rollout of the $g$-MPC policy; the second row shows the snapshot of the environment at the end of rollout; and third row shows the zoom-in details of the contact interaction. The bottom title in each column describes the main physical interactions for each strategy. From Fig. 10, we have the following observations and comments.

(i) Fig. 10 clearly shows the learned reduced-order LCS $g()$ enables generating different strategies for different targets. Particularly, $g()$ 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...
Fig. 10: Different contact strategies generated by the same learned reduced-order LCS $g()$ in its $g$-MPC policy rollout given different targets. The first row shows the mode activation of $g$ over the rollout; the second row shows the snapshot of the environment at the end of the rollout; and 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.

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 $g$ 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 because the cube during this period is all still and all fingertips are separate from the cube. Around after $t > 4$, different modes in $g$-MPC begin to activate, leading to different manipulation strategies.

(iii) Recall that all strategies shown in Fig. 10 are generated the same learned reduced-order LCS $g()$. The results clearly show the effectiveness of the learned reduced-order $g$-MPC to capture and make use of its task-relevant hybrid modes to produce different strategies for accomplishing 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 cube reorientation failure. Here, some snapshots at key time steps of the $g$-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).

We observe that the reorientation failure are target dependent, meaning that the reorientation failures happen more frequently at some targets than at others. Also, we observe that changing the random seed for target distribution (38) also changes the failure target locations. This makes us believe that the orientation failure could be caused by the insufficient target sampling at such regions. In fact, the whole training process sampled from fewer than 200 target poses from (38). Some regions of the target space could be less sampled than other regions, leading to the learned reduced-order $g()$ not well representing those regions. We expect 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 observations, and note some limitations of the proposed method for solving contact-rich manipulation tasks.

1) LCS with Fewer Hybrid Modes: In the above three-finger manipulation tasks, we have used LCS models $g()$ in (8) with a fixed $\dim \lambda = 5$, which allows the representation of 32 hybrid modes. Results in Table IV and Table VII show that
the learned reduced-order LCS \( g() \) 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 \( g() \) with different \( \dim \lambda \) for the Cube Tuning manipulation task, under the same other conditions as in Table IV. The results are in Fig. 12.

Fig. 12: Performance with different \( \dim \lambda \), and the metric follows the same definitions as in Section VII-C2. Specifically, (a) is the total cost of a rollout with the \( g \)-MPC controller; (b) is the cube’s terminal orientation error; (c) is the on-policy ME; and (d) is running frequency of the reduced-order \( g \)-MPC.

The results in Fig. 12 show that on average, learning a LCS with fewer hybrid modes, such as \( \dim \lambda \leq 3 \), will lead to a degraded performance, although we occasionally encounter good performance with lower \( \dim \lambda \) on some ‘lucky’ random seeds. The previous Table IV has showed that the successful manipulation task needs around 14 hybrid modes in \( g() \). Thus, the successful manipulation on average requires at least \( \dim \lambda \geq 4 \). The results in Fig. 12 confirm this by showing the improved performances with \( \dim \lambda \geq 4 \). Another way to explain the results in Fig. 12 is that with a large \( \dim \lambda \), \( g() \) will have more expressive power to decrease the model prediction error, leading to better control performance, as suggested by Lemma 1.

Fig. 12 also shows that further increasing of \( \dim \lambda \), say \( \dim \lambda = 6 \), will not help the performance too much. Also, we notice that increasing \( \dim \lambda \) will slow the speed of the closed-loop \( g \)-MPC controller in Fig. 12d. In practice, the choice of \( \dim \lambda \) depends on tasks and systems, and one typically needs to find a \( \dim \lambda \) (through trial and error) by balancing its task performance and computational complexity.

2) 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 [45]. However, using ‘pieces’ to approximate smooth non-linearity is not the interest of this paper; instead, we focuses on using pieces to capture the hybrid structure (i.e., mode boundaries) of a non-linear hybrid system. But this observation motivates a future direction to extend LCS representation for non-linear complementarity models, which can be our future work.

Lastly, we want to point out that choosing multiple weights in a continuous cost function can be non-trivial for dexterous manipulation tasks. The possible solutions to overcome those challenges could include using inverse optimal control [46], [47] to learn cost functions from demonstrations [4].

VIII. CONCLUSIONS

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 model learning. The future work includes building the hardware and testing it on the real robotic manipulation system.

ACKNOWLEDGEMENTS

Toyota Research Institute provided funds to support this work.

APPENDIX A

PROOF OF LEMMA 2

Proof. Given any \( x_0 \sim p(x_0) \) and \( \beta \sim p(\beta) \), as \( u^0(x_0, \beta) \) is an optimal solution to (11), it satisfies the following first-order condition

\[
\nabla_u J_\beta(u^0, G(u^0, x_0)) = \begin{bmatrix} \frac{\partial J_\beta}{\partial u^0} \\ \frac{\partial J_\beta}{\partial G(u^0)} \end{bmatrix} = 0, \quad (50)
\]

where \( \frac{\partial J_\beta}{\partial u^0} \) denotes the partial gradient of \( J_\beta(u, G(u, x_0)) \) with respect to \( u \) evaluated at \( u^0(x_0, \beta) \), and similar notations
applies to $\frac{\partial J_\beta}{\partial u}$ and $\frac{\partial G}{\partial u}$ and below. Using (50), one has

$$\left\| \nabla_u J_\beta(u^g, F(u^g, x_0)) \right\|$$

$$= \left\| \nabla_u J_\beta(u^g, F(u^g, x_0)) - \nabla_u J_\beta(u^g, G(u^g, x_0)) \right\|$$

$$= \left\| \frac{\partial J_\beta(u, F)}{\partial u} - \frac{\partial J_\beta(u, G)}{\partial u} \right\| + \left\| \frac{\partial G}{\partial u} \right\| \leq \frac{M_j}{2} \left\| \frac{\partial F}{\partial u} - \frac{\partial G}{\partial u} \right\| + \frac{M_j}{2} L_1 \left\| F(u^g) - G(u^g) \right\|,$$

where the last inequality is due to the bound $\left\| \nabla_x J_\beta(u, x) \right\| \leq M_1$, $L_1$-Lipschitz continuity of $\nabla_x J_\beta(u, x)$, and the bound $\left\| \nabla_u G(u) \right\| \leq M_j$ given in Lemma 2.

Combining (51)-(53) and replacing $\left\| \frac{\partial F}{\partial u} - \frac{\partial G}{\partial u} \right\|$ compactly with $\left\| \nabla_u F(u) - \nabla_u G(u) \right\|$ we have (21) and (22) in Lemma 2. This completes the proof. \qed

REFERENCES


