Consensus Complementarity Control for Multi-Contact MPC

Alp Aydinoglu[1], Adam Wei[2], Wei-Cheng Huang[1], and Michael Posa[1]

***Abstract*—We propose a hybrid model predictive control algorithm, consensus complementarity control (C3), for systems that make and break contact with their environment. Many state-of-the-art controllers for tasks which require initiating contact with the environment, such as locomotion and manipulation, require *a priori* mode schedules or are too computationally complex to run at real-time rates. We present a method based on the alternating direction method of multipliers (ADMM) that is capable of high-speed reasoning over potential contact events. Via a consensus formulation, our approach enables parallelization of the contact scheduling problem. We validate our results on five numerical examples, including four high-dimensional frictional contact problems, and a physical experimentation on an underactuated multi-contact system. We further demonstrate the effectiveness of our method on a physical experiment accomplishing a high-dimensional, multi-contact manipulation task with a robot arm.**

***Index Terms*—Multi-Contact Control, Optimization and Optimal Control, Dexterous Manipulation, Contact Modeling**

I. Introduction

FOR many important tasks such as manipulation and locomotion, robots need to make and break contact with the environment. Even though such multi-contact systems are common, they are notoriously hard to control. The main challenge is finding policies and/or trajectories that explicitly consider the interaction of the robot with its environment in order to enable stable, robust motion. For a wide range of problems, it is computationally challenging to discover control policies and/or trajectories [1], [2], [3], [4], [5], [6], [7] and existing methods are not capable of running at real-time rates for complex problems.

Within robotics, model predictive control (MPC) [8], [9] has become a predominant tool for automatic control, most commonly via linearization of the governing dynamics leading to quadratic programs which can be solved efficiently (e.g. [10], [11] and many others). However, for multi-contact systems, the algorithm must also decide when to initiate or break contact, and when to stick or when to slide. These discrete choices are critical to tasks like dexterous manipulation, where modeling error or disturbances can easily perturb any pre-planned or nominal contact sequence, requiring real-time decision making to adjust and adapt. For example, for a robot arm to roll a spherical object across a surface (Figure 1), it must alternate between rolling and repositioning the hand while reacting to adversarial disturbances; for multi-object manipulation problems, the tasks becomes exponentially more difficult, as the robot must also decide which object or objects to touch at any given instant.

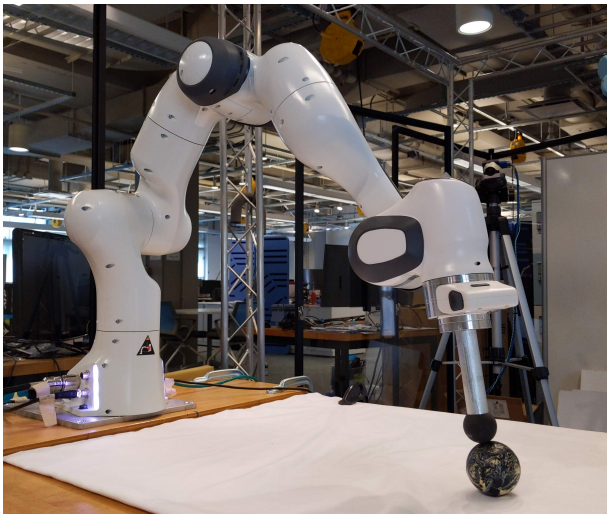


Fig. 1. Manipulating a rigid object (sphere) with a spherical end-effector attached to the Franka Emika Panda arm.

The result is a *hybrid* control problem. The resulting multi-modal dynamics are non-smooth and fundamentally cannot be accurately captured by linearization. MPC for these problems has classically focused on the use of abstract representations for hybrid systems, where discrete variables encode switching between modes, and continuous variables describe motion within modes. The result is often encoded via mixed-integer formulations, for instance as a mixed-integer quadratic program (MIQP) [12], [13]. However, due to the combinatoric complexity of reasoning over potential mode sequences, these methods struggle to scale to realistic robotics problems and real-time rates.

Rather than start with a generic hybrid representation, we instead focus on the dynamics of multi-contact robotics, paired with ADMM algorithms [14], [15], [16], [17], [18]. Specifically, we locally approximate multi-modal dynamics as a linear complementarity system (LCS) [19], and design an ADMM-based algorithm which can take advantage of the structure of the resulting multi-contact dynamics.

The primary contribution of this paper is an algorithm, consensus complementarity control (C3), for solving the hybrid

*This work was supported by the National Science Foundation under Grant No. EFRI-1935294 and CMMI-1830218.

[1]Alp Aydinoglu, Wei-Cheng Huang and Michael Posa are with the General Robotics, Automation, Sensing and Perception (GRASP) Laboratory, University of Pennsylvania, USA. [2]Adam Wei is with the Department of Electrical and Computer Engineering, University of Toronto, Canada. {alpayd, wchuang, posa}@seas.upenn.edu, adam.wei@mail.utoronto.ca (*Corresponding author: Alp Aydinoglu*)

MPC problem approximately for multi-contact systems. We exploit the distributed nature of ADMM and demonstrate that the hard part of the problem, reasoning about contact events, can be parallelized. This enables our algorithm to be fast, robust to disturbances and also minimizes the effect of control horizon on the run-time of the algorithm.

A preliminary version of this article was presented at the International Conference on Robotics and Automation [20]. In this work, extensions are as follows.

1) For the first time, we demonstrate that solving complex robotic manipulation tasks on hardware via real-time contact-implicit MPC is possible.
2) To achieve this, we show that multi-contact manipulation problems can be formulated as a linear complementarity system that is amenable to real-time MPC (via C3).
3) In addition, we propose a control framework that integrates C3 with a low-level impedance controller and a vision-based state estimation setup.
4) We introduce a novel convex projection that can improve the solve times for the C3 algorithm by more than 3 times as shown in Section VII-B. We also include theoretical analysis of this approximate projection, connecting its limiting behavior to linear complementarity problems.
5) The effectiveness of the method is demonstrated on two new numerical examples of high-dimensional manipulation tasks.

II. Related Work

Multi-contact robotics has drawn growing interest over the last decade [21], [22]. Early methods for both planning and control were focused on known contact sequences and hierarchical planners [23], [24]. Over the course of the last decade, contact-implicit planners were developed (originally for offline motion synthesis [1]) and this body of work is growing rapidly [4], [25]. These were first paired with controllers designed to track the planned mode (and limited to reasoning about that specific mode sequence) [26], [27].

Other approaches achieved real-time hybrid planning via utilizing task-specific model simplifications. Some of these methods specialize in legged systems [28], [29], [30] and focus on simplified dynamics models tailored for locomotion tasks, often times combined with gait-scheduling heuristics. However, these methods rely heavily on application-specific model simplifications and can not be applied across a wide range of robotics systems. Alternative methods rely on an offline phase to reduce the search space. This can lead to efficient real-time planning, but it requires a lengthy offline phase for each problem instance [31], [32], [33], [34]. Others approximate the dynamics via smooth contact models and modify the dynamics (e.g. diagonal approximation of Delassus operator) to enable high performance [35], [36], [37]. While such methods can perform well in simulation, they have not been validated on hardware performing dynamic tasks that require constantly making and breaking contact.

Building upon this growing body of literature, this paper presents an approach to real-time generic multi-contact MPC. In the preliminary version of this paper [20], and in the related work [38], there has been very recent progress in developing fast contact-implicit MPC algorithms. While contact-implicit trajectory optimization approaches use global models and can take minutes to solve [1], [4], these novel approaches focus on local hybrid models to significantly reduce the computational cost of optimal control. Le Cleac'h/Howell et al. [38] use a softened contact model in combination with a custom primal-dual interior-point solver to achieve real-time rates. They focus on tracking a pre-computed trajectory, which provides a nominal mode sequence to track (although their algorithm can adapt this sequence online). In contrast, our approach is demonstrated to be fully capable of mode sequence synthesis without the need for a nominal trajectory or any other offline computations.

III. Background

We first introduce some of the definitions and notation used throughout this work. The function $\mathcal{I}_{\mathcal{A}}$ is the $0-\infty$ indicator function for an arbitrary set $\mathcal{A}$ such that $\mathcal{I}_{\mathcal{A}}(z)=0$ if $z \in \mathcal{A}$ and $\mathcal{I}_{\mathcal{A}}(z)=\infty$ if $z \notin \mathcal{A}$. $\mathbb{N}_0$ denotes natural numbers with zero. For a positive semi-definite matrix Q, $Q^{1/2}$ denotes a matrix P such that $PP=P^TP=Q$. The notation $\mathbf{blkdiag}(A_0,\dots,A_N)$ denotes a block diagonal matrix with entries in the given order (from top left (A_0) to bottom right (A_N)). The notation $\mathbf{diag}(a_0,\dots,a_N)$ is used if entries a_i are scalars. For two vectors $a \in \mathbb{R}^m$ and $b \in \mathbb{R}^m$, $0 \le a \perp b \ge 0$ denotes $a \ge 0$, $b \ge 0$, $a^Tb=0$.

A. Linear Complementarity Problem

In this work, we utilize complementarity problems to represent the contact forces. These models are common in modeling multi-contact robotics problems [39] and can also be efficiently learned from data [40]. The theory of linear complementarity problems (LCP) is well established [41].

Definition 1: Given a vector $q \in \mathbb{R}^m$, and a matrix $F \in \mathbb{R}^{m\times m}$, the LCP$(q,F)$ describes the following mathematical program:

$$\begin{aligned} \text{find} \quad & \lambda \in \mathbb{R}^m \\ \text{subject to} \quad & y = F\lambda + q, \\ & 0 \le \lambda \perp y \ge 0. \end{aligned}$$

Here, the vector λ typically represents the contact forces and slack variables [39], y represents the gap function and orthogonality constraint embeds the hybrid structure.

B. Linear Complementarity System

We use linear complementarity systems (LCS) as local representations of multi-contact systems [42]. An LCS [19] is a differential/difference equation coupled with a variable that is the solution of an LCP.

Definition 2: A linear complementarity system describes the trajectories $(x_k)_{k\in\mathbb{N}_0}$ and $(\lambda_k)_{k\in\mathbb{N}_0}$ for an input sequence $(u_k)_{k\in\mathbb{N}_0}$ starting from x_0 such that

$$\begin{aligned} & x_{k+1} = Ax_k + Bu_k + D\lambda_k + d, \\ & 0 \le \lambda_k \perp Ex_k + F\lambda_k + Hu_k + c \ge 0, \end{aligned} \tag{1}$$

where $x_k \in \mathbb{R}^{n_x}$, $\lambda_k \in \mathbb{R}^{n_\lambda}$, $u_k \in \mathbb{R}^{n_u}$, $A \in \mathbb{R}^{n_x \times n_x}$, $B \in \mathbb{R}^{n_x \times n_u}$, $D \in \mathbb{R}^{n_x \times n_\lambda}$, $d \in \mathbb{R}^{n_x}$, $E \in \mathbb{R}^{n_\lambda \times n_x}$, $F \in \mathbb{R}^{n_\lambda \times n_\lambda}$, $H \in \mathbb{R}^{n_\lambda \times n_u}$ and $c \in \mathbb{R}^{n_\lambda}$. Equation (1) is often called as generalized LCS [43] or inhomogeneous LCS [44] due to existence of the vectors c and d, but we use LCS for brevity.

Vector x_k represents the state and often consists of the generalized positions q_k and velocities v_k. For a given k, x_k and u_k, the corresponding complementarity variable λ_k can be found by solving LCP$(Ex_k + Hu_k + c, F)$ (see Definition 1). Similarly, x_{k+1} can be computed using the first equation in (1) when x_k, u_k and λ_k are known.

For the frictional systems we study, LCPs can lack solutions or have multiple solutions. Here, we assume that the mapping $(x_k, u_k) \mapsto x_{k+1}$ is unique even though λ_k is not necessarily unique [42]. This is a common assumption in contact-implicit trajectory optimization and is an outcome from relaxed simulation approaches (MuJoCo [45], Dojo [46] etc.). This is a necessary assumption for MPC to not immediately become a robust control problem reasoning over every possible outcome.

IV. Multi-Contact Dynamics and Local Approximations

Before presenting the core algorithmic contributions of this paper, we describe how multi-contact robot dynamics can be approximated by an LCS. Rigid-body systems with contact can be modeled by the manipulator equation:

$$M(q)\dot{v} + C(q, v) = Bu + J(q)^T \lambda, \tag{2}$$

where q is the generalized positions vector, v is the generalized velocities, λ represents the contact forces (potentially includes slack variables to capture frictional contact [39]), $M(q)$ is the inertia matrix, $C(q, v)$ represents the combined Coriolis, centrifugal and gravitational terms, B maps control inputs to joint coordinates and $J(q)$ is the projection matrix that is typically the contact Jacobian (potentially padded with zeros for the corresponding slack variables).

One approach to represent λ is via the complementarity framework:

$$0 \leq \lambda \perp \psi(q, v, u, \lambda) \geq 0. \tag{3}$$

Here ψ relates the position q, velocity v and input u with the vector λ that includes the contact forces ([39], [47] for more details). For compactness, we denote a nonlinear multi-contact model as $\mathcal{M}$ where $\mathcal{M} = \{M, C, B, J, \psi\}$.

In this work, we are interested in solving the following optimal control problem with model $\mathcal{M}$ (we consider discrete-time formulations as in [1]):

$$\min_{q_k, v_k, u_k, \lambda_k} \quad \sum_k g_c(q_k, v_k, u_k) \tag{4}$$

$$\text{s.t.} \quad (q_{k+1}, v_{k+1}, q_k, v_k, u_k, \lambda_k) \text{ respects } \mathcal{M} \tag{5}$$

where g_c is the cost function (often quadratic). Now, we introduce two different approaches to transcribe the dynamics.

A. Stewart-Trinkle Formulation

In most of the examples in this paper, we transcribe (5) using the following implicit time-stepping scheme based on Stewart and Trinkle's seminal work [39] due to its intuitive, simple form:

$$\begin{aligned} q_{k+1} &= q_k + \Delta t v_{k+1}, \\ v_{k+1} &= v_k + \Delta t M^{-1}(q_k)\Big(C(q_k, v_k) + Bu_k \\ &\qquad + J_n(q_k)^T \lambda_k^n + J_t(q_k)^T \lambda_k^t\Big), \end{aligned} \tag{6}$$

where Δt is the discretization time-step, $\lambda_k^n \in \mathbb{R}^{n_c}$ represents the normal forces, $\lambda_k^t \in \mathbb{R}^{n_c n_e}$ represents the tangential (friction) forces, n_c is the number of rigid body pairs that can interact, n_e represents the number of edges of the polyhedral approximation of the friction cone, and J_n, J_t are contact Jacobians for normal and tangential directions. The forces λ_k^n and λ_k^t can be described via the following complementarity problem [39]:

$$\begin{aligned} &0 \leq \gamma_k \perp \mu \lambda_k^n - E_t \lambda_k^t \geq 0, \\ &0 \leq \lambda_k^n \perp \phi(q_k) + J_n(q_k)(q_{k+1} - q_k) \geq 0, \\ &0 \leq \lambda_k^t \perp E_t^T \gamma_k + J_t(q_k) v_{k+1} \geq 0, \end{aligned} \tag{7}$$

where $\gamma_k \in \mathbb{R}^{n_c}$ is a slack variable, $\mu = \mathbf{diag}(\mu_1, \ldots, \mu_{n_c})$ represents the coefficient of friction between rigid body pairs, $E_t = \mathbf{blkdiag}(e, \ldots, e)$ with $e = [1, \ldots, 1] \in \mathbb{R}^{1 \times n_e}$ and ϕ represents the distance between rigid body pairs.

The dynamics formulation in (6) and (7), as a nonlinear complementarity problem, has been employed for trajectory optimization of multi-contact robotics [1], though such methods have been limited to offline motion planning due to the inherent complexity in the nonlinear model. To reduce the complexity of the model, it is possible to locally approximate $\mathcal{M}$ with an LCS. Even though an LCS approximation is slightly less complex than the nonlinear model, it still captures the multi-modal nature of the problem and enables one to make decisions such as making or breaking contact (as demonstrated in Sections VI and VII). Hence we focus on LCS approximations as they lead to more tractable optimal control problems than their nonlinear counterparts but note that these are still hybrid optimal control problems that are difficult to solve (Section V).

Now, we present an approach to obtain an LCS approximation given the complementarity system model (6)-(7). The use of LCS approximations is not new to this work [42], [38], [13], though we note there is no single canonical LCS formulation. Here, we present an approach inspired by Stewart and Trinkle [39] in detail. For a given state $(x^*)^T = [(q^*)^T, (v^*)^T]$ and input u^*, we approximate (6) as:

$$\begin{aligned} q_{k+1} &= q_k + \Delta t v_{k+1}, \\ v_{k+1} &= v_k + \Delta t (J_f \begin{bmatrix} q_k \\ v_k \\ u_k \end{bmatrix} + D_1 \lambda_k^n + D_2 \lambda_k^t + d_v). \end{aligned} \tag{8}$$

Here $J_f = J_f(q^*, v^*, u^*)$ is the Jacobian of $f(q, v, u) = M^{-1}(q)Bu - M^{-1}(q)C(q, v)$ evaluated at (q^*, v^*, u^*), $d_v =$

$f(q^*, v^*, u^*) - J_f \begin{bmatrix}(q^*)^T & (v^*)^T & (u^*)^T\end{bmatrix}^T$ is a constant vector, $D_1 = M^{-1}(q^*)J_n(q^*)^T$ and $D_2 = M(q^*)^{-1}J_t(q^*)^T$ represent the effect of contact forces. Similarly, we approximate (7) with the following LCP:

$$\begin{aligned} &0 \leq \gamma_k \perp \mu\lambda_k^n - E_t\lambda_k^t \geq 0, \\ &0 \leq \lambda_k^n \perp \phi(q^*) + J_n(q^*)q_k + \Delta t J_n(q^*)v_{k+1} \\ &\qquad\qquad - J_n(q^*)q^* \geq 0, \\ &0 \leq \lambda_k^t \perp E_t^T\gamma_k + J_t(q^*)v_{k+1} \geq 0. \end{aligned} \tag{9}$$

Observe that (8) and (9) can be written in the LCS form (1) where $x_k^T = [q_k^T, v_k^T]$ and $\lambda_k^T = [\gamma_k^T, (\lambda_k^n)^T, (\lambda_k^t)^T]$.

B. Anitescu Formulation

In Section V-B, we propose a novel, approximate projection method that requires convexity of the contact model (i.e. $F = F(x_k)$ as in (1) is positive semi-definite for any x_k) but the Stewart-Trinkle formulation does not satisfy this assumption. Hence, we alternatively employ a formulation from Anitescu [48], which does guarantee that $F \succeq 0$:

$$\begin{aligned} q_{k+1} &= q_k + \Delta t v_{k+1}, \\ v_{k+1} &= v_k + M^{-1}(q_k)\Big(\Delta t B u_k - \Delta t C(q_k, v_k) \\ &\qquad + J_c(q_k)^T\lambda_k\Big), \end{aligned} \tag{10}$$

where the contact Jacobian is defined as $J_c(q_k) = E_t^T J_n(q_k) + \mu J_t(q_k)$. Via the complementarity equations, λ_k is described as:

$$\begin{aligned} 0 \leq \lambda_k \perp \frac{E_t^T\phi(q_k)}{\Delta t} + \frac{1}{\Delta t}E_t^T J_n(q_k)(q_{k+1} - q_k) \\ + \mu J_t(q_k)v_{k+1} \geq 0. \end{aligned} \tag{11}$$

Similarly (for the Anitescu formulation), given state $(x^*)^T = [(q^*)^T, (v^*)^T]$ and input u^*, we can approximate (10) as:

$$\begin{aligned} q_{k+1} &= q_k + \Delta t v_{k+1}, \\ v_{k+1} &= v_k + \Delta t J_f \begin{bmatrix} q_k \\ v_k \\ u_k \end{bmatrix} + D\lambda_k + \Delta t d_v. \end{aligned} \tag{12}$$

where J_f, d_v are same as in (8) and $D = M^{-1}(q^*)J_c(q^*)^T$. The complementarity part (11) is:

$$\begin{aligned} 0 \leq \lambda_k \perp \frac{1}{\Delta t}E_t^T\Big(\phi(q^*) + J_n(q^*)q_k - J_n(q^*)q^*\Big) \\ + J_c(q^*)v_{k+1} \geq 0. \end{aligned} \tag{13}$$

Similar to the LCS approximation of the Stewart-Trinkle formulation, (12) and (13) can be written in the LCS form (1) where $x_k^T = [q_k^T, v_k^T]$. While a full comparison of these methods is outside the scope of this paper, we note that convexity in the Anitescu formulation is not without cost: this version does introduce some modeling artifacts, particularly during high-speed sliding motion.

It is important to note that to obtain these LCS models, we approximate the differentiable terms (of the model $\mathcal{M}$), which arise form standard Lagrangian dynamics, by linearizing with respect to (q, v, u) about (q^*, v^*, u^*) but leave the multi-modal structure intact. Moving forward, we focus on LCS models of the form (1) as it is a general form. We do not include time-varying LCS (where matrices such as A in (1) depend on k) for ease of notation but note that this work can easily be extended to the time-varying setting (and code we provide can deal with time-varying LCS). With a slight abuse of notation, we denote the models as $\mathcal{L}_{\Delta t}(x, u)$ where dependence on a given state x^* and input u^* (e.g. as in (8), (9)) is suppressed. Furthermore we use the notation $(x_{k+1}, \lambda_k) = \mathcal{L}_{\Delta t}(x_k, u_k)$ to denote the state of the LCS system after one time step (Δt time later). Here, λ_k is the solution of LCP$(Ex_k + Hu_k + c, F)$ and $x_{k+1} = Ax_k + Bu_k + D\lambda_k + d$.

V. Model Predictive Control of Multi-Contact Systems

As we consider LCS models, we focus on the following mathematical optimization problem:

$$\begin{aligned} \min_{x_k, \lambda_k, u_k} \quad & \sum_{k=0}^{N-1}(x_k^T Q_k x_k + u_k^T R_k u_k) + x_N^T Q_N x_N \\ \text{s.t.} \quad & x_{k+1} = Ax_k + Bu_k + D\lambda_k + d, \\ & Ex_k + F\lambda_k + Hu_k + c \geq 0, \\ & \lambda_k \geq 0, \\ & \lambda_k^T(Ex_k + F\lambda_k + Hu_k + c) = 0, \\ & (\boldsymbol{x}, \boldsymbol{\lambda}, \boldsymbol{u}) \in \mathcal{C}, \\ & \text{for } k = 0, \ldots, N-1, \text{given } x_0, \end{aligned} \tag{14}$$

where N is the planning horizon, Q_k, Q_N are positive semidefinite matrices, R_k are positive definite matrices and $\mathcal{C}$ is a convex set (e.g. input bounds, safety constraints, or goal conditions) and $\boldsymbol{x}^T = [x_1^T, \ldots, x_N^T]$, $\boldsymbol{\lambda}^T = [\lambda_0^T, \lambda_1^T, \ldots, \lambda_{N-1}^T]$, $\boldsymbol{u}^T = [u_0^T, u_1^T, \ldots, u_{N-1}^T]$.

Given x_0, one solves the optimization (14) and applies u_0 to the plant and repeats the process in every time step in a receding horizon manner.

A. Mixed Integer Formulation

One straightforward, but computationally expensive, approach to solving (14) is via a mixed integer formulation which exchanges the non-convex orthogonality constraints for binary variables:

$$\begin{aligned} \min_{x_k, \lambda_k, u_k, s_k} \quad & \sum_{k=0}^{N-1}(x_k^T Q_k x_k + u_k^T R_k u_k) + x_N^T Q_N x_N \\ \text{s.t.} \quad & x_{k+1} = Ax_k + Bu_k + D\lambda_k + d, \\ & Ms_k \geq Ex_k + F\lambda_k + Hu_k + c \geq 0, \\ & M(\mathbf{1} - s_k) \geq \lambda_k \geq 0, \\ & (\boldsymbol{x}, \boldsymbol{\lambda}, \boldsymbol{u}) \in \mathcal{C}, s_k \in \{0, 1\}^{n_\lambda}, \\ & \text{for } k = 0, \ldots, N-1, \text{given } x_0, \end{aligned} \tag{15}$$

where $\mathbf{1}$ is a vector of ones, M is a scalar used for the big M method and s_k are the binary variables. This approach requires solving 2^{Nn_λ} quadratic programs as a worst case. This method is popular because, in practice, it is much faster than this worst case analysis. Even still, for the multi-contact problems

explored in this work, directly solving (15) via state-of-the-art commercial solvers remains too slow for real-time control. Methods that learn the MIQP problem offline are promising, but this line of work requires large-scale training on every problem instance [49], [50].

B. Consensus Complementarity Control (C3)

Utilizing a method based on ADMM, we will solve (14) more quickly than with mixed integer formulations. Since the problem we are addressing is non-convex, our method is not guaranteed to find the global solution or converge unlike MIQP-based approaches, but is significantly faster.

Towards this direction, we first transform problem (15) into a consensus formulation. Then, we apply ADMM to solve the problem written in consensus form (16) by transforming it into a set of three distinct operations. Finally, we propose four different approaches to solve one of those operations (the projection operation).

1) Consensus Formulation: First, we rewrite (14), equivalently, in the consensus form [17] where we create copies (named δ_k) of variables $z_k^T = [x_k^T, \lambda_k^T, u_k^T]$ and move the constraints into the objective function using $0-\infty$ indicator functions:

$$\begin{aligned} \min_{z} \quad & c(z) + \mathcal{I}_{\mathcal{D}}(z) + \mathcal{I}_{\mathcal{C}}(z) + \sum_{k=0}^{N-1} \mathcal{I}_{\mathcal{H}_k}(\delta_k) \\ \text{s.t.} \quad & z_k = \delta_k, \ \forall k, \end{aligned} \tag{16}$$

where $z^T = [z_0^T, z_1^T, \ldots, z_{N-1}^T]$. Note that $\delta^T = [\delta_0^T, \delta_1^T, \ldots, \delta_{N-1}^T]$ is a copy of z^T and equality constraints can also be written as $z = \delta$. $c(z)$ is the cost function[1] in (14). The set $\mathcal{D}$ includes the dynamics constraints:

$$\cap_{k=0}^{N-2}\{z : x_{k+1} = Ax_k + Bu_k + D\lambda_k + d\},$$

and the sets $\mathcal{H}_k$ represent the LCP (contact) constraints:

$$\begin{aligned} \mathcal{H}_k = \{(x_k, \lambda_k, u_k) : & Ex_k + F\lambda_k + Hu_k + c \geq 0, \\ & \lambda_k \geq 0, \lambda_k^T(Ex_k + F\lambda_k + Hu_k + c) = 0\}. \end{aligned}$$

Note that we leverage the time-dependent structure in the complementarity constraints to separate δ_k's so that each set $\mathcal{H}_k$ depends only on δ_k, and not variables corresponding to any other timestep.

2) ADMM Steps for the Consensus Formulation: In this part, we discuss how to solve the problem in consensus formulation (16) via ADMM. The general augmented Lagrangian ([14], Section 3.4.2.) for the problem in consensus form (16) is (Appendix for details):

$$\begin{aligned} \mathcal{L}_\rho(z, \delta, w) = & \, c(z) + \mathcal{I}_{\mathcal{D}}(z) + \mathcal{I}_{\mathcal{C}}(z) \\ & + \sum_{k=0}^{N-1} \big(\mathcal{I}_{\mathcal{H}_k}(\delta_k) + \rho(r_k^T G_k r_k - w_k^T G_k w_k)\big), \end{aligned} \tag{17}$$

[1]The cost function has the form $c(z) = \sum_{k=0}^{N-1}(x_k^T Q_k x_k + u_k^T R_k u_k) + ||Q_N^{1/2}(Ax_{N-1} + Bu_{N-1} + D\lambda_{N-1} + d)||_2^2$.

where $\rho > 0$ is the penalty parameter, $w^T = [w_0^T, w_1^T, \ldots, w_{N-1}^T]$, w_k are scaled dual variables, $r_k = z_k - \delta_k + w_k$, G_k is a positive definite matrix. Observe that the standard augmented Lagrangian [17] is recovered for $G_k = I$ for all k.

In order to solve (16), we apply the ADMM algorithm, consisting of the following operations (Appendix for details):

$$z^{i+1} = \text{argmin}_z \mathcal{L}_\rho(z, \delta^i, w^i), \tag{18}$$

$$\delta_k^{i+1} = \text{argmin}_{\delta_k} \mathcal{L}_\rho^k(z_k^{i+1}, \delta_k, w_k^i), \ \forall k, \tag{19}$$

$$w_k^{i+1} = w_k^i + z_k^{i+1} - \delta_k^{i+1}, \ \forall k, \tag{20}$$

where $\mathcal{L}_\rho^k(z_k, \delta_k, w_k) = \mathcal{I}_{\mathcal{H}_k}(\delta_k) + \rho(r_k^T G_k r_k - w_k^T G_k w_k)$. Here, (18) requires solving a quadratic program, (19) is a projection onto the LCP constraints and (20) is a dual variable update. Next, we analyze these operations in the given order.

a) Quadratic Step: Equation (18) can be represented by the convex quadratic program

$$\begin{aligned} \min_{z} \quad & c(z) + \sum_{k=0}^{N-1} (z_k - \delta_k^i + w_k^i)^T \rho G_k (z_k - \delta_k^i + w_k^i) \\ \text{s.t.} \quad & z \in \mathcal{D} \cap \mathcal{C}. \end{aligned} \tag{21}$$

The linear dynamics constraints are captured by the set $\mathcal{D}$, and the convex inequality constraints on states, inputs, contact forces are captured by $\mathcal{C}$. The complementarity constraints do not explicitly appear, but their influence is found, iteratively, through the variables δ_k^i.

The QP in (21) can be solved quickly via off-the-shelf solvers and is analogous to solving the MPC problem for a linear system without contact.

b) Projection Step: This step requires projecting onto the LCP constraints $\mathcal{H}_k$ and is the most challenging part of the problem. (19) can be represented with the following quadratic program with a non-convex constraint:

$$\begin{aligned} \min_{\delta_k} \quad & (\delta_k - (z_k^{i+1} + w_k^i))^T \rho G_k (\delta_k - (z_k^{i+1} + w_k^i)) \\ \text{s.t.} \quad & \delta_k \in \mathcal{H}_k, \end{aligned} \tag{22}$$

where $\delta_k^T = [(\delta_k^x)^T, (\delta_k^\lambda)^T, (\delta_k^u)^T]$. Within this framework, we present four alternative algorithms for this projection stage. Three are approximate projections, common for minimization problems over non-convex sets [51].

3) Projection Operation: As stated, we propose four different methods for the projection step. The first three apply to any LCS model, while the fourth requires $F \succeq 0$, and thus is limited to the Anitescu formulation of dynamics.

a) MIQP Projection: The projection can be calculated by exactly formulating (22) as a small-scale MIQP

$$\begin{aligned} \min_{\delta_k, s_k} \quad & (\delta_k - (z_k^{i+1} + w_k^i))^T U (\delta_k - (z_k^{i+1} + w_k^i)) \\ \text{s.t.} \quad & Ms_k \geq E\delta_k^x + F\delta_k^\lambda + H\delta_k^u + c \geq 0, \\ & M(\mathbf{1} - s_k) \geq \delta_k^\lambda \geq 0, \\ & s_k \in \{0,1\}^{n_\lambda}, \end{aligned} \tag{23}$$

where U is a positive semi-definite matrix. For $U = \rho G_k$, one recovers the problem in (22); however, in our experience, we found significantly improved performance using alternate, but

fixed, choices for U. Observe that while (23) is non-convex, it is written only in terms of variables corresponding to a single time step k. While the original MIQP formulation in (15) has Nn_λ binary variables, here we have N independent problems, each with n_λ binary variables. This decoupling leads to dramatically improved performance (worst-case $N2^{n_\lambda}$ vs 2^{Nn_λ}).

b) LCP Projection: In cases where (23) cannot be solved quickly enough, we propose three approximate solutions with faster run-time. Consider the limiting case where U has no penalty on the force elements. Here, (23) can be solved with optimal objective value of 0 by setting $\delta_k^x = z_k^{(i+1),x} + w_k^{(i),x}$ and $\delta_k^u = z_k^{(i+1),u} + w_k^{(i),u}$. Then, δ_k^λ can be found by solving $\text{LCP}(E\delta_k^x + H\delta_k^u + c, F)$. We note that this projection is different than shooting based methods since we are simulating the $z_k^{(i+1),x,u} + w_k^{(i),x,u}$ instead of $z_k^{(i+1),x,u}$.

c) ADMM Projection: We note that (23) has an equivalent quadratically constrained quadratic program (QCQP) representation, where prior work has solved problems of this form using ADMM ([17], Section 4.4). This approach relies on the fact that QCQPs with a single constraint are solvable in polynomial time via the bisection method ([17], Appendix B). Performing the projection step via this method leads to nested ADMM algorithms, but this formulation can produce faster solutions than MIQP solvers without guarantees that it produces a feasible or optimal value. Note that this formulation fared poorly for high dimensional projections or when applied directly to the original problem (14), rarely satisfying the complementarity constraints.

d) Convex Projection: Here, we introduce a novel approximate projection that is computationally efficient, as it only requires solving a single quadratic program:

$$\begin{aligned} \min_{\delta_k} \quad & \frac{1}{2}||\delta_k^x - x_d||_{Q_x} + \frac{1}{2}||\delta_k^u - u_d||_{Q_u} \\ & + \frac{\alpha}{2}||\delta_k^\lambda - \lambda_d||_F + \frac{1-\alpha}{2}||\delta_k^\lambda||_F \\ \text{s.t.} \quad & E\delta_k^x + F\delta_k^\lambda + H\delta_k^u + c \geq 0, \\ & \delta_k^\lambda \geq 0, \end{aligned} \tag{24}$$

where $x_d = z_k^{(i+1),x} + w_k^{(i),x}$, $u_d = z_k^{(i+1),u} + w_k^{(i),u}$, $\lambda_d = z_k^{(i+1),\lambda} + w_k^{(i),\lambda}$ are the desired x, u, λ targets for the projection step respectively, and $\alpha \in [0, 1]$ is a hyperparameter. The cost term in (24) mimics that of the true projection, but blends in terms which approximate complementarity error. We note the role of F (from (1)), to which we add a regularizing term ϵI to transform the positive semi-definite F from the Anitescu model into a strictly positive definite matrix.

One benefit of this approximate projection is that, in the limit $\alpha \to 0$, any error in the complementarity constraints also approaches zero, and so α can be seen as a hyperparameter which regulates the complementarity violation.

Lemma 3: The complementarity error is bounded linearly with α, i.e. $(\delta_k^\lambda)^T(E\delta_k^x + F\delta_k^\lambda + H\delta_k^u + c) \to 0$ as $\alpha \to 0$.

Proof: In Appendix. ∎

Empirically, solving the full MIQP (23) typically is the best at exploring a wide range of modes, while the LCP and convex projection are usually the fastest (depending on matrix F). The ADMM projection can be viewed as a middle ground. We underline that both MIQP and LCP projections produce feasible solutions whereas ADMM and the convex projection have no such guarantee. For frictional contact problems (Sections VI and VII), we observed that the MIQP and convex projections performed better than the others.

Algorithm 1 Consensus Complementarity Control (C3)

Require: $\theta = \{Q_k, R_k, Q_N, \delta_k^0, w_k^0, G_k, s, \rho, \rho_s, N\}$, $\mathcal{L}_{\Delta t}$, x_0
Initialization : $i = 1$
while $i \leq s$ **do**
  Compute z^{i+1} via (21)
  Compute δ_k^{i+1} via (23), $\forall k$
  $w_k^{i+1} \leftarrow w_k^i + z_k^{i+1} - \delta_k^{i+1}, \forall k$
  $\rho \leftarrow \rho_s \rho$
  $w_k^{i+1} \leftarrow w_k^{i+1}/\rho_s, \forall k$
  $i \leftarrow i + 1$
end while
return u_0 from z^{s+1}

After discussing each of the individual steps, we present the full C3 algorithm (Algorithm 1). Here θ represents the C3 parameters. Both δ_k^0 and w_k^0 are usually initialized as zero vectors. G_k are positive definite matrices, $s > 0$ is the number of ADMM steps, $\rho_s > 0$ is the scaling parameter for ρ, and N is the number of planning steps. Furthermore $\mathcal{L}_{\Delta t}$ is the LCS model (Section IV) and x_0 is an initial state. Given this information, the algorithm returns u_0 as in standard model-predictive control frameworks.

VI. Illustrative Examples

In this section, we demonstrate the effectiveness of C3 presenting multiple simulation results and an hardware experiment with an under-actuated multi-contact system. Concrete steps of method used throughout this section is shown in Algorithm 2. Given a multi-contact model $\mathcal{M}$, we obtain an LCS model $\mathcal{L}_{\Delta t}$ around the estimated state $\hat{x}$ and nominal input $\hat{u}$. Then, we run Algorithm 1 to compute u_0 and that is directly applied to the system.

For these results, OSQP [52] is used to solve quadratic programs and Gurobi [53] is used for mixed integer programs. PATH [54] and Lemke's algorihm have been used to solve LCPs. SI units (meter, kilogram, second) are used. The experiments are done on a desktop computer with the processor Intel *i7-11800H* and *16GB RAM*. Reported run-times include all steps in the algorithm. The code for all examples is available[2], and we optimized the code implementation leading to algorithm speeding up especially for the finger gaiting and pivoting examples compared to the preliminary conference version. We have added the specific systems matrices for all the examples[3]. Experiments are also shown in the supplementary video[4].

[2]https://github.com/AlpAydinoglu/coptimal
[3]https://github.com/AlpAydinoglu/coptimal/blob/main/system_matrices.txt
[4]https://youtu.be/L57Jz3dPwO8

Algorithm 2

Require: $\mathcal{M}, \theta, \Delta t, \hat{u}$

1: Estimate state as $\hat{x}$
2: Obtain LCS model $\mathcal{L}_{\Delta t}$ around $(\hat{x}, \hat{u})$ using $\mathcal{M}$
3: Run Algorithm 1 with $\mathcal{L}_{\Delta t}$, θ, $\hat{x}$ and obtain u_0

A. Finger Gaiting

In this subsection, we want to show that our algorithm is capable of mode exploration. Our goal is to lift a rigid object upwards using four fingers. The setup for this problem is illustrated in Figure 2. The red circles indicate where the grippers interact the object and we assume that the grippers are always near the surface of the object and the force they apply on the object can be controlled. This force affects the friction between the object and grippers. Since the grippers never leave the surface, we assume that there is no rotation. The goal of this task is to lift the object vertically, while the fingers are constrained to stay close to their original locations (constraints shown in yellow). This task, therefore, requires finger gaiting to achieve large vertical motion of the object.

We use the formulation in [39] for modeling the system and denote the positions of the grippers as $g^{(1)}$, $g^{(2)}$ respectively and position of the object as o. We choose $g = 9.81$ as gravitational acceleration and $\mu = 1$ is the coefficient of friction for both grippers. The system has six states ($n_x = 6$), including the position and velocity of the object and two fingers. Also, there are six complementarity variables ($n_\lambda = 6$) where each contact is represented by 3 complementarity variables. There are four inputs ($n_u = 4$), including the normal forces for finger contacts and the acceleration of the fingers.

We design a controller based on Algorithm 1 where $G_k = I$, $N = 10$, $\Delta t = 0.1$ for $\mathcal{L}_{\Delta t}$. The controller uses the MIQP projection method since the other two projection methods failed to move the object close to the desired target. For this example, we use $s = 10$ and $\rho_s = 1.2$ and C3 ran at 30.3 Hz, an approximately 2.5x speedup from 10 Hz [20]. We also enforce limits:

$$1 \leq g^{(1)} \leq 3, \ \forall k,$$
$$3 \leq g^{(2)} \leq 5, \ \forall k.$$

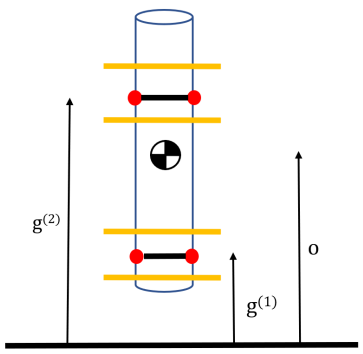


Fig. 2. Lifting an object using two grippers indicated by red circles. The limits where the grippers should not cross are indicated by yellow lines.

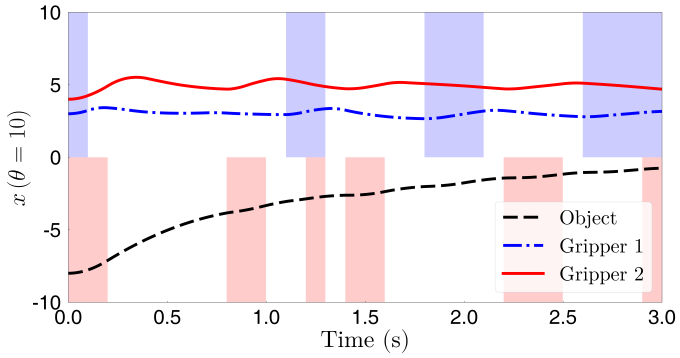


Fig. 3. Finger gaiting with $s = 10$. The upper blue shading implies that gripper 1 is applying normal force to the object whereas the lower red shading implies that gripper 2 is applying normal force.

TABLE I
SOLVE TIME AND OPTIMALITY COMPARISON FOR FINGER GAITING

Item	Formulation	MPC Horizon N			
		10	20	30	50
Solve Time (s)	**MIQP**	0.112	0.294	0.791	2.315
	C3	0.025	0.051	0.067	0.095
Accumulated Cost ($\times 10^7$)	**MIQP**	3.467	3.454	3.451	3.452
	C3	4.979	4.248	4.101	9.552

We performed 100 trials starting from different initial conditions where $o(0) \sim U[-6, -8]$, $g^{(1)}(0) \sim U[2, 3]$, $g^{(2)}(0) \sim U[3, 4]$ are uniformly distributed and both the grippers and the object have zero initial speed. The controller managed to lift the object in all cases. We present a specific example in Figure 3 where the grippers first throw the object into the air and then catch it followed by some finger gaiting.

In order to show the effectiveness of our approach, we compare[5] the computation speed of C3 and the MIQP formulation (as in (15)) for different MPC horizons N. For the comparison experiments, we pick $s = 5, \rho = 1, \rho_s = 1.01$, and present the computation times (in seconds) in Table I. We also include the solution optimality comparison of those two methods where the optimality is compared by calculating the accumulated cost for a given time (2 seconds for this example). The tabulated results show that C3 is substantially faster than a baseline MIQP solver, particularly as the horizon increases, and that this speed comes at only a modest increase in accumulated cost.

B. Pivoting

Here, we want to demonstrate that our algorithm is capable of making decisions about stick-slip transitions as well as making/breaking contact. We consider pivoting a rigid object that can make and break contact with the ground inspired by Hogan et. al [55]. Two fingers (indicated via blue) interact with the object as in Figure 4. The goal is to balance the rigid-object at the midpoint.

The positions of the fingers with respect to the object are described via f_1, f_2 respectively. The normal force that the

[5]The tests were run in a Python script for both methods.

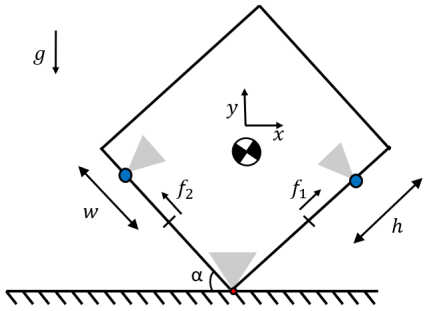


Fig. 4. Pivoting a rigid object with two fingers (blue). The object can make and break contact with the ground and gray areas represent the friction cones.

fingers exert onto the box can be controlled. The center of mass position is denoted by x and y respectively, α denotes the angle with the ground and $w = 1$, $h = 1$ are the dimensions of the object. The coefficient of friction for the fingers are $\mu_1 = \mu_2 = 0.1$, and the coefficient of friction with the ground is $\mu_3 = 0.1$. We take the gravitational acceleration as $g = 9.81$ and mass of the object as $m = 1$. We model the system using an implicit time-stepping scheme [39]. The system has three contacts where the finger contacts are represented by 3 complementarity variables each (1 slack variable and 2 frictional force variables), and the ground contact is modeled via 4 complementarity variables (1 normal force variable, 1 slack variable and 2 frictional force variables). There are ten states ($n_x = 10$), including the pose and velocity of the cube and the fingers. Also, the system has ten complementarity variables ($n_\lambda = 10$), and 4 inputs ($n_u = 4$) that consist of the normal forces of the finger contacts and the acceleration of the fingers. We note that this system has $3 \times 3 \times 4 = 36$ hybrid modes.

For this example, as the LCS-representation is only an approximation, we compute a new local LCS approximation at every time step k. We pick $s = 5$, $N = 10$, $\rho_s = 1.1$, $\Delta t = 0.01$ for $\mathcal{L}_{\Delta t}$ and use the local LCS approximation given at time-step k while planning. With improvements, C3 runs at around 43.4 Hz, an approximately 4x increase from the previously reported performance of 16 Hz [20].

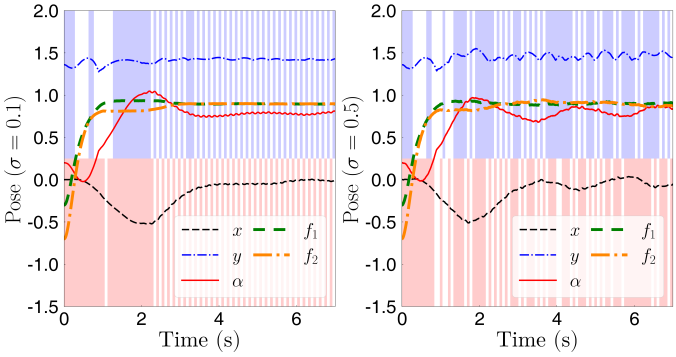


Fig. 5. For the pivoting example, Gaussian process noise is added with with standard deviations σ. The upper blue shading implies that gripper 1 is applying normal force to the object whereas the lower red shading implies that gripper 2 is applying normal force.

TABLE II
SOLVE TIME AND OPTIMALITY COMPARISON FOR PIVOTING

Item	Formulation	MPC Horizon N			
		10	20	30	50
Solve Time (s)	MIQP	0.132	0.376	1.676	> 300
	C3	0.059	0.111	0.167	0.334
Accumulated Cost ($\times 10^5$)	MIQP	1.214	0.931	0.865	-
	C3	1.217	0.954	0.951	0.968

TABLE III
PROJECTION RUN-TIME AND AVERAGED COST-TO-GO

Projection Method	Mean $\pm$ Std (s)	Cost
LCP	$1.4 \cdot 10^{-5} \pm 1.8 \cdot 10^{-6}$	22.09
MIQP	$1 \cdot 10^{-3} \pm 1.2 \cdot 10^{-4}$	24.98
ADMM	$5.3 \cdot 10^{-4} \pm 3.3 \cdot 10^{-5}$	35.74

We present an example where the fingers start close to the pivot point where $f_1 = -0.3$, $f_2 = -0.7$ and the objects configuration is given by $x = 0$, $y = 1.36$ and $\alpha = 0.2$. The goal is to balance the object at the midpoint ($x = 0$, $y = \sqrt{2}$, $\alpha = \pi/4$) while simultaneously moving the fingers towards the end of the object ($f_1 = f_2 = 0.9$). Figure 5 demonstrates the robustness of the controller for different Gaussian disturbances (added to dynamics) with standard deviations (for $\sigma = 0.1, 0.5$). Note that at every time step, all positions and velocities (including angular) are affected by the process noise. The object approximately reaches the desired configuration (midpoint where $\alpha = \frac{\pi}{4}$) for $\sigma = 0, 0.05, 0.1$ and starts failing to get close to the desired configuration for $\sigma = 0.5$. Plots with $\sigma = 0$ and $\sigma = 0.05$ are omitted as those were similar to the one with $\sigma = 0.1$. We emphasize that the process noise causes unplanned mode changes and the controller seamlessly reacts. We also observe that C3 succeeds across a wide range of frictional conditions; we sweep the ground contact parameter μ_3 from 0.1 to 1 with 0.1 increments, and observe that controller is always successful. This example demonstrates that our method works well with successive linearizations as many multi-contact systems can not be captured via a single LCS approximation. Similar to the finger gaiting example, we report the speed and optimality comparison[6] between C3 and the MIQP formulation in Table II. For this example, we calculate the accumulated cost until 1.5 s with $s = 5, \rho = 0.02, \rho_s = 1.1$. We can observe that with a horizon of $N = 50$, it takes more than 300 seconds for the MIQP formulation to compute the control input while C3 managed to solve the problem in less than 0.5 seconds with only a minimal increase in accumulated cost.

C. Cart-pole with Soft Walls

In this subsection, we focus on an under-actuated, multi-contact system and demonstrate the effectiveness of our approach via hardware experiments. We consider a cart-pole that can interact with soft walls as in Figure 6. This is a benchmark in contact-based control algorithms [13], [42]. In the hardware

[6]The tests were run in a Python script for both methods.

setup, a DC motor with a belt drive generates the linear motion of the cart. Soft walls are made of open-cell polyurethane foam. The experiments in this subsection are done on a desktop computer with the processor Intel *i7-6700HQ* and *16GB RAM*.

First, numerical experiments are presented. Here, $x^{(1)}$ represents the position of the cart, $x^{(2)}$ represents the position of the pole and $x^{(3)}, x^{(4)}$ represent their velocities respectively. The forces that affect the pole are described by $\lambda^{(1)}$ and $\lambda^{(2)}$ for right and left walls respectively.

The model is linearized around $x^{(2)} = 0$ and $m_c = 0.978$ is the mass of the cart, $m_p = 0.411$ is the combined mass of the pole and the rod, $l_p = 0.6$ is the length of the pole, $l_c = 0.4267$ is the length of the center of mass position, $k_1 = k_2 = 50$ are the stiffness parameter of the walls, $d = 0.35$ is the distance between the origin and the soft walls. Dynamics are discretized using the explicit Euler method with time step $T_s = 0.01$ to obtain the system matrices and use the model in [42]. We note that the MIQP formulation (as in (15)) runs at approximately 10 Hz.

We design a controller where $s = 10$, $\rho = 0.1$, $G_k = I$, $\rho_s = 2$, $N = 10$, and $\Delta t = 0.01$ for $\mathcal{L}_{\Delta t}$. There is a clear trade-off between solve time and planning horizon [56]. We test three projection methods described in Section V and report run-times (single solve of (23)) on Table III averaged for 800 solves. We also report the average of cost-to-go value assuming all of the methods can run at 100 Hz. Even though the MIQP projection is usually better at exploring new contacts, it does not always result in a better cost. We note that the controller can run slightly faster than 240 Hz if the LCP-based projection is used. For this example, the QP in (21) has no inequality constraints and can therefore the KKT conditions can be directly solved.

Next, we test the multi-contact MPC algorithm on the experimental setup shown in Figure 6. We consider the same LCS model with $k_1 = k_2 = 100$ as the stiffness parameters of the walls and $d = 0.39$ as the distance between the origin and walls.

For the actual hardware experiments, the parameters of the MPC algorithm are $N = 10$, $\rho_s = 2.3$, $s = 10$, and $G_k = I$, $\rho = 0.5$. We use the LCP-based projection method, and solve the quadratic programs via utilizing the KKT system.

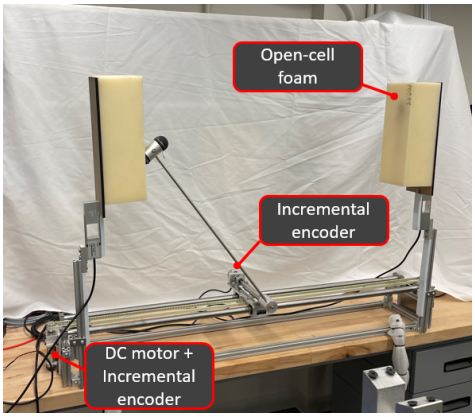


Fig. 6. Experimental setup for cart-pole with soft walls.

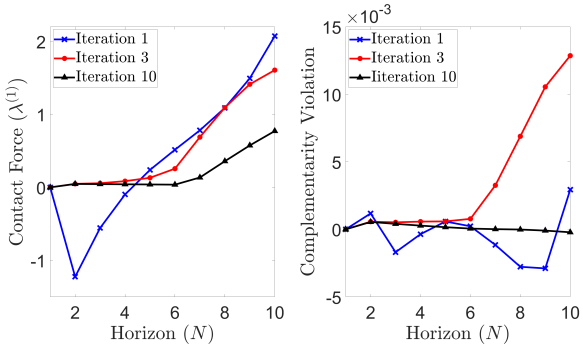


Fig. 7. Evolution of contact force and complementarity violation during ADMM iteration when the cart is close to a contact surface.

While the algorithm is capable of running faster than 240 Hz, due to the limited bandwidth between our motor controller and computer, we run the system at 100 Hz. In Figure 7, we illustrate, for one particular state, the evolution of the contact force throughout the ADMM process (at ADMM steps 1, 3, and 10). Notice that as the the algorithm progresses, complementarity violation decreases.

We initialize the cart at the origin and introduce random perturbations to cover a wide range of initial conditions that lead to contact events. Specifically, we start the cart-pole at the origin where our controller is active. Then, we apply an input disturbance $u_{\text{dist}} \sim U[10, 15]$ for 250 ms to force contact events. We repeated this experiment 10 times and our controller managed to stabilize the system in all trials.

To empirically evaluate the gap between C3, which is sub-optimal, and true solutions to (14), and to assess the impact of modeling errors, we report the approximate cost-to-go values for our method, MIQP solution (as in (15)), and the actual observed states. More precisely, given the current state of the nonlinear plant, we calculate the cost as in (14) using the inputs recovered from both the C3 algorithm and MIQP algorithm. Since C3 algorithm is not guaranteed to produce a (x, u, λ) that strictly satisfies complementarity, the predicted cost may not match the simulated cost once u is applied. For the actual plant, we use the data from N steps into the future and calculate the same cost. Notice that even

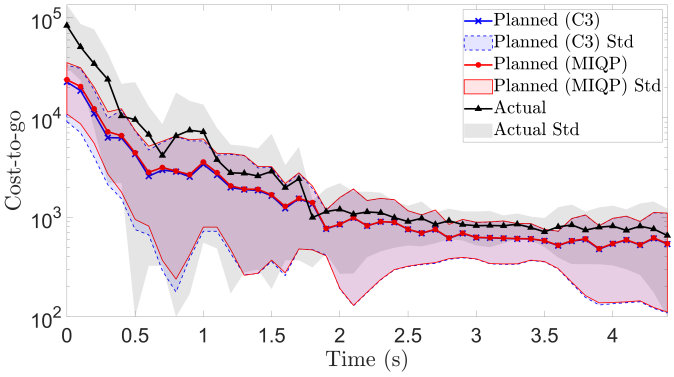


Fig. 8. Approximate cost-to-go (cost as in (15)) values for the cart-pole experiment.

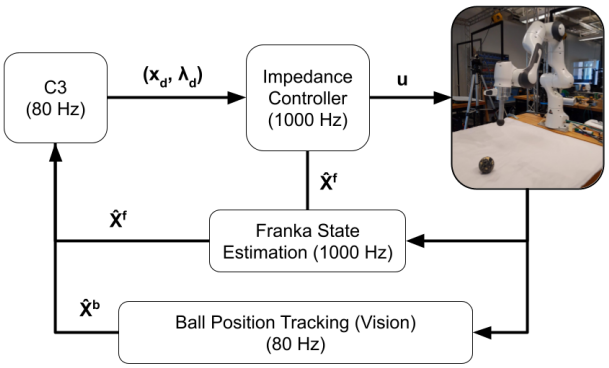


Fig. 9. Key elements of the controller diagram.

though the cost-to-go of MIQP solution is always lower, as expected, the optimality gap is fairly small. This highlights that C3 finds near-optimal solutions, at least for this particular problem. Also, notice that the actual cost-to-go matches the planned one which further motivates the applicability of LCS representations in model-based control for nonlinear multi-contact systems.

VII. Robot Arm Manipulation

In this section, we demonstrate the effectiveness of our method on multiple robot arm manipulation tasks. In addition to that, we show that C3 can reliably be used as a high-level, real-time controller for multi-contact manipulation tasks that require high-speed reasoning about contact events. For these tasks, our goal is moving one or more rigid spheres along a desired trajectory using a Franka Emika Panda Arm with a spherical end-effector (Figure 1). We note that the tasks involve multi-contact interaction between the arm and sphere(s). The robot has to decide on which side/location to make contact, and, in the multi-sphere case, which sphere to touch. Similar to the finger gaiting task in Section VI, and in contrast with comparatively simpler planar pushing tasks, these sphere-rolling problems require constantly making and breaking contact to reposition the hand. In addition to that the spheres are quite rigid, and they roll with very little dissipation. Hence it is critical for the controller to make rapid decisions to prevent them from escaping.

Towards this goal, we use the control architecture in Figure 9. In this scheme, C3 acts as a high-level controller and provides the desired state, contact force pair as proposed in Algorithm 3 (we note that one can choose to use simplified models if the model is high dimensional). Then, a low-level impedance controller tracks these desired values. Please check the Appendix for details on Algorithm 3, the simplified model and the impedance control scheme.

Both C3 and the impedance controller were implemented using the Drake toolbox [57], and simulations were performed in Drake environment. OSQP [52] is used to solve quadratic programs and Gurobi [53] is used for mixed integer programs. The experiments are done on a desktop computer with the processor Intel *i7-11800H* and *16GB RAM*. Reported run-times include all steps in the algorithm.

Algorithm 3

Require: $\mathcal{M}, \theta, \Delta t, \hat{u}$

1: Estimate state as $\hat{x}$
2: Obtain LCS model $\mathcal{L}_{\Delta t}$ around $(\hat{x}, \hat{u})$ using $\mathcal{M}$
3: Run Algorithm 1 with $\mathcal{L}_{\Delta t}$, θ, $\hat{x}$ and obtain u_{C3}
4: Compute Δt_c: Time spent during steps 1,2,3
5: **return** $(x_d, \lambda_d) = \mathcal{L}_{\Delta t_c}(\hat{x}, u_{\text{C3}})$

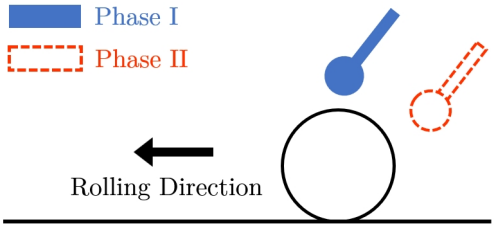


Fig. 10. A cartoon illustrating the the time-based heuristic. The blue solid end-effector represents the end-effector bias in Phase I, and the red dashed end-effector represents the end-effector bias in Phase II.

A. Simulation example: Trajectory tracking with a single ball

In this subsection, our goal is to move a rigid ball in a circular path inspired by works such as [58]. We also aim to make our simulation experiments as close to our hardware experiments (Section VII-C) as possible. For the hardware experiments, the vision setup tracks position of the ball, but not orientation. To emulate this, we measure the translational ball state $x^b_{x,y,z}$ at 80 Hz rate and estimate translational velocity of the ball via finite differencing. Similarly, angular velocity is estimated by assuming that the ball is always rolling without slipping. The angular displacement is calculated by integrating the estimated angular velocity.

To track a circular path, we form an LCS approximation around the current state and a quadratic cost for distance to a target state. The target (desired) state for the ball x^b_d is calculated according to its current state x^b. Let $(x_c, y_c) = (0.55, 0)$ be the center of the desired ball path and $r = 0.1$ be its radius. We define α^b as the current phase angle of the ball along its circular path, measured clockwise from the positive y-axis of Franka's base frame. More concretely, $\alpha^b = \text{atan2}(x^b_x - x_c, x^b_y - y_c)$, where x^b_x and x^b_y are the current x and y coordinates of the ball respectively. Given the center of the path (x_c, y_c), the radius of the path r, and the phase angle of the ball α^b, we generate the next desired ball state x^b_d as:

$$\begin{aligned} x^b_{d,x} &= x_c + r\sin(\alpha^b + \alpha^l), \\ x^b_{d,y} &= y_c + r\cos(\alpha^b + \alpha^l), \end{aligned} \tag{25}$$

where $x^b_{d,x}$ is the desired x coordinate of the ball, $x^b_{d,y}$ is the desired y coordinate of the ball and $\alpha^l = 20^\circ$ is a design parameter that controls how far ahead the desired ball state should be along the circle with respect to its current position.

One might easily warm-start C3 with an initial trajectory, and corresponding nominal hybrid plan. Here, to demonstrate that the algorithm can reliably discover trajectories in real-time

from scratch, we explicitly do not provide such a warm-start. Instead, we add a time-based heuristic as shown in Figure 10. In Phase I (1 s long), we pick the end-effector bias slightly above the ball's current state (shown in blue). In Phase II (0.5 s long), we pick a set of end-effector biases so that it ends up slightly behind the ball with respect to the desired ball location (shown in red) and also increase the cost on end-effector deviation. It is important to note that we do not specify when and how the contact interactions should happen. All the end-effector biases are above the ball, and algorithm decides that the end-effector should interact with the ball on its own (as there is also penalty on ball location error). Moreover, C3 can decide that an interaction should happen in either Phase I and Phase II, but we softly encourage interactions to happen in Phase I and the end-effector to reposition in Phase II.

The simplified model, $\mathcal{M}_s$, has nineteen states ($n_x = 19$), twelve complementarity variables ($n_\lambda = 12$) and three inputs ($n_u = 3$). We design a high-level controller where $s = 2$, $\rho = 0.1$, $G_k = I$, $\rho_s = 3$, $\Delta t = 0.1$ and $N = 5$. The controller, C3, can run around 73 Hz (we note that solving the problem via MIQP approach as in (15) runs around 5 Hz and fails to accomplish the task, also similarly fails when we early terminate the MIQP to make it faster). We strictly follow Algorithm 3 where a different LCS model $\mathcal{L}$ is generated using $\mathcal{M}_s$ to generate the desired state, contact force pair, (x_d, λ_d). Afterwards, the impedance controller (36) is used to track the the state, contact force pair as discussed.

We performed two different trials to demonstrate the accuracy of our approach as well its robustness against disturbances. For the first (Figure 11, left), we assume that there is no noise on ball position estimation. It is important to emphasize that there are still errors caused due to imperfect state estimation (as we emulate the hardware vision setup) but C3 is robust to these errors and the ball stays within the 2 cm error band.

For the second example (Figure 11, right), our goal is to demonstrate that C3 is robust and can recover even if things go wrong (e.g. ball goes out of 2 cm error band). We add noise in state estimation of the ball (Gaussian noise with 1 mm standard deviation) and increase the cost on desired ball location error to achieve faster motions. Since we increase

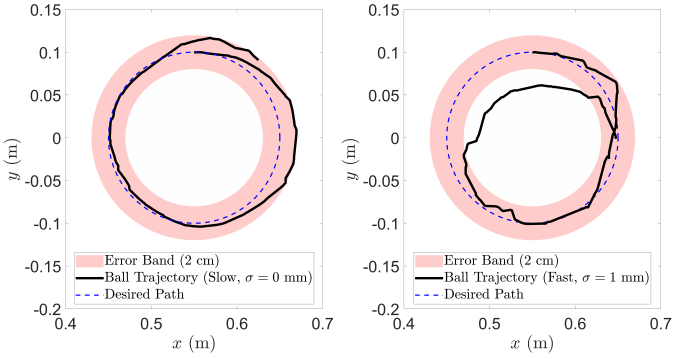


Fig. 11. Trajectory tracking with single ball: Pink regions are the 2 cm error bands, black lines represent the trajectory of the ball, and blue dashed lines represent the desired paths. Left shows the slow trajectory ($\sim$ 30 s for full circle) and right shows the fast trajectory ($\sim$ 21 s for full circle).

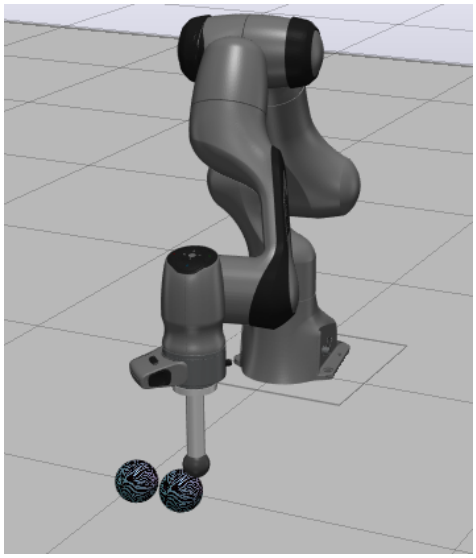


Fig. 12. The Franka Emika Panda manipulating multiple rigid spheres.

the cost on ball position error, C3 gives more aggressive commands to the low-level controller. Regardless, the ball stays in 2 cm error band in roughly 75% of the motion. Even if the ball rolls away from the 2 cm error band, the controller manages to recover and bring it back into the band utilizing its real-time planning capabilities. This further demonstrates the ability of C3 to recover from perturbations via re-planning contact sequences. In the supplementary video, we present multiple circular trajectories tracked in a row (for both cases) to demonstrate the reliability of our approach.

B. Simulation example: Trajectory tracking with multiple balls

We consider a similar scenario to the previous example but with multiple balls (Figure 12). The goal is to manipulate all of the balls to follow a line. Since the number of potential interactions (ball-ball and end-effector-ball interactions) are higher than the previous example, the hybrid decision making problem is significantly harder to solve. As the number of balls increase, the problem becomes high dimensional (with respect to both the state and the complementarity variable). To slightly reduce the dimension of this problem, we assume that the balls are always on the table and do not slide. We have tried this task both with 2 balls and 3 balls. The reduced-order model for 2-ball setup has fourteen states ($n_x = 14$), eighteen complementarity variables ($n_\lambda = 18$) and three inputs ($n_u = 3$). For the 3-ball setup, the reduced-order model has eighteen states ($n_x = 18$), twenty four complementarity variables ($n_\lambda = 24$) and three inputs ($n_u = 3$).

For i^{th} ball, x^{b_i}, we pick the desired ball state as:

$$x^{b_i}_{d,x} = x_c,$$
$$x^{b_i}_{d,y} = x^{b_i}_{y} - d_y,$$

where x_c is a constant value and d_y is the desired translation on y direction. We also follow the same heuristic as the previous section and set the end-effector bias with respect to the ball that has made the least progress in the y direction.

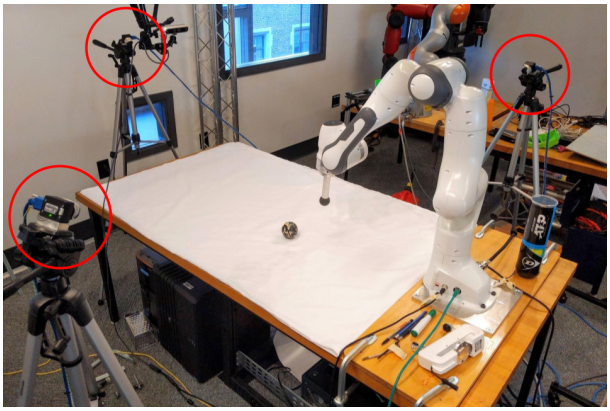


Fig. 13. Camera setup for hardware experiments. The 3 PointGrey cameras are circled in red.

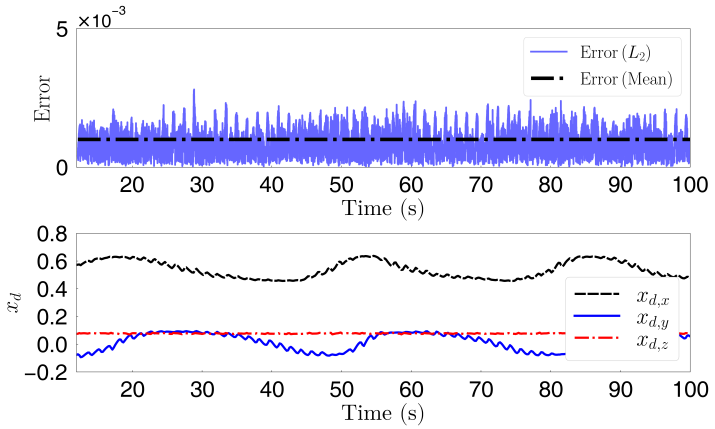


Fig. 14. End-effector position tracking error ($\sqrt{(\tilde{x}_x^e)^2 + (\tilde{x}_y^e)^2 + (\tilde{x}_z^e)^2}$) and desired end-effector locations given by C3 (x_d) .

Fig. 15. An example of a left to right roll using C3 as a high-level planner. Despite the provided heuristic, C3 positions the end-effector directly above the ball at the start of the motion (left image) and rolls from the front.

For the 2-ball setup, we design a controller that can run around 40 Hz where $s = 2$, $\rho = 0.1$, $G_k = I$, $\rho_s = 3$, $\Delta t = 0.1$ and $N = 5$. Similar to the previous section, we follow Algorithm 3 to generate desired states and forces (x_d, λ_d). Impedance controller (36) is used to track the desired states, and feedforward torque term is taken as zero $(\tau_{ff} = 0)$ for this specific example.

We demonstrate the value of the convex projection (Section V-B3) on the 3-ball setup. Here, with identical parameters as in the 2-ball setting, the MIQP projection (with the Anitescu contact model from Section IV-B) runs at approximately 30 Hz. With the convex projection, the controller rate increases to 100 Hz. While this projection is approximate, we note empirically that the resulting controller is high performance and succeeds at the task.

It is important to emphasize that we do not warm-start our algorithm with an initial trajectory and C3 decides on contact interactions (e.g. which ball to interact with, the interaction between balls themselves) on its own. We have successfully accomplished the task with 2 balls where the controller runs around 40 Hz. For the 3-ball setup, C3 with MIQP projection ran at 30 Hz, which was empirically too slow to stably complete the task. By using the convex projection, we can increase the control rate to 100 Hz, where the task was completed successfully. A full video of both experiments are provided in the supplementary material.

C. Hardware experiment: State-based trajectory

In addition to providing numerical examples, we also performed the experiment from Section VII-A on hardware. As before, our goal is to roll the rigid ball in Figure 1 along a circular path of radius 0.1 m, centered at (x_c, y_c) in Franka's base frame. We use the same low-level impedance controller presented in (36) and C3 as a high-level controller. The next desired state for the ball is computed using the same state-based method described in equation (25).

In order to complete this experiment on hardware, we need a vision pipeline to estimate the translational state of the ball, $x^b_{x,y,z}$. We accomplish this by positioning 3 PointGrey cameras around the robot as shown in Figure 13. Each camera locates the ball using the Circle Hough Transform algorithm [59],[60], and computes an estimate of $x^b_{x,y,z}$ according to its extrinsic and intrinsic calibration. Next, the position estimates from all 3 cameras are compared, outliers are rejected, and the remaining estimates are averaged. The averaged measurements pass through a low-pass filter to produce the final estimate of the ball's translational state, $\hat{x}^b_{x,y,z}$. The ball's translational velocity, orientation, and angular velocity are estimated from $\hat{x}^b_{x,y,z}$ using the same procedure described in Section VII-A. The full vision pipeline operates at roughly 80-90 Hz and the position estimates typically vary between a range of ± 1 mm.

The desired end-effector positions (x^e_d) that C3 generates as the high-level controller and the tracking error of the low-level impedance controller are shown in Figure 14. Note that the magnitude of the end-effector's translational error is consistently lower than 0.003 m. These plots suggests that C3 produces effective trajectories that a low-level controller, such as our impedance controller, can realistically track.

The trajectory of the ball is shown in Figure 17, where the black line represents the ball path and the red line represents the desired trajectory. Each full rotation around the circle takes 35 s on average to complete. We allow the robot to complete multiple loops to demonstrate the consistency of our approach. A full video of this experiment is provided in the supplementary material.

In this experiment, we use the same time-based heuristic for end-effector biasing shown in Figure 10. Although Phase II of this heuristic encourages the robot to roll the ball from the back, C3 can roll from the front instead (adapting to

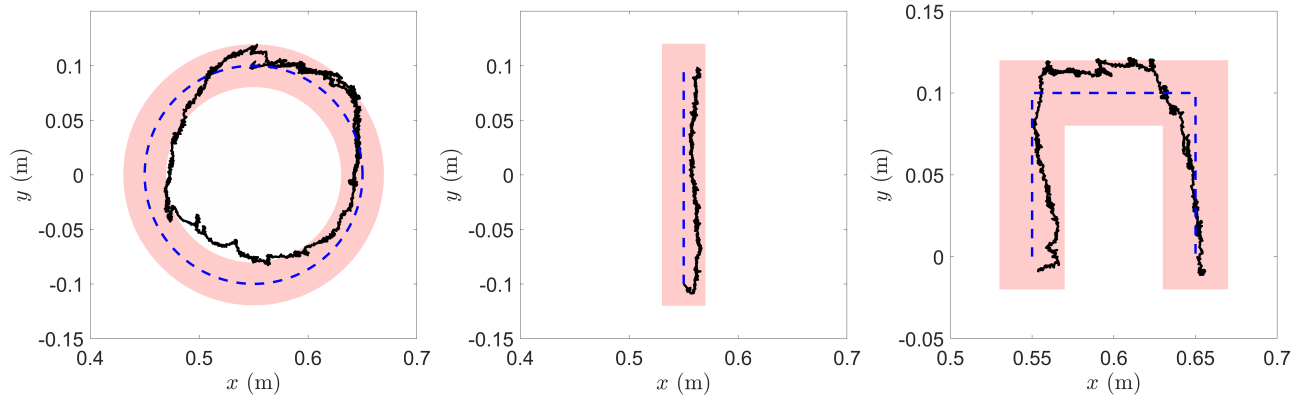


Fig. 16. Hardware experiment: Tracking time-based trajectories of different shapes. Pink regions are the 2 cm error bands, black lines represent the states of the ball during the motion, and blue dashed lines represents the desired path.

errors) during the actual experiments (see Figure 15). This re-emphasizes the fact that we do not specify when or how the robot should interact with the ball, or predefine any trajectory for the end-effector to follow. It also demonstrates

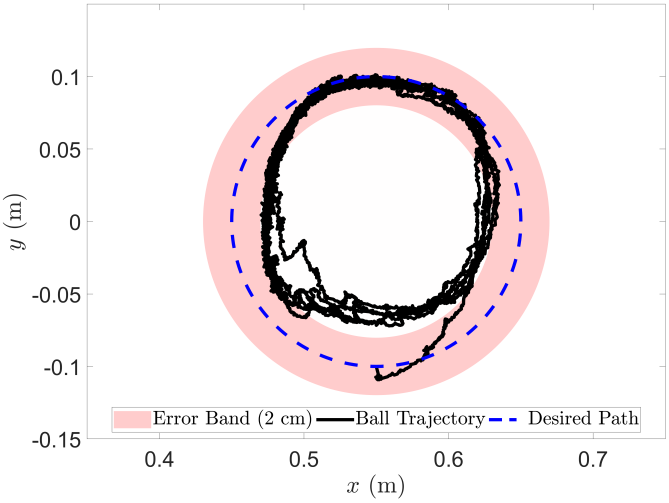


Fig. 17. Hardware experiment: Tracking a state-based circular path. Pink region is the 2 cm error band, black line demonstrates the actual trajectory of the ball, and blue dashed line represents the desired path.

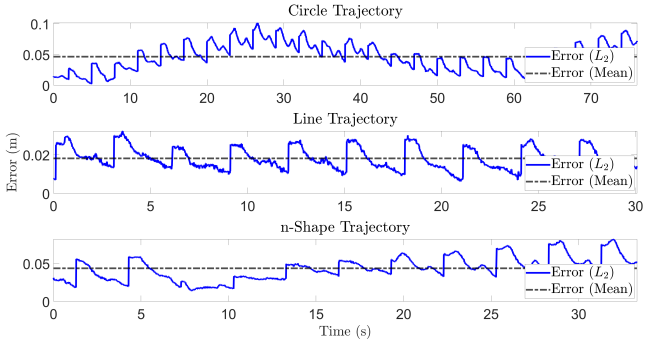


Fig. 18. Ball position tracking error ($\sqrt{\left(\tilde{x}_x^b(t)\right)^2 + \left(\tilde{x}_y^b(t)\right)^2}$ where $\tilde{x}^b(t) = x_d^b(t) - x^b(t)$) for the three time based trajectories shown in Figure 16.

that C3 does not require perfectly tailored a priori heuristics to perform well. This provides two major advantages. Firstly, it enables C3 to recover, even when its plans deviate from the provided heurisitc. Secondly, it eliminates the need for heavily-engineered heuristics, which may be difficult or time-consuming to develop in practise.

Since C3 plans in real-time ($70 - 80$ Hz), it also exhibits some robustness to model error. Recall that C3 plans using the simplified model, $\mathcal{M}_s$. This model assumes that the rolling surface is perfectly horizontal; however, the table in the hardware setup is slightly tilted.

This tilt is observable in the supplementary video: note that the ball tends to roll towards the right side of the camera frame whenever it is not in contact with the robot. As a result, the overall shape of the ball's trajectory in Figure 17 is circular, but it is shifted in the positive y direction. Another potential source of model error in $\mathcal{M}_s$ is the friction coefficient between the ball and end-effector, which was simply set to $\mu = 1$. Despite these model errors, C3 was still able to generate plans through contact to successfully and consistently complete the task.

Furthermore, the real-time planning capabilities of C3 enable it to continually correct any errors in the ball's trajectory. This may occur if the high-level controller generates an ineffective plan or if the low-level controller momentarily fails to track the end-effector trajectory. For instance, on one of the loops in Figure 17, the robot erroneously rolls the ball to $(x_x^b, x_y^b) \approx (0.5, -0.02)$, nearly 5 cm away from the desired path. Nonetheless, C3 is able to correct this mistake on the subsequent rolls and return the ball to its nominal trajectory. This demonstrates C3's self-correction abilities.

Lastly, we also demonstrate that C3 can account for external disturbances where we manually disturb the ball during the experiment. C3 accomplishes the task well even with such disturbances (shown in supplementary video).

D. Hardware experiment: Time-based trajectory

In this section, we conduct hardware experiments where the desired ball trajectory is parameterized as a function of time rather than the ball's current state. All other aspects of the experimental setup (the system models, high-level control

algorithm, low-level controller, and vision pipeline) remain unchanged from Section VII-C.

We experimented with the 3 different time-based trajectories, $x_d^b = x_d^b(t)$ shown in Figure 16, and the tracking error for these three trajectories are shown in Figure 18. For all 3 time-based trajectories, C3 successfully and consistently discovers strategies to roll the ball around the 2 cm error band. Additionally, the shape of the ball's trajectory closely matches the desired trajectory. These experiments demonstrate that C3 can also track time-based trajectories. Note that the same model errors (e.g. table slant, friction coefficient estimates, etc) from Section VII-C are present in these experiments as well; however, since the high-level control algorithm has not changed, C3 remains sufficiently robust to successfully execute the tasks. The full videos of these experiments are available in the supplementary material.

VIII. Conclusion

In this work, we present an algorithm, C3, for model predictive control of multi-contact systems. The algorithm relies on solving QP's accompanied by projections, both of which can be solved efficiently for multi-contact systems. The effectiveness of our approach is verified on five numerical examples and our results are validated on two different experimental setups. We demonstrate that C3 can be reliably used both as a high-level and a low-level controller. For fairly complex examples with frictional contact, our method has a fast run-time. We also demonstrate through experiments that our heuristic can find close-to-optimal strategies.

The framework tackles the hybrid MPC problem by shifting the complexity to the projection sub-problems. These projections can be difficult to solve, especially as we rely on the MIQP-based projection method for problems with frictional contact. Exploring alternate heuristics for the projection step is of future interest. Similarly, it would be interested to leverage learning-based approaches [49] to speed up the process.

As our approach is purely model-based, it is highly dependent on model parameters but some of those such as coefficient of friction can be hard to estimate accurately. Integration with learned models [61], hopefully in an adaptive way is in the scope of future work.

As discussed in Section VII, C3 generates desired contact forces but these have been used in a feedforward manner. It would be interesting to utilize tactile sensors and tactile feedback controllers [42], [62] that track these desired contact forces. We believe that this can improve the robustness of our approach and is one of the crucial steps moving forward.

The choice of parameters (such as C3 parameters θ) greatly affects the performance of the algorithm and exploring different approaches for deciding on parameters is of future interest too.

Appendix

A. ADMM

We will clarify the derivation of general augmented Lagrangian (17) and ADMM iterations (18), (19), (20). Consider the equivalent optimization problem to (16):

$$\begin{aligned} \min_{z} \quad & c(z) + \mathcal{I}_{\mathcal{D}}(z) + \mathcal{I}_{\mathcal{C}}(z) + \sum_{k=0}^{N-1} \mathcal{I}_{\mathcal{H}_k}(\delta_k) \\ \text{s.t.} \quad & T(z-\delta) = 0 \end{aligned} \tag{26}$$

where $T = \mathbf{blkdiag}(T_0, \ldots, T_{N-1})$ and $T_k^T T_k = G_k$. The augmented Lagrangian is:

$$\begin{aligned} \mathcal{L}_y(z,\delta,y) &= c(z) + \mathcal{I}_{\mathcal{D}}(z) + \mathcal{I}_{\mathcal{C}}(z) \\ &+ \sum_{k=0}^{N-1} \big(\mathcal{I}_{\mathcal{H}_k}(\delta_k) + y_k^T T_k(z_k - \delta_k) + \rho ||T_k(z_k-\delta_k)||_2^2\big), \end{aligned}$$

where $y^T = [y_0^T, y_1^T, \ldots, y_{N-1}^T]$, y_k are the dual variables. In order to solve (26), ADMM iterations are:

$$z^{i+1} = \text{argmin}_z \mathcal{L}_y(z, \delta^i, y^i), \tag{27}$$

$$\delta^{i+1} = \text{argmin}_\delta \mathcal{L}_y(z^{i+1}, \delta, y^i), \tag{28}$$

$$y^{i+1} = y^i + 2\rho T(z^{i+1} - \delta^{i+1}). \tag{29}$$

Notice that individual vectors δ_k^{i+1} in (28) can be computed as

$$\delta_k^{i+1} = \text{argmin}_{\delta_k} \mathcal{L}_y^k(z_k^{i+1}, \delta_k, y_k^i)$$

due to separability of $\mathcal{L}_y$, where $\mathcal{L}_y^k(z_k,\delta_k,y_k) = \big(\mathcal{I}_{\mathcal{H}_k}(\delta_k) + y_k^T T_k(z_k-\delta_k) + \rho||T_k(z_k-\delta_k)||_2^2\big)$. Also note that δ_k^{i+1} only depends on z_k^{i+1} and y_k^i. Similarly (29) can be written as

$$y_k^{i+1} = y_k^i + 2\rho T_k(z_k^{i+1} - \delta_k^{i+1}).$$

After that, define the scaled dual variables w_k such that $y_k = 2\rho T_k w_k$. With w_k, the augmented Lagrangian is of the following form:

$$\begin{aligned} \mathcal{L}_\rho(z,\delta,w) &= c(z) + \mathcal{I}_{\mathcal{D}}(z) + \mathcal{I}_{\mathcal{C}}(z) \\ &+ \sum_{k=0}^{N-1} \big(\mathcal{I}_{\mathcal{H}_k}(\delta_k) + 2\rho w_k^T G_k(z_k - \delta_k) \\ &+ \rho(z_k-\delta_k)^T G_k(z_k-\delta_k)\big) \end{aligned}$$

following the fact that $G_k = T_k^T T_k$. It is equivalent to:

$$\begin{aligned} \mathcal{L}_\rho(z,\delta,w) &= c(z) + \mathcal{I}_{\mathcal{D}}(z) + \mathcal{I}_{\mathcal{C}}(z) \\ &+ \sum_{k=0}^{N-1} \big(\mathcal{I}_{\mathcal{H}_k}(\delta_k) + \rho(r_k^T G_k r_k - w_k^T G_k w_k)\big). \end{aligned}$$

The corresponding ADMM iterations are:

$$\begin{aligned} z^{i+1} &= \text{argmin}_z \mathcal{L}_\rho(z, \delta^i, w^i), \\ \delta_k^{i+1} &= \text{argmin}_{\delta_k} \mathcal{L}_\rho^k(z_k^{i+1}, \delta_k, w_k^i), \ \forall k, \\ w_k^{i+1} &= w_k^i + z_k^{i+1} - \delta_k^{i+1}, \ \forall k \end{aligned}$$

as $y_k = 2\rho T_k w_k$ and T_k is a positive definite matrix with $\mathcal{L}_\rho^k(z_k,\delta_k,w_k) = \mathcal{I}_{\mathcal{H}_k}(\delta_k) + \rho(r_k^T G_k r_k - w_k^T G_k w_k)$.

B. Proof of Lemma 3

The Lagrangian for the optimization problem (24) is:

$$\begin{aligned}\mathcal{L}\left(\delta_k^x, \delta_k^\lambda, \delta_k^u, \mu, \eta\right) = &\frac{1}{2}\left\|\delta_k^x - x_d\right\|_{Q_x} + \frac{1}{2}\left\|\delta_k^u - u_d\right\|_{Q_u} \\ &+ \frac{\alpha}{2}\left\|\delta_k^\lambda - \lambda_d\right\|_F + \frac{1-\alpha}{2}\left\|\delta_k^\lambda\right\|_F \\ &- \mu^T\left(E\delta_k^x + F\delta_k^\lambda + H\delta_k^u + c\right) - \eta^T\delta_k^\lambda,\end{aligned}$$

where $\mu \geq 0$ and $\eta \geq 0$. We consider the stationarity conditions:

$$\begin{aligned}&\frac{d\mathcal{L}}{d\delta_k^x} = 0 \implies Q_x\left(\delta_k^x - x_d\right) - E^T\mu = 0,\\ &\frac{d\mathcal{L}}{d\delta_k^u} = 0 \implies Q_u\left(\delta_k^u - u_d\right) - H^T\mu = 0,\\ &\frac{d\mathcal{L}}{d\delta_k^\lambda} = 0 \implies \alpha F\left(\delta_k^\lambda - \lambda_d\right) + (1-\alpha)F\delta_k^\lambda - F\mu - \eta = 0.\end{aligned}$$

It follows that:

$$\begin{aligned}\delta_k^x &= Q_x^{-1}E^T\mu + x_d,\\ \delta_k^u &= Q_u^{-1}H^T\mu + u_d,\\ \delta_k^\lambda &= \mu + F^{-1}\eta + \alpha\lambda_d.\end{aligned}$$

Primal and dual feasibility, and the complementary slackness constraints are:

$$\begin{aligned}&E\delta_k^x + F\delta_k^\lambda + H\delta_k^u + c \geq 0,\\ &\mu \geq 0,\\ &\mu^T\left(E\delta_k^x + F\delta_k^\lambda + H\delta_k^u + c\right) = 0,\\ &\eta \geq 0,\\ &\delta_k^\lambda \geq 0,\\ &\eta^T\delta_k^\lambda = 0.\end{aligned}$$

Let $\Theta = E\delta_k^x + F\delta_k^\lambda + H\delta_k^u + c$, consider the complementarity error term from the original LCS (1):

$$\begin{aligned}\text{error}_{\text{comp}} = (\delta_k^\lambda)^T\Theta &= \left(\delta_k^\lambda - \mu + \mu\right)^T\Theta = \left(\delta_k^\lambda - \mu\right)^T\Theta \\ &\leq \left\|\delta_k^\lambda - \mu\right\|\|\Theta\|.\end{aligned} \tag{30}$$

We wish to show that this error is governed by α (specifically, that it goes to 0 as $\alpha \to 0$). First, we analyze the term $\left\|\delta_k^\lambda - \mu\right\|$, and observe that:

$$\delta_k^\lambda - \mu = F^{-1}\eta + \alpha\lambda_d \implies \left\|\delta_k^\lambda - \mu\right\| \leq \left\|F^{-1}\right\|\|\eta\| + \alpha\|\lambda_d\|.$$

From the complementarity slackness and stationarity, we have:

$$\eta^T\delta_k^\lambda = \eta^TF^{-1}\eta + \eta^T\mu + \alpha\eta^T\lambda_d = 0,$$

and from dual feasibility we have:

$$\eta \geq 0, \mu \geq 0 \implies \eta^T\mu \geq 0.$$

Combining the two, it follows that:

$$\eta^TF^{-1}\eta \leq -\alpha\eta^T\lambda_d.$$

Since F is symmetric positive definite, it follows that:

$$\|\eta\| \leq \frac{\alpha\|\lambda_d\|}{\sigma_{\min}(F^{-1})},$$

and therefore that,

$$\left\|F^{-1}\right\|\|\eta\| \leq \alpha\|\lambda_d\|\,\text{cond}(F^{-1}). \tag{31}$$

Thus, we can conclude that

$$\left\|\delta_k^\lambda - \mu\right\| \leq \alpha\|\lambda_d\|\left(1 + \text{cond}(F^{-1})\right) \tag{32}$$

proving a linear bound in α. Next, we analyze the term $\|\Theta\|$, expanding the term using the stationarity condition:

$$\begin{aligned}\|\Theta\| = \|&(EQ_x^{-1}E\mu + HQ_u^{-1}H + F)\mu + \eta \\ &+ Ex_d + \alpha F\lambda_d + Hu_d + c\|,\end{aligned}$$

so we need to analyze how μ behaves when α changes. While the primal solution is unique as the quadratic program is strictly convex, we do not assume LICQ and thus the optimal dual solution (e.g. μ) may not be unique. However, the optimal value can still be bounded. Since F is positive definite, it follows that there exists a $\bar{\delta}_k^\lambda$ such that ([41], Lemma 3.1.3):

$$\bar{\delta}_k^\lambda > 0, \quad F\bar{\delta}_k^\lambda > 0.$$

Therefore, there exists some $\bar{\delta}_k^x, \bar{\delta}_k^\lambda, \bar{\delta}_k^u$ where $\bar{\delta}_k^\lambda > 0$ and $E\bar{\delta}_k^x + F\bar{\delta}_k^\lambda + H\bar{\delta}_k^u + c > 0$. With this, we prove the Slater condition and that strong duality holds. With a slight abuse of notation, let p denote the primal objective function, g denote the inequality constraints in the form $g \leq 0$, d denote Lagrangian dual function with dual problem formulated as maximization, M denote the optimal Lagrangian multiplier set. We have the following bound ([63], Lemma 1):

$$\|\mu\| \leq \max_{[\mu^T,\eta^T]^T \in M}\left\|\left[\mu^T, \eta^T\right]^T\right\| \leq \frac{1}{\gamma}\left(p\left(\bar{\delta}_k^x, \bar{\delta}_k^u, \bar{\delta}_k^\lambda\right) - d\left(\mu, \eta\right)\right),$$

where $\gamma = \min_{1\leq i\leq m}\left\{-g_i\left(\bar{\delta}_k^x, \bar{\delta}_k^\lambda, \bar{\delta}_k^u\right)\right\} > 0$. In our context (optimization problem (24)), we have:

$$\begin{aligned}p\left(\bar{\delta}_k^x, \bar{\delta}_k^u, \bar{\delta}_k^\lambda\right) &= \frac{\alpha}{2}\left(\lambda_d - 2\bar{\delta}_k^x\right)^T F\lambda_d + \frac{1}{2}\left\|\bar{\delta}_k^x - x_d\right\|_{Q_x} \\ &\quad + \frac{1}{2}\left\|\bar{\delta}_k^u - u_d\right\|_{Q_u} + \frac{1}{2}\left\|\bar{\delta}_k^\lambda\right\|_F,\\ g\left(\bar{\delta}_k^x, \bar{\delta}_k^\lambda, \bar{\delta}_k^u\right) &= \left[-\left(E\bar{\delta}_k^x + F\bar{\delta}_k^\lambda + H\bar{\delta}_k^u + c\right)^T, -\left(\bar{\delta}_k^\lambda\right)^T\right]^T.\end{aligned}$$

The point $(\bar{\delta}_k^x, \bar{\delta}_k^\lambda, \bar{\delta}_k^u)$ is independent of α (which does not appear in the constraints), and can thus be regarded as constant for this analysis. Since the primal objective is non-negative, $d(\mu, \eta) \geq 0$, and strong duality holds, the dual optimal value must be non-negative and we arrive at the bound

$$\|\mu\| \leq \frac{1}{\gamma}\left(\bar{t}_0 + \bar{t}_1\alpha\right) \tag{33}$$

for some $\bar{t}_0, \bar{t}_1 \geq 0$. And thus, that $\|\Theta\|$ is similarly bounded

$$\|\Theta\| \leq t_0 + t_1\alpha, \tag{34}$$

for some $t_0, t_1 \geq 0$. Combining (30), (32) and (34), we have:

$$\text{error}_{\text{comp}} \leq \alpha\|\lambda_d\|\left(1 + \text{cond}(F^{-1})\right)\left(t_0 + t_1\alpha\right). \tag{35}$$

Therefore, $\text{error}_{\text{comp}} \to 0$ as $\alpha \to 0$ with linear rate for small α. ■

C. Algorithm 3 and Simplified Model

The algorithm requires a model $\mathcal{M}$ (Section IV), C3 parameters θ (Section V-B), discretization step length Δt, and a nominal input $\hat{u}$. First, an LCS approxmation ($\mathcal{L}_{\Delta t}$) of model $\mathcal{M}$ around the state estimate $\hat{x}$ and nominal input $\hat{u}$ is obtained. Then, (following Algorithm 1) one can use this LCS model along with the state estimate and pre-specified C3 parameters (θ) and this process returns u_{C3}. Next, we compute the time spent (Δt_c) during the previous steps and calculate an LCS model $\mathcal{L}_{\Delta t_c}$ using Δt_c as the discretization time-step. Lastly, we compute the desired state x_d and the desired contact force λ_d using $\mathcal{L}_{\Delta t_c}$, $\hat{x}$ and u_{C3} (Section IV).

Often times, C3 generates the desired end-effector trajectories and contact forces using a simplified model of the robot. Using such models for planning [64] is extremely common and improves the solve time of the planning algorithms, e.g. C3. This specific simple model, $\mathcal{M}_s$, assumes one can control the translational accelerations of the spherical end-effector (Figure 1) in all directions ($\ddot{x}^e_x, \ddot{x}^e_y, \ddot{x}^e_z$) and that the end-effector does not rotate.

D. Impedance Control

We use impedance control [65] to track the end-effector trajectories and contact forces that C3 produces which relies on model of Franka Emika Panda Arm, $\mathcal{M}_f$. Concretely, the controller we implemented is:

$$\begin{aligned} u =& J_f^T \Lambda \begin{bmatrix} K & 0 \\ 0 & B \end{bmatrix} \tilde{x}^e + C_f + \tau_{ff} \\ &+ N_f \big(K_n (q_d^f - q^f) + B_n (\dot{q}_d^f - \dot{q}^f) \big), \end{aligned} \tag{36}$$

where x^e is the current state of the end-effector, x_d^e is the desired end-effector state given by C3 and $\tilde{x}^e = x_d^e - x^e$ represents the error. The K and B are gain matrices for position error and velocity error respectively. Given $\mathcal{M}_f$, J_f is the manipulator Jacobian for the end-effector and $\Lambda = \left(J_f M_f^{-1} J_f^T\right)^{-1}$ is the end-effector inertia matrix, where M_f is the manipulator's mass matrix. The nullspace projection matrix N_f directly follows from [66], q_d^f and $\dot{q}_d^f$ are the desired nullspace position and velocity for the manipulator, and K_n, B_n are the corresponding gain matrices. The feedforward torque term is computed as $\tau_{ff} = J_c^T \lambda_d$ where λ_d is the desired contact force obtained from C3 and J_c is the contact Jacobian computed using the full-order model.

ACKNOWLEDGEMENT

We thank Philip Sieg and Terry Kientz for their help with the cart-pole experimental setup, Tianze Wang and Yike Li for their earlier explorations of ADMM-based multi-contact optimal control, Mathew Halm and William Yang for helpful discussions related to the pivoting example and visualization code, Leon Kim, Bibit Bianchini and Peter Szczesniak for their help with the Franka experimental setup.

REFERENCES

[1] 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.

[2] T. Pang, H. J. T. Suh, L. Yang, and R. Tedrake, "Global planning for contact-rich manipulation via local smoothing of quasi-dynamic contact models," 2023.

[3] Y. Shirai, X. Lin, A. Schperberg, Y. Tanaka, H. Kato, V. Vichathorn, and D. Hong, "Simultaneous contact-rich grasping and locomotion via distributed optimization enabling free-climbing for multi-limbed robots," in *2022 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pp. 13563–13570, IEEE, 2022.

[4] A. Ö. Önol, P. Long, and T. Padır, "Contact-implicit trajectory optimization based on a variable smooth contact model and successive convexification," in *2019 International Conference on Robotics and Automation (ICRA)*, pp. 2447–2453, IEEE, 2019.

[5] 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.

[6] X. Cheng, E. Huang, Y. Hou, and M. T. Mason, "Contact mode guided motion planning for quasidynamic dexterous manipulation in 3d," in *2022 International Conference on Robotics and Automation (ICRA)*, pp. 2730–2736, IEEE, 2022.

[7] A. U. Raghunathan, D. K. Jha, and D. Romeres, "Pyrobocop: Python-based robotic control & optimization package for manipulation," in *2022 International Conference on Robotics and Automation (ICRA)*, pp. 985–991, IEEE, 2022.

[8] C. E. Garcia, D. M. Prett, and M. Morari, "Model predictive control: Theory and practice—a survey," *Automatica*, vol. 25, no. 3, pp. 335–348, 1989.

[9] F. Borrelli, A. Bemporad, and M. Morari, *Predictive control for linear and hybrid systems*. Cambridge University Press, 2017.

[10] Y. Ding, A. Pandala, and H.-W. Park, "Real-time model predictive control for versatile dynamic motions in quadrupedal robots," in *2019 International Conference on Robotics and Automation (ICRA)*, pp. 8484–8490, IEEE, 2019.

[11] A. Zhakatayev, B. Rakhim, O. Adiyatov, A. Baimyshev, and H. A. Varol, "Successive linearization based model predictive control of variable stiffness actuated robots," in *2017 IEEE international conference on advanced intelligent mechatronics (AIM)*, pp. 1774–1779, IEEE, 2017.

[12] A. Bemporad and M. Morari, "Control of systems integrating logic, dynamics, and constraints," *Automatica*, vol. 35, no. 3, pp. 407–427, 1999.

[13] T. Marcucci and R. Tedrake, "Warm start of mixed-integer programs for model predictive control of hybrid systems," *IEEE Transactions on Automatic Control*, 2020.

[14] S. Boyd, N. Parikh, and E. Chu, *Distributed optimization and statistical learning via the alternating direction method of multipliers*. Now Publishers Inc, 2011.

[15] B. Stellato, V. V. Naik, A. Bemporad, P. Goulart, and S. Boyd, "Embedded mixed-integer quadratic optimization using the osqp solver," in *2018 European Control Conference (ECC)*, pp. 1536–1541, IEEE, 2018.

[16] R. Takapoui, N. Moehle, S. Boyd, and A. Bemporad, "A simple effective heuristic for embedded mixed-integer quadratic programming," *International journal of control*, vol. 93, no. 1, pp. 2–12, 2020.

[17] J. Park and S. Boyd, "General heuristics for nonconvex quadratically constrained quadratic programming," *arXiv preprint arXiv:1703.07870*, 2017.

[18] D. Frick, A. Georghiou, J. L. Jerez, A. Domahidi, and M. Morari, "Low-complexity method for hybrid mpc with local guarantees," *SIAM Journal on Control and Optimization*, vol. 57, no. 4, pp. 2328–2361, 2019.

[19] W. Heemels, J. M. Schumacher, and S. Weiland, "Linear complementarity systems," *SIAM journal on applied mathematics*, vol. 60, no. 4, pp. 1234–1269, 2000.

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

[21] P. M. Wensing, M. Posa, Y. Hu, A. Escande, N. Mansard, and A. Del Prete, "Optimization-based control for dynamic legged robots," *arXiv preprint arXiv:2211.11644*, 2022.

[22] M. Suomalainen, Y. Karayiannidis, and V. Kyrki, "A survey of robot manipulation in contact," *Robotics and Autonomous Systems*, vol. 156, p. 104224, 2022.

[23] J. Park, J. Haan, and F. C. Park, "Convex optimization algorithms for active balancing of humanoid robots," *IEEE Transactions on Robotics*, vol. 23, no. 4, pp. 817–822, 2007.

[24] Y. Abe, M. Da Silva, and J. Popović, "Multiobjective control with frictional contacts," in *Proceedings of the 2007 ACM SIGGRAPH/Eurographics symposium on Computer animation*, pp. 249–258, 2007.

[25] Z. Manchester and S. Kuindersma, "Variational contact-implicit trajectory optimization," in *Robotics Research: The 18th International Symposium ISRR*, pp. 985–1000, Springer, 2020.

[26] 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 *2020 IEEE International Conference on Robotics and Automation (ICRA)*, pp. 2536–2542, IEEE, 2020.

[27] 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.

[28] G. Bledt, *Regularized predictive control framework for robust dynamic legged locomotion*. PhD thesis, Massachusetts Institute of Technology, 2020.

[29] S. Kuindersma, R. Deits, M. Fallon, A. Valenzuela, H. Dai, F. Permenter, T. Koolen, P. Marion, and R. Tedrake, "Optimization-based locomotion planning, estimation, and control design for the atlas humanoid robot," *Autonomous robots*, vol. 40, no. 3, pp. 429–455, 2016.

[30] R. Grandia, F. Jenelten, S. Yang, F. Farshidian, and M. Hutter, "Perceptive locomotion through nonlinear model predictive control," *arXiv preprint arXiv:2208.08373*, 2022.

[31] T. Marcucci, R. Deits, M. Gabiccini, A. Bicchi, and R. Tedrake, "Approximate hybrid model predictive control for multi-contact push recovery in complex environments," in *2017 IEEE-RAS 17th International Conference on Humanoid Robotics (Humanoids)*, pp. 31–38, IEEE, 2017.

[32] 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.

[33] A. Cauligi, P. Culbertson, E. Schmerling, M. Schwager, B. Stellato, and M. Pavone, "Coco: Online mixed-integer control via supervised learning," *IEEE Robotics and Automation Letters*, vol. 7, no. 2, pp. 1447–1454, 2021.

[34] H. Zhu and L. Righetti, "Efficient object manipulation planning with monte carlo tree search," *arXiv preprint arXiv:2206.09023*, 2022.

[35] Y. Tassa, T. Erez, and E. Todorov, "Synthesis and stabilization of complex behaviors through online trajectory optimization," in *2012 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pp. 4906–4913, IEEE, 2012.

[36] J. Koenemann, A. Del Prete, Y. Tassa, E. Todorov, O. Stasse, M. Bennewitz, and N. Mansard, "Whole-body model-predictive control applied to the hrp-2 humanoid," in *2015 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pp. 3346–3351, IEEE, 2015.

[37] V. Kumar, Y. Tassa, T. Erez, and E. Todorov, "Real-time behaviour synthesis for dynamic hand-manipulation," in *2014 IEEE International Conference on Robotics and Automation (ICRA)*, pp. 6808–6815, IEEE, 2014.

[38] S. L. Cleac'h, T. Howell, S. Yang, C.-Y. Lee, J. Zhang, A. Bishop, M. Schwager, and Z. Manchester, "Fast contact-implicit model-predictive control," 2023.

[39] D. Stewart and J. C. Trinkle, "An implicit time-stepping scheme for rigid body dynamics with coulomb friction," in *Proceedings 2000 ICRA. Millennium Conference. IEEE International Conference on Robotics and Automation. Symposia Proceedings (Cat. No. 00CH37065)*, vol. 1, pp. 162–169, IEEE, 2000.

[40] W. Jin, A. Aydinoglu, M. Halm, and M. Posa, "Learning linear complementarity systems," in *Learning for Dynamics and Control Conference*, pp. 1137–1149, PMLR, 2022.

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

[42] A. Aydinoglu, P. Sieg, V. M. Preciado, and M. Posa, "Stabilization of complementarity systems via contact-aware controllers," *arXiv preprint arXiv:2008.02104*, 2020.

[43] J.-S. Pang and D. E. Stewart, "Differential variational inequalities," *Mathematical programming*, vol. 113, no. 2, pp. 345–424, 2008.

[44] M. K. Camlibel, J.-S. Pang, and J. Shen, "Lyapunov stability of complementarity and extended systems," *SIAM Journal on Optimization*, vol. 17, no. 4, pp. 1056–1101, 2007.

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

[46] 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.

[47] B. Brogliato and B. Brogliato, *Nonsmooth mechanics*, vol. 3. Springer, 1999.

[48] M. Anitescu, "Optimization-based simulation of nonsmooth rigid multibody dynamics," *Mathematical Programming*, vol. 105, pp. 113–143, 2006.

[49] A. Cauligi, P. Culbertson, B. Stellato, D. Bertsimas, M. Schwager, and M. Pavone, "Learning mixed-integer convex optimization strategies for robot planning and control," in *2020 59th IEEE Conference on Decision and Control (CDC)*, pp. 1698–1705, IEEE, 2020.

[50] A. Aydinoglu, M. Fazlyab, M. Morari, and M. Posa, "Stability analysis of complementarity systems with neural network controllers," in *Proceedings of the 24th International Conference on Hybrid Systems: Computation and Control*, pp. 1–10, 2021.

[51] S. Diamond, R. Takapoui, and S. Boyd, "A general system for heuristic minimization of convex functions over non-convex sets," *Optimization Methods and Software*, vol. 33, no. 1, pp. 165–193, 2018.

[52] 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.

[53] Gurobi Optimization, LLC, "Gurobi Optimizer Reference Manual," 2021.

[54] S. P. Dirkse and M. C. Ferris, "The path solver: a nommonotone stabilization scheme for mixed complementarity problems," *Optimization methods and software*, vol. 5, no. 2, pp. 123–156, 1995.

[55] F. R. Hogan, J. Ballester, S. Dong, and A. Rodriguez, "Tactile dexterity: Manipulation primitives with tactile feedback," in *2020 IEEE international conference on robotics and automation (ICRA)*, pp. 8863–8869, IEEE, 2020.

[56] H. Li, R. J. Frei, and P. M. Wensing, "Model hierarchy predictive control of robotic systems," *IEEE Robotics and Automation Letters*, vol. 6, no. 2, pp. 3373–3380, 2021.

[57] R. Tedrake and the Drake Development Team, "Drake: Model-based design and verification for robotics," 2019.

[58] V. Kurtz and H. Lin, "Contact-implicit trajectory optimization with hydroelastic contact and ilqr," *arXiv preprint arXiv:2202.13986*, 2022.

[59] G. Bradski, "The OpenCV Library," *Dr. Dobb's Journal of Software Tools*, 2000.

[60] R. O. Duda and P. E. Hart, "Use of the hough transformation to detect lines and curves in pictures," *Commun. ACM*, vol. 15, p. 11–15, jan 1972.

[61] S. Pfrommer, M. Halm, and M. Posa, "Contactnets: Learning discontinuous contact dynamics with smooth, implicit representations," in *Conference on Robot Learning*, pp. 2279–2291, PMLR, 2021.

[62] S. Kim, D. K. Jha, D. Romeres, P. Patre, and A. Rodriguez, "Simultaneous tactile estimation and control of extrinsic contact," *arXiv preprint arXiv:2303.03385*, 2023.

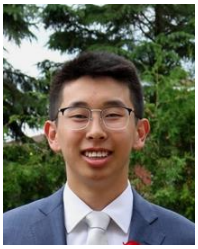
[63] A. Nedić and A. Ozdaglar, "Approximate primal solutions and rate analysis for dual subgradient methods," *SIAM Journal on Optimization*, vol. 19, no. 4, pp. 1757–1780, 2009.

[64] Y.-M. Chen and M. Posa, "Optimal reduced-order modeling of bipedal locomotion," in *2020 IEEE International Conference on Robotics and Automation (ICRA)*, pp. 8753–8760, IEEE, 2020.

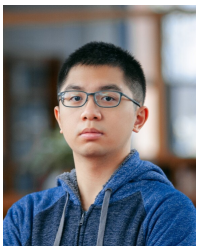
[65] N. Hogan, "Impedance control: An approach to manipulation: Part ii—implementation," 1985.

[66] J. Hermus, J. Lachner, D. Verdi, and N. Hogan, "Exploiting redundancy to facilitate physical interaction," *IEEE Transactions on Robotics*, vol. 38, no. 1, pp. 599–615, 2021.

Alp Aydinoglu completed his B.S. in Control Engineering from Istanbul Technical University in 2017 and is currently pursuing his Ph.D. in Electrical and Systems Engineering at University of Pennsylvania, working with Michael Posa in Dynamic Autonomy and Intelligent Robotics (DAIR) Lab. His research emphasizes control of multi-contact systems.



Adam Wei received his B.ASc. in electrical engineering from the University of Toronto in 2019. He is currently pursuing a Ph.D. in robotics at the Massachussets Institute of Technology where he is advised by Prof. Russ Tedrake. His research interests include controls and imitation learning for robotic manipulation.



Wei-Cheng Huang received his B.E. degree in Mechanical Engineering from Shanghai Jiao Tong University, Shanghai, China, in 2022, and his M.S.E. degree in Robotics from the General Robotics, Automation, Sensing and Perception (GRASP) Lab, University of Pennsylvania, Philadelphia, PA, USA, in 2024. He is currently pursuing his Ph.D. degree in Computer Science, advised by Professor Kris Hauser, at the University of Illinois Urbana-Champaign, Urbana, IL, USA. His research interests primarily focus on planning and control for contact-rich dexterous robotic manipulation.

Michael Posa received the B.S. and M.S. degrees in mechanical engineering from Stanford University, Stanford, CA, USA, in 2007 and 2008, respectively, and 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 the 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.