Fast Contact-Implicit Model Predictive Control

Simon Le Cleac'h[1*], Taylor A. Howell[1*], Shuo Yang[2], Chi-Yen Lee[2],
John Zhang[2], Arun Bishop[2], Mac Schwager[3], and Zachary Manchester[2]

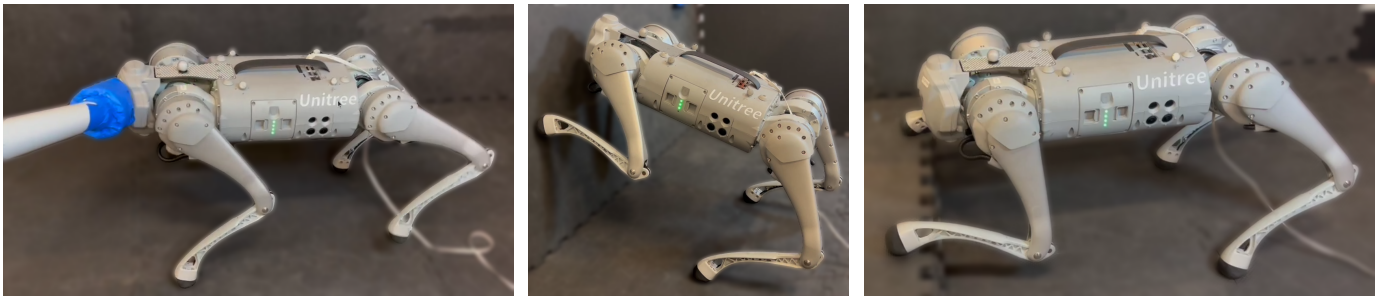


Fig. 1: Hardware demonstrations on a Unitree Go1 quadruped: stable trotting while being pushed (left), transitioning from ground to standing against a wall (center), and placing two feet onto a step (right).

***Abstract*—We present a general approach for controlling robotic systems that make and break contact with their environments. Contact-implicit model predictive control (CI-MPC) generalizes linear MPC to contact-rich settings by utilizing a bi-level planning formulation with lower-level contact dynamics formulated as time-varying linear complementarity problems (LCPs) computed using strategic Taylor approximations about a reference trajectory. These dynamics enable the upper-level planning problem to reason about contact timing and forces, and generate entirely new contact-mode sequences online. To achieve reliable and fast numerical convergence, we devise a structure-exploiting interior-point solver for these LCP contact dynamics and a custom trajectory optimizer for the tracking problem. We demonstrate real-time solution rates for CI-MPC and the ability to generate and track non-periodic behaviours in hardware experiments on a quadrupedal robot. We also show that the controller is robust to model mismatch and can respond to disturbances by discovering and exploiting new contact modes across a variety of robotic systems in simulation, including a pushbot, planar hopper, planar quadruped, and planar biped.**

***Index Terms*—Model Predictive Control, Legged Robots, Contact Modeling, Optimization and Optimal Control.**

I. Introduction

CONTROLLING systems that make and break contact with their environments is one of the grand challenges in robotics. Numerous approaches have been employed for controlling such systems, ranging from hybrid-zero dynamics [1, 2, 3], to complementarity controllers [4], to neural-network policies [5, 6], and model predictive control (MPC) [7, 8]. There have also been numerous successes deploying such approaches on complex systems in recent years: direct trajectory optimization and LQR on Atlas [9], smooth-contact models and differential dynamic programming on HRP-2 [10, 11, 12], zero-moment point and feedback linearization on ASIMO [13], and MPC with simplified dynamics models on Cheetah [14] and ANYmal [15]. However, reliable *general-purpose* control techniques that can reason about contact events and can be applied across a wide range of robotic systems without requiring application-specific model simplifications, gait-generation heuristics, or extensive parameter tuning remain elusive.

Our approach combines fast, differentiable rigid-body dynamics with contact, strategic approximations about a reference trajectory, and specialized numerical optimization techniques for the application of local tracking control for systems that experience contact interactions with their environments. The result is a bi-level model predictive control algorithm that can effectively reason about contact changes in the presence of large disturbances while remaining fast enough for real-time execution.

We formulate contact dynamics as a complementarity problem and devise a fast interior-point solver to reliably optimize this feasibility problem. Smooth gradients are efficiently computed through the non-smooth dynamics by exploiting intermediate solutions from within this solver using implicit differentiation. To enable real-time performance for control, we pre-compute linearizations of the system's dynamics, signed-distance functions, and friction cones about a reference trajectory, while explicitly retaining complementarity constraints that encode contact switching behavior, resulting in a sequence of lower-level time-varying linear-complementarity problems (LCP) which represent the model's contact dynamics. An upper-level trajectory optimization problem is then optimized using fast linear algebra. We refer to this algorithm as *Contact-*

[1] Simon Le Cleac'h and Taylor A. Howell are with the Department of Mechanical Engineering, Stanford University, Stanford, CA 94305, USA. {simonlc, thowell}@stanford.edu

[2] Shuo Yang, Chi-Yen Lee, John Zhang, Arun Bishop, and Zachary Manchester are with The Robotics Institute, Carnegie Mellon University, Pittsburgh, PA 15213, USA. {shuoyang, chiyenl, ziyangz3, arunleob, zacm}@andrew.cmu.edu

[3] Mac Schwager is with the Department of Aeronautics and Astronautics, Stanford University, Stanford, CA 94305, USA. schwager@stanford.edu

(Corresponding authors: Simon Le Cleac'h and Taylor Howell)

[*] These authors contributed equally to this work.

Implicit Model Predictive Control (CI-MPC).

Finally, we demonstrate that CI-MPC can generate new contact sequences online and reliably track reference trajectories despite significant model mismatch and while large external disturbances are applied for a number of qualitatively different robotic systems, including: a pushbot, and planar hopper, quadruped, and biped systems in simulation; and on Unitree Go1 quadruped hardware [16].

Our contributions are:

- Fast approximate contact dynamics that can be reliably evaluated and efficiently differentiated with a custom interior-point solver
- Structure-exploiting solvers for the contact-dynamics and trajectory optimization problems
- A model predictive control framework for robotic systems with contact dynamics
- A collection of simulation and hardware experiments demonstrating the performance of CI-MPC on a variety of robotic systems across a range of highly dynamic tasks

In the remainder of this paper, we first review related work on control through contact with MPC, as well as complementarity-based contact dynamics in Section II. Next, we present a brief overview of MPC, outline the classic complementarity formulation for contact dynamics, and provide background on interior-point methods and implicit differentiation in Section III. Then, we present CI-MPC in Section IV. Results are presented in Section V including both simulation and hardware experiments. Finally, we discuss our results, limitations of this approach, potential directions for future work in Section VI.

II. Related Work

In this section, we review related work on MPC for the control of dynamical systems that make and break contact with their environments and provide an overview of complementarity-based contact dynamics.

A. Model Predictive Control

Today, most successful approaches for controlling legged robots utilize MPC in combination with simplified models and heuristics originally pioneered by Raibert for hopping robots [17]. The key insight of this work is that the control problem can be decoupled into a high-level controller that plans body motions while ignoring the details of the leg dynamics, and a low-level controller that determines the necessary leg motions and joint torques to generate the forces and torques on the body determined by the high-level controller.

Arguably the most impressive control work on humanoids has utilized centroidal dynamics with full kinematics to enable Atlas to navigate various scenarios with obstacles [18] and perform parkour [19]. Integrating hardware design and controller synthesis has also recently enabled small humanoids to perform agile acrobatic maneuvers in simulation [20].

There have also been impressive advances for quadrupeds, achieved by designing hardware that aims to closely match the modeling approximations made in the controller, e.g., building very light legs [14]. Whole-body control, which has the benefit of simpler overall control structures and the ability to leverage a system's dynamics, has been achieved at real-time rates on hardware [21]. Approaches that utilize both force-based MPC and whole-body control have also demonstrated agile locomotion [22].

A major limitation of these prior works is that the control policies are highly specialized to a specific robotic system. In this work, we compare CI-MPC to a number of system-specific control methods that perform quite well for their given system, but do not generalize to other systems, whereas our policy generalizes to many different systems that experience contact interactions while achieving comparable or better performance.

B. Complementarity-Based Contact Dynamics

The classic approach for simulating rigid-body dynamics with contact interactions is a velocity-based time-stepping scheme formulated as a linear complementarity problem (LCP) [23, 24, 25]. The LCP searches for the next state of the system while enforcing impact and friction constraints. Solvers for this class of problems utilize pivoting methods [26], such as Lemke's algorithm [27], or interior-point methods [28]. Implementations of pivoting methods can be found in general-purpose LCP solvers such as PATH [29], or physics engines including: Bullet [23] and DART [24].

Derivatives of LCP-based contact dynamics can be efficiently computing using implicit differentiation [30]. However, the quality of these results is dependent on the method employed for optimization. Pivoting approaches enforce strict complementarity at each iteration, returning solutions at non-differentiable points. As a consequence, this differentiation will return subgradients that make, typically efficient, second-order optimization slower and less reliable. In contrast, interior-point methods relax the complementarity constraints at each iteration, only converging in the limit. These intermediate results can be implicitly differentiated to return smooth gradients [31]. Alternative approaches for computing gradients for contact dynamics include utilizing auto-differentiation tools [6] and analyzing the LCP solution to select subgradients [32].

In addition to simulation, contact dynamics represented as LCPs have been utilized for planning. Collocation approaches [33] directly encode the LCP problem as as constraints in order to enforce contact dynamics, along with an objective specifying desired behavior, in a large non-convex problem [34]. This approach enables the optimizer to plan without pre-specified mode sequences for locomotion and simple manipulation tasks. Subsequent work improved this approach by introducing higher-order integrators for the dynamics and a numerically robust, exact ℓ_1-penalty for handling the complementarity constraints [35]. Alternative rollout-based approaches utilize LCPs for forward simulation and subsequently differentiate through the solution of one-step dynamics in order to compute derivatives for gradient-based optimization [36, 32].

Another popular contact-dynamics formulation is MuJoCo's soft-contact model [10], which solves a convex optimization problem and trades physical realism for fast and reliable performance. MuJoCo's gradients are computed using a finite-difference scheme since the collision detection routine is

not differentiable. However, this approach is computationally less efficient. An alternative model solved a strictly convex quadratic program [37]. This method is amenable to implicit differentiation, however it approximates the contact model. Finally, the LCP complementarity constraints can be relaxed, resulting in a soft-contact model that exhibits improved numerical properties in some scenarios [38].

III. Background

In this section, we provide technical background on MPC, complementarity-based contact dynamics, interior-point methods, and implicit differentiation.

A. Model Predictive Control

Predictive control policies [39] optimize a planning problem:

$$\begin{array}{ll} \underset{x_{1:T}, u_{1:T-1}}{\text{minimize}} & g_T(x_T) + \sum_{t=1}^{T-1} g_t(x_t, u_t) \\ \text{subject to} & x_{t+1} = f_t(x_t, u_t), \quad t = 1, \ldots, T-1, \\ & (x_1 \text{ given}), \end{array} \tag{1}$$

for a given initial state in order to compute controls for a dynamical system we aim to control. If planning is performed at a sufficiently high rate, the sequence of open-loop plans provide *feedback*. For a system with state $x \in \mathbf{R}^n$, control $u \in \mathbf{R}^m$, time index t, initial state x_1, and discrete-time dynamics $f : \mathbf{R}^n \times \mathbf{R}^m \to \mathbf{R}^n$, the optimizer aims to minimize an objective with costs, $g : \mathbf{R}^n \times \mathbf{R}^m \to \mathbf{R}$, over a planning horizon T.

Solving a (potentially) non-convex problem (1) online can be unreliable or computationally too expensive. Instead, a proxy problem is solved online that makes strategic approximations. A common simplification is to track a reference trajectory, $\bar{\tau} = (\bar{x}_1, \bar{u}_1, \ldots, \bar{x}_T)$, denoted with an overbar ($\bar{}$), that is precomputed offline. In this setting, the computational complexity for planning is reduced by utilizing dynamics:

$$\delta x_{t+1} = A_t \delta x_t + B_t \delta u_t, \tag{2}$$

linearized about the reference, where $A = \partial f(\bar{x}, \bar{u})/\partial x$, $B = \partial f(\bar{x}, \bar{u})/\partial x$, and the decision variables are relative to the reference trajectory, i.e., $\delta a = a - \bar{a}$; and a quadratic objective:

$$\frac{1}{2}\delta x_t^T Q_t \delta x_t + q_t^T \delta x_t + \frac{1}{2}\delta u_t^T R_t \delta u_t + r_t^T \delta u_t, \tag{3}$$

where $Q = \partial^2 g(\bar{x}, \bar{u})/\partial x^2, q = \partial g(\bar{x}, \bar{u})/\partial x, R = \partial^2 g(\bar{x}, \bar{u})/\partial u^2, r = \partial g(\bar{x}, \bar{u})/\partial u$, similarly comprise a second-order expansion about the reference trajectory.

This formulation, potentially with additional affine state and control constraints, is commonly referred to as *linear* MPC. Without such constraints, the problem (1) with linear dynamics (2) and quadratic objective (3) is the LQR problem [40] and is efficiently solved with a backward Riccati recursion.

Predictive control iteratively optimizes (1), or an approximate version of it (2 - 3), for a given state, and the optimized controls are utilized to compute an input for the system. After the system evolves, the problem is re-optimized for a new state in order to compute a new control for the system. By repeating this procedure at a high rate, feedback control is achieved [41]. In practice, applying the controls optimized with time-varying linearized dynamics and quadratic costs, to the actual nonlinear system is extremely effective, especially for applications that track a reference trajectory.

B. Complementarity-Based Contact Dynamics

Contact dynamics can be simulated with a velocity time-stepping scheme [42], formulated as complementarity problem [27]:

$$\begin{array}{rl} \text{find} & q, \gamma, \beta, \psi \\ \text{subject to} & [M_+(q - q_-) - M_-(q_- - q_{--})]/h + hC = \\ & \quad J^T(\gamma, \beta) + Bu, \\ & \gamma \circ \phi = 0, \\ & \beta \circ [P(q - q_-)/h + \psi \mathbf{1}] = 0, \\ & \psi \circ \left[\mu\gamma - \mathbf{1}^T\beta\right] = 0, \\ & \phi, \gamma \geq 0, \\ & \beta, \psi, [P(q - q_-)/h + \psi\mathbf{1}], [\mu\gamma - \mathbf{1}^T\beta] \geq 0, \end{array} \tag{4}$$

that finds the next configuration of the system $q \in \mathbf{R}^{n_q}$ using implicitly defined velocities, $v = (q - q_-)/h \in \mathbf{R}^{n_v}$. Subscripts indicate a previous time step. This formulation considers a single contact point, but generalizes to systems with multiple contacts. The problem utilizes: the mass matrix $M : \mathbf{R}^{n_q} \to \mathbf{S}^{n_v}_{++}$; dynamics bias $C : \mathbf{R}^{n_q} \times \mathbf{R}^{n_v} \to \mathbf{R}^{n_v}$ that includes Coriolis and gravitational terms; contact Jacobian $J : \mathbf{R}^{n_q} \to \mathbf{R}^{(p(d-1)+1) \times n_v}$ that maps contact forces in the contact frame into the generalized coordinates; input Jacobian $B : \mathbf{R}^{n_q} \to \mathbf{R}^{n_v \times m}$ that maps control inputs, typically joint torques, into the generalized coordinates; $P : \mathbf{R}^{n_q} \to \mathbf{R}^{p(d-1) \times n_v}$ is a mapping from the generalized velocity space to an overparameterized contact tangent space; time step $h \in \mathbf{R}_+$; with normal force $\gamma \in \mathbf{R}$ and overparameterized friction forces $\beta \in \mathbf{R}^{p(d-1)}$ that are constrained by a linearized friction cone; $\psi \in \mathbf{R}$ is a dual variable associated with friction, representing the magnitude of the contact point velocity; signed-distance function, $\phi : \mathbf{R}^{n_q} \to \mathbf{R}$, that returns distance between a specified contact point on the robot (e.g., feet) and the closest surface in the environment (e.g., the floor); and where $\circ$ is an element-wise (Hadamard) vector product. We use p to denote the overparameterization dimension (typically p = 2) and d to denote the environment dimension $d = 2$ for planar systems and $d = 3$ otherwise. This formulation extends to multiple contact points.

The smooth dynamics (5) are discretized with a semi-implicit Euler scheme [43]; complementarity constraints (6- 8) encode contact switching behavior; impact is encoded in (6) and (9); and friction terms (7-8) and (10), are derived from the maximum dissipation principle [44].

Problem data include previous configurations q_-, q_{--}, time step h and control inputs u. A nonlinear formulation uses the

following mappings:

$$\begin{aligned} &M_+ \leftarrow M(q), && M_- \leftarrow M(q_-), \\ &C \leftarrow C(q, (q-q_-)/h), && J \leftarrow J(q), \\ &B \leftarrow B(q), && P \leftarrow P(q), \\ &\phi \leftarrow \phi(q), \end{aligned} \tag{11}$$

evaluating these terms at the next configuration. In practice, a partial linearization of the dynamics is performed to satisfy a linear complementarity problem (LCP) [42], resulting in the following mappings:

$$\begin{aligned} &M_+ \leftarrow M(q_-), && M_- \leftarrow M(q_{--}), \\ &C \leftarrow C(q_-, (q_- - q_{--})/h), && J \leftarrow J(q_-), \\ &B \leftarrow B(q_-), && P \leftarrow P(q_-), \\ &\phi \leftarrow \phi(q_-) + N(q-)(q-q_-), \end{aligned} \tag{12}$$

where $N = d\phi/dq$.

C. Interior-Point Method

Classically, LCPs are solved using active-set methods [29], which strictly enforce complementarity at each iteration. An alternative approach is interior-point methods [28, 45], which relax these conditions during intermediate iterations, only satisfying these constraints in the limit.

LCPs can be generally formulated as:

$$\begin{aligned} \text{find} \quad & x, y, z \\ \text{subject to} \quad & Ex + Fy + f = 0, \\ & Gx + Hy + z + h = 0, \\ & y \circ z = 0, \\ & y, z \geq 0, \end{aligned} \tag{13}$$

with decision variables $x \in \mathbf{R}^n$, $y, z \in \mathbf{R}^m$ and problem data $\theta = (E, F, G, H, f, h) \in \mathbf{R}^{n\times n} \times \mathbf{R}^{n\times m} \times \mathbf{R}^{m\times n} \times \mathbf{R}^{m\times m} \times \mathbf{R}^n \times \mathbf{R}^m = \mathbf{R}^p$. Interior-point methods parameterize (13) by a central-path parameter $\kappa \in \mathbf{R}_+$ that relaxes the following bilinear constraint:

$$y \circ z = \kappa \mathbf{1}, \tag{14}$$

where $\mathbf{1}$ a vector of ones.

The equality and relaxed bilinear constraints form a residual vector or solution map, $r : \mathbf{R}^{n+2m} \times \mathbf{R}^p \times \mathbf{R}_+ \rightarrow \mathbf{R}^{n+2m}$, that takes $w = (x, y, z) \in \mathbf{R}^{n+2m}$, the problem data, and central-path parameter as inputs. The problem data and central-path parameter are fixed during optimization. In the context of contact dynamics, these data encode the mechanical properties of the robots, its current configuration and velocity, and properties of the environment like friction coefficients. Newton or quasi-Newton methods are used to find search directions that reduce the norm of the residual and a backtracking line search is employed to ensure that the inequality constraints are strictly satisfied for candidate points at each iteration. Once the residual is optimized to a desired tolerance, the central-path parameter is decreased and the new subproblem is warm-started with the current solution and then optimized. This procedure is repeated in order to find solutions to (13) with $\kappa \rightarrow 0$ until the central-path parameter, also referred to as complementary slackness, is below a desired tolerance.

Importantly, our interior-point method utilizes a predictor-corrector algorithm [46] that leads to significantly improved convergence. First, the corrector step modifies the pure Newton search direction and typically reduces the number of iterations required for convergence by half (compared to the pure search direction). Second, the central-path parameter is adapted at each iteration to prevent premature numerical ill-conditioning. In practice, we find that this approach is significantly more reliable and has improved convergence behavior compared to prior work that employed relaxed complementarity conditions [35].

D. Implicit Differentiation

In addition to simulating contact by solving a feasibility problem, we would like to compute gradients of these dynamics, requiring us to differentiate through an optimization problem. This is accomplished with implicit differentiation [30].

An implicit function, $r : \mathbf{R}^k \times \mathbf{R}^p \rightarrow \mathbf{R}^k$, is defined such that:

$$r(w^*; \theta) = 0, \tag{15}$$

for solutions $w^* \in \mathbf{R}^k$ and problem data $\theta \in \mathbf{R}^p$. At a stationary point, $w^*(\theta)$, the sensitivity of the solution with respect to the problem data, i.e., $\frac{\partial w^*}{\partial \theta}$, can be computed by utilizing the implicit-function theorem [30]. We expand (15) to first order:

$$\frac{\partial r}{\partial w}\delta w + \frac{\partial r}{\partial \theta}\delta\theta = 0, \tag{16}$$

and then solve for δw:

$$\frac{\partial w^*}{\partial \theta} = -\Big(\frac{\partial r}{\partial w}\Big)^{-1}\frac{\partial r}{\partial \theta}, \tag{17}$$

to compute the sensitivities. For the interior-point method (13), the residual is:

$$r(w; \theta) = \begin{bmatrix} Ex + Fy + f \\ Gx + Hy + z + h \\ y \circ z - \kappa \mathbf{1} \end{bmatrix}. \tag{18}$$

A differentiable interior-point method is summarized in Algorithm 1. Importantly, we can compute gradients for intermediate results, corresponding to non-zero values for the central-path parameter, i.e., $\kappa_{\text{grad}} \neq 0$. For additional details about differentiating through intermediate results of contact dynamics that are solved with interior-point methods, see [31].

IV. Contact-Implicit Model Predictive Control

In this section we present Contact-Implicit Model Predictive Control, a tracking policy for systems that make and break contact with their environments. First, we formulate time-varying LCP contact dynamics that are selectively approximated about a reference trajectory. Then, we devise a fast solver for the resulting LCP. Next, we discuss how to compute smooth gradients through the dynamics. A bi-level planning formulation, which utilizes these dynamics for direct trajectory optimization [47], follows. To enable the policy to work well in environments with uncertain terrain we propose a contact-height heuristic. Finally, we summarize the approach and provide an algorithm for CI-MPC.

Algorithm 1 Differentiable Interior-Point Method

procedure OPTIMIZE(x, θ)
  Settings: $\beta = 0.5, \gamma = 0.1, \epsilon_\kappa = 10^{-6}, \epsilon_r = 10^{-8}$
  Initialize: $y, z = \mathbf{1}, \kappa = 0.1, \kappa_{\text{grad}} = 10^{-4}$
  Until $\kappa < \epsilon_\kappa$ **do**
    $\Delta w = (\frac{\partial r}{\partial w})^{-1} r(w; \theta, \kappa)$
    $\alpha \leftarrow 1$
    Until $(y, z) - \alpha(\Delta y, \Delta z) > 0$ **do** $\alpha \leftarrow \beta\alpha$
    Until $\|r(w - \alpha\Delta w; \theta, \kappa)\| < \|r(w; \theta, \kappa)\|$ **do**
      $\alpha \leftarrow \beta\alpha$
    $w \leftarrow w - \alpha\Delta w$
    If $\|r(w; \theta, \kappa)\| < \epsilon_r$ **do** $\kappa \leftarrow \gamma\kappa$
  $\frac{\partial w}{\partial \theta} \leftarrow$ **Differentiate**$(w, \theta, \kappa_{\text{grad}})$ ▷ Eq. 17
  Return w, $\frac{\partial w}{\partial \theta}$

A. Time-Varying Contact Dynamics

We formulate alternative LCP dynamics that utilize a reference trajectory, resulting in the following mappings:

$$\begin{aligned} &M_+ \leftarrow M(\bar{q}), && M_- \leftarrow M(\bar{q}_-), \\ &C \leftarrow C(\bar{q}, (\bar{q} - \bar{q}_-)/h), && J \leftarrow J(\bar{q}), \\ &B \leftarrow B(\bar{q}), && P \leftarrow P(\bar{q}), \\ &\phi \leftarrow \phi(\bar{q}) + N(\bar{q})(q - \bar{q}). \end{aligned} \tag{19}$$

The LCP problem, formulated for an interior-point method, has the form:

$$\begin{aligned} \text{find} \quad & w \\ \text{subject to} \quad & C(w - \bar{w}) + D(\theta - \bar{\theta}) = 0 \\ & \gamma \circ s_\phi = \kappa\mathbf{1}, \\ & \psi \circ s_\psi = \kappa\mathbf{1}, \\ & \beta \circ \eta = \kappa\mathbf{1}, \\ & \gamma, \psi, \beta, \eta, s_\phi, s_\psi \geq 0, \end{aligned} \tag{20}$$

with decision variables $w = (q, \gamma, \psi, \beta, \eta, s_\phi, s_\psi)$, where slack variables, $s_\phi, s_\psi \in \mathbf{R}$, are introduced for convenience. For the LCP formulation $\kappa = 0$, while the interior-point method's subproblems are specified by $\kappa > 0$.

Importantly, C and D are matrices that define a linear system of equations resulting from approximations about the reference trajectory and they are pre-computed offline. These contact dynamics:

$$q_{t+1} = \mathbf{LCP}_t(q_{t-1}, q_t, u_t), \tag{21}$$

$\mathbf{LCP}_t : \mathbf{R}^{n_q} \times \mathbf{R}^{n_q} \times \mathbf{R}^m \to \mathbf{R}^{n_q}$, solve (20) and return the configuration at the next time step. The contact forces at the current time step can also be returned.

B. Fast Contact Dynamics

The most expensive procedure in evaluating the LCP and computing gradients of a solution is solving the linear system of equations:

$$R_w \Delta w = r, \tag{22}$$

required by the interior-point method, where $R_w = \partial r / \partial w$, and Δw is the new search direction.

To reduce the computational cost of this routine, we exploit both the sparsity pattern and the property that most of R_w remains constant across iterations and, therefore, can be pre-factorized offline [48].

We partition the LCP variables (13) as follows: $x = q$, $y = (\gamma, \psi, \beta)$, and $z = (\eta, s_\phi, s_\psi)$, and similarly split the residual: $r = (r_x, r_y, r_z)$. The Jacobian's sparsity pattern is:

$$R_w = \begin{bmatrix} E & F & 0 \\ G & H & I \\ 0 & \mathbf{diag}(z) & \mathbf{diag}(y) \end{bmatrix}, \tag{23}$$

where I denotes the identity matrix. By exploiting sparsity in the third row of (23), we can form the following condensed system:

$$\begin{bmatrix} E & F \\ G & \tilde{H} \end{bmatrix} \begin{bmatrix} \Delta x \\ \Delta y \end{bmatrix} = \begin{bmatrix} r_x \\ \tilde{r}_y \end{bmatrix} \Leftrightarrow \tilde{R}_w \, \Delta\tilde{w} = \tilde{r}, \tag{24}$$

where:

$$\tilde{H} = H - \mathbf{diag}(y^{-1} \circ z), \tag{25}$$

$$\tilde{r}_y = r_y - y^{-1} \circ r_z, \tag{26}$$

$$\Delta z = y^{-1} \circ (r_z - z \circ \Delta y), \tag{27}$$

and y^{-1} denotes the element-wise reciprocal of vector y. This term is always well-defined because a line search enforces $y > 0$ at each iteration.

To solve for $\Delta\tilde{w}$, we leverage the fact that, apart from the bottom-right block, $\tilde{R}_w$ can be computed offline. We perform a QR decomposition on the Schur complement of (24):

$$Q, R \leftarrow \mathbf{QR}(\tilde{H} - GE^{-1}F), \tag{28}$$

and then solve for the search directions:

$$\Delta y = -R^{-1}Q^T(GE^{-1}r_x - \tilde{r}_y), \tag{29}$$

$$\Delta x = E^{-1}(r_x - F\Delta y). \tag{30}$$

Additionally, E^{-1}, GE^{-1}, and $GE^{-1}F$ are precomputed offline. Finally, after solving for $\Delta\tilde{w}$, we obtain Δz with cheap vector-vector operations (27).

For a system with configuration dimension n_q and c contact points, the computational complexity of solving (22) with a naive approach is $O\Big((n_q + 2cd)^3\Big)$. Our structure-exploiting approach is $O(8c^3d^3)$ during the online phase. In practice, this provides a factor of 15 speed-up, compared to LAPACK LU, for evaluating the LCP dynamics across all the robotic systems presented in this paper and, in turn, results in a factor of 2.5 speed-up for CI-MPC.

C. Gradients

Contact dynamics gradients are computed by differentiating through the LCP problem with respect to data: $\theta = (q_{--}, q_-, u)$, which could also include the time step, friction coefficients, and other system values like masses or inertia terms.

At fixed points, w^*_κ, parameterized by a (potentially non-zero) central-path value, gradients are computed using implicit-differentiation. Importantly, at solution where $\kappa \approx 0$, this approach will return subgradients, which often fail to provide useful information through contact events. Indeed,

Algorithm 2 Contact-Implicit Model Predictive Control

procedure POLICY
  Offline
    $\bar{\tau} \leftarrow$ generate reference trajectory
    $\mathbf{LCP}_t \leftarrow$ generate fast contact dynamics
    $H \leftarrow$ set planning horizon,
  Online
    For $i = 1, \ldots, \infty$
      $u \leftarrow \pi(x)$ ▷ Eq. (31)
      $x \leftarrow \mathbf{dynamics}(x, u)$
    End
End

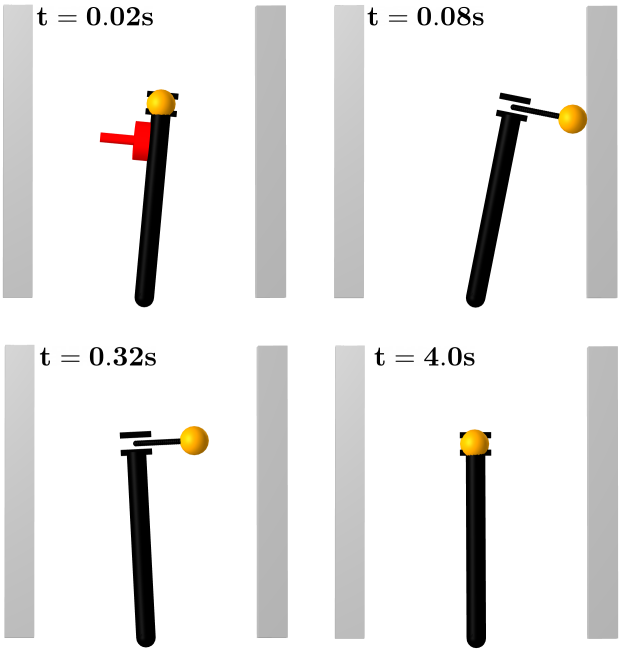


Fig. 2: PushBot performing push recovery. A disturbance (red) creates an impulse on the system and the policy generates a new contact sequence that extends the prismatic joint toward the right wall in order to make contact. After stabilizing, PushBot pushes against the wall, eventually breaking contact, in order to return to the nominal upright configuration.

these gradients lacks information about nearby potential contact events (contact $\leftrightarrow$ no contact) and nearby contact mode switches (sticking $\leftrightarrow$ sliding). However, we exploit intermediate results from the interior-point solver, where $\kappa \neq 0$, in order to compute smooth gradients. Large values of κ will produce smoother gradients than those computed with small values of κ, which more closely approximates a true subgradient at nondifferentiable points. Practically, we find that smooth gradients, computed using intermediate results, provide information through contact events.

TABLE I: Comparison between CI-MPC and MIQP policies for PushBot example. For a fixed replanning rate of 25 Hz, we report the mean and standard deviations for the optimization times and compare this to the associated time budget (0.04 s). Both policies successfully regulate the system around the equilibrium point. However, the MIQP policy is slower than real-time, whereas the CI-MPC policy always remains within time budget, ensuring real-time performance.

Policy	Planning Time	Real-Time
CI-MPC	$\mathbf{0.014 \pm 0.027s}$	✓
MIQP	0.18 ± 0.09 s	×

D. Planning

The CI-MPC policy is formulated as:

$$u = \pi(x) \begin{cases} \underset{x_{1:H}, u_{1:H}}{\text{minimize}} & \sum_{t=1}^{H} \frac{1}{2}(x_t - \bar{x}_t)^T Q_t (x_t - \bar{x}_t) \\ & + \frac{1}{2}(u_t - \bar{u}_t)^T R_t (u_t - \bar{u}_t) \\ \text{subject to} & x_{t+1} = \mathbf{LCP}_t(x_t, u_t), \\ & \quad t = 1, \ldots, H-1, \\ & x_1 = x, \end{cases} \tag{31}$$

comprising an upper-level planning problem (1) that optimizes a trajectory: $\tau = (x_1, u_1, \ldots, x_H, u_H) \in \mathbf{R}^{(2n_q+m)H}$ of configurations and controls over a horizon H using a state representation: $x_t = (q_{t-1}^{(t)}, q_t^{(t)})$, with two configurations. This problem is solved using a direct trajectory optimization approach. Lower-level LCP problems (4) enforce the dynamics with the first state x_1 fixed. For convenience, we overload notation for the LCP dynamics:

$$\mathbf{LCP}_t(x_t, u_t) = \begin{bmatrix} q_t^{(t)} \\ \mathbf{LCP}_t(q_{t-1}^{(t)}, q_t^{(t)}, u_t) \end{bmatrix}, \tag{32}$$

for state-based LCP dynamics, and define constraints, $k_t(x_t, u_t, x_{t+1}) = x_{t+1} - \mathbf{LCP}_t(x_t, u_t)$, that couple states across adjacent time steps. The constraint Jacobian:

$$\nabla k = \begin{bmatrix} -B_1 & I & 0 & 0 & 0 & 0 & \\ 0 & -A_2 & -B_2 & I & 0 & 0 & \\ 0 & 0 & 0 & -A_3 & -B_3 & I & \\ & & & & & & \ddots \end{bmatrix}, \tag{33}$$

where $k = (k_1, \cdots, k_{H-1}) \in \mathbf{R}^{2n_q(H-1)}$, is comprised of one-step dynamics Jacobians:

$$A_t = \begin{bmatrix} 0 & I \\ \frac{\partial \mathbf{LCP}_t}{\partial q_{t-1}} & \frac{\partial \mathbf{LCP}_t}{\partial q_t} \end{bmatrix}, B_t = \begin{bmatrix} 0 \\ \frac{\partial \mathbf{LCP}_t}{\partial u_t} \end{bmatrix}. \tag{34}$$

The planning objective is a convex quadratic function (3) and velocities are penalized using finite-difference approximations. Because the problem is lifted by using states comprising two configurations, these costs do not introduce coupling

across more than one time step. The resulting Hessian of the objective:

$$\nabla^2 J = \begin{bmatrix} R_1 & 0 & 0 & 0 \\ 0 & Q_2 & 0 & 0 \\ 0 & 0 & R_2 & 0 \\ 0 & 0 & 0 & \ddots \end{bmatrix}, \tag{35}$$

and its inverse:

$$(\nabla^2 J)^{-1} = \begin{bmatrix} R_1^{-1} & 0 & 0 & 0 \\ 0 & Q_2^{-1} & 0 & 0 \\ 0 & 0 & R_2^{-1} & 0 \\ 0 & 0 & 0 & \ddots \end{bmatrix}, \tag{36}$$

are block diagonal and are pre-computed offline.

The resulting KKT system:

$$\begin{bmatrix} \nabla^2 J & \nabla k^T \\ \nabla k & 0 \end{bmatrix} \begin{bmatrix} \Delta\tau \\ \Delta\nu \end{bmatrix} = \begin{bmatrix} \nabla J + \nabla k^T \nu \\ k \end{bmatrix}, \tag{37}$$

with dual variables $\nu \in \mathbf{R}^{2n_q(H-1)}$ associated with the constraints, uses a Gauss-Newton approximation of the constraints when computing the Hessian of the Lagrangian and is solved using a sparse LDL^T solver. In the following experiments we utilize QDLDL, a general-purpose sparse solver, for its efficient implementation [49].

E. Contact-Height Heuristic

To enable the policy to robustly adapt to unknown variations in terrain height, we employ a simple heuristic that we find to be effective in practice. The policy maintains a height estimate, $a \in \mathbf{R}^c$, for each contact and utilizes a modified signed-distance function:

$$\phi_{MPC}(q) = \phi(q) + a, \tag{38}$$

that is updated using the current contact height. When contact is detected, the height estimate is updated. In simulation, a threshold on the impact-force magnitude is set; and in practice, force sensors can reliably detect such an event.

This simple heuristic does not affect the structure of (23) and only requires c more addition operations to compute r when evaluating the fast contact dynamics (22). In our experiments, we find the heuristic to be effective and reliable across unknown terrain for the systems tested.

F. Algorithm

CI-MPC comprises offline and online stages. The offline stage generates a reference trajectory along with a set of time-varying LCP problems. In this work we employ contact-implicit trajectory optimization [35] to design these references. Additionally, a planning horizon, typically less than the total behavior duration, is specified. During the online stage, planning is performed (1) for the current state over the specified horizon, and the optimized control trajectory is used to compute a control that is applied to the system. The CI-MPC policy is summarized in Algorithm 2.

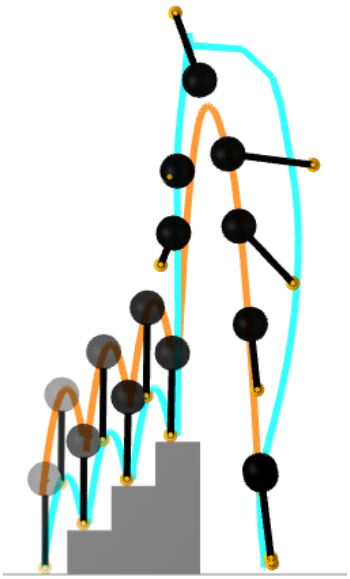


Fig. 3: Hopper in 2D performing parkour. The system tracks the body (orange) and foot (blue) reference trajectories while ascending three stairs before performing a front flip.

G. Heuristics

To enable real-time performance for the policy (31), a number of heuristics are employed. First, the planning problem is only solved approximately. Instead of optimizing until convergence, a fixed number of iterations are performed and then the current best solution is returned. This enables the current plan to be improved but significantly reduces the total computation required. Generally, we find that performance is greatly improved by returning approximate solutions quickly, enabling replanning with newer state information, compared to returning higher quality solutions to planning problems that are utilizing older state information. Second, the policy extensively utilizes warm starting. Providing the optimizer with a good initial guess for the solution greatly reduces the number of iterations required to converge. Initially, the policy utilizes the reference trajectory. At subsequent evaluations, the previous best solution is used. Third, the LCP problems are solved for a single value of the central path parameter instead of a sequence that converges to zero. In practice, we find that $\kappa \approx$ 1e-4 is a good balance between computation time, physical accuracy, and gradient smoothness. Note, when verifying the performance of the policy in simulation, we solve the nonlinear contact dynamics complementarity problem (4) to convergence, i.e., $\kappa =$ 1e-6.

V. Results

We demonstrate the CI-MPC algorithm in simulation and on hardware by controlling a variety of robotic systems that make and break contact with their environments. In the examples we show that the policy can generate new contact sequences online; is robust to disturbances, model mismatch, and unknown terrain; and is faster than real-time, see Table

TABLE II: Comparison between CI-MPC and the Raibert heuristic for a hopper system on 4 scenarios: flat, sinusoidal, and piecewise linear terrains; and a parkour stunt (Fig. 3). For each terrain profile, we report the number of hops achieved by the policy. For the parkour scenario, we report if the stunt is successfully completed.

Policy	Flat	Sinusoidal	Piecewise	Parkour
CI-MPC	+100	+100	+100	✓
Raibert	+100	+100	+100	×

IV. The code, including a Julia implementation of the policy and all of the experiments, is available at:

https://github.com/dojo-sim/ContactImplicitMPC.jl.

A. Simulation

We verify the policy performance in simulation where we solve the nonlinear contact dynamics complementarity problem (4) to convergence. Additionally, all examples are simulated using a different sample rate, typically 5-10× faster than the reference trajectory, in order to ensure that the policy is robust to sampling rates. For example, if the reference is optimized with a time step of 0.1 seconds, then we simulate the nonlinear dynamics with a smaller time step, 0.01 or 0.02 seconds.

PushBot: In this example, we demonstrate that our policy can generate qualitatively new, unspecified contact sequences online in order to respond to unplanned disturbances. The system, PushBot, is modeled as an inverted pendulum with a prismatic joint located at the end of the pendulum (Fig. 2). There are two control inputs: a torque at the revolute joint and a force at the prismatic joint. The system is located between two walls and has two contact points, one between the prismatic-joint end effector and each wall.

PushBot is tasked with remaining vertical and the policy utilizes a reference trajectory that does not include any contacts. When we apply a large impulse to the system, the policy generates a behavior that commands the prismatic joint to push against the wall in order to stabilize. By tuning the policy's cost function we can generate different behaviors, including maintaining contact to stabilize and pushing against the wall in order to return to the nominal position. The latter behavior is shown in Fig. 2.

We compare CI-MPC to a method that relies on a mixed-integer quadratic progam (MIQP) formulation [50] applied to a simplified version of the PushBot (an inverted pendulum between two stiff walls). The MIQP minimizes a quadratic objective function subject to piecewise-linearized dynamics. Each linear dynamics domain corresponds to a single contact mode, and discrete decision variables are introduced to encode contact mode switches. Our CI-MPC approach is fast enough to be run online, however, this is not the case for the MIQP policy, as shown in Table I. Moreover, the complexity of the MIQP increases exponentially with the number of contact modes, making it an intractable approach for more complex systems. Warm-starting the MIQP [51] may make this approach more amenable to online optimization.

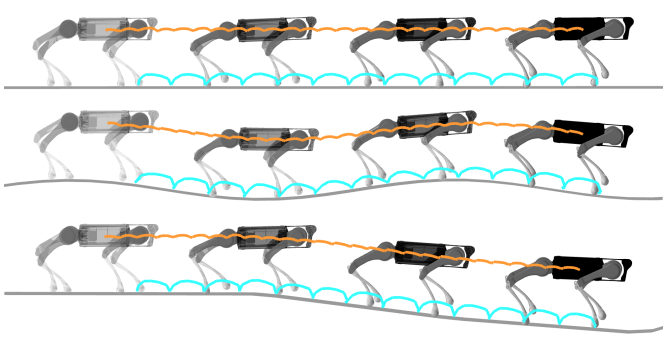


Fig. 4: Planar quadruped walking over uneven terrain. The reference gait is optimized for flat ground. Our CI-MPC policy, with orange center-of-mass and blue foot position trajectories, is able to adapt online to the unmodeled variation in terrain and track the reference trajectory.

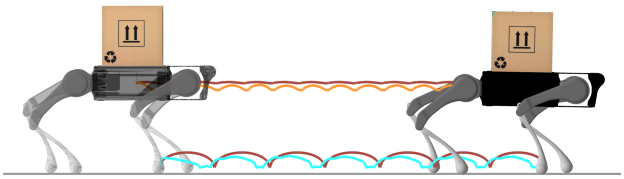


Fig. 5: Quadruped tracking a reference trajectory (red) while carrying an unmodeled 3-kg payload. We depict the torso (orange) and front-left foot (blue) trajectories.

Hopper: Inspired by the Raibert Hopper [17], we model a 2D hopping robot with $n_q = 4$ generalized coordinates: lateral and vertical positions, body orientation, and leg length, respectively; $m = 2$ controls: body moment, e.g., controlled with an internal reaction wheel, and leg force; and a single contact at the foot.

The centroidal-dynamics modeling assumption we make—consistent with Raibert's work—is to locate the leg and foot mass at the body's center of mass. This results in a configuration-independent mass matrix and no bias term in the dynamics.

The hopper is tasked with locomoting over unknown terrain. The CI-MPC policy uses a reference trajectory that is optimized with a flat surface and no incline. We compare our policy to the Raibert heuristic, which we similarly tune for flat ground and no incline. We observe that our policy is able to adapt to the varying surface heights that range from 0-24 cm and that the robot can slip multiple times and is able to recover while traversing steep inclines. We find that, when tuned well, the Raibert heuristic also works very well on terrains

Additionally, we task the hopper with climbing a staircase and executing a front flip (Fig. 3). This complex trajectory cannot be directly executed using the Raibert heuristic as it is not a periodic hopping gait. Our policy, however, successfully tracks this complex trajectory, illustrating the more general capabilities of CI-MPC. Results are summarized in Table II.

Planar quadruped: We model a planar quadruped with $n_q = 11$ configuration variables and $m = 8$ control inputs. The

system has four contacts, one at each point foot.

The quadruped is tasked with moving to the right over three different terrains: flat, sinusoidal, and piecewise-linear surfaces (Fig. 4). Additionally, we test the robustness of the CI-MPC policy by introducing model mismatch. We provide the policy with the nominal model of the quadruped while the simulator uses a quadruped with a 3-kg payload, representing 25% of its nominal mass. Despite the unmodeled load, the policy successfully tracks the nominal gait with good performance.

We note that the same CI-MPC policy was used across all quadruped experiments and no retuning was required to transfer from the nominal case (flat terrain, no payload) to more complex scenarios. Further, it is easy and intuitive to rapidly retune the policy in order to achieve improved tracking performance in the other scenarios.

Planar biped: We model a planar biped based on Pratt's Spring Flamingo [52] with $n_q = 9$ configuration variables and $m = 7$ control inputs. The system is modeled with four contact points, one at the toe and heel of each foot.

The biped is tasked with moving to the right over three different terrains: flat, sinusoidal, and piecewise-linear surfaces (Fig. 6) using the same policy. In Table III, we compare this to Pratt's policy [52], which relies on a state-machine architecture and a number of proportional-derivative controllers. Our CI-MPC policy—with no additional tuning—can easily walk on all of the terrains and reliably walks up inclines of up to ten degrees. Pratt reports that Spring Flamingo can only walk up inclines of five degrees without requiring the controllers to be re-tuned [53].

Monte Carlo: In order to assess the robustness of CI-MPC, we perform Monte Carlo analysis on two systems: the hopper and planar quadruped. The robots are tasked with tracking a reference gait and we initialize the systems with configurations that are randomly perturbed from the reference trajectory. We use 100 randomly sampled initial conditions for each system; the hopper recovers from significant orientation offsets and the quadruped is robust to large drops (Fig. 7).

B. Hardware

In this section we present hardware results that demonstrate two capabilities of this work: 1) real-time performance of our algorithm on hardware 2) the robustness of CI-MPC policies to unmodeled disturbances. Three behaviors are executed on a Unitree Go1 quadruped [16]: first, a trotting gait that is robust to large external disturbances; second, a non-periodic motion where the system moves towards a wall before balance against it using its front feet; third, the system placing both of its feet onto a step. Videos of the experiments are included with the associated materials.

Point-foot model: We utilize a simplified point-foot model of the quadruped that neglects leg dynamics. This model comprises 36 states which include the positions and velocities of the body and each foot. The orientation of the body is represented with Euler angles. The controls are three-dimensional forces applied to each foot, which is modeled as a point mass. Reference motions are generated offline with this model using contact-implicit trajectory optimization [35].

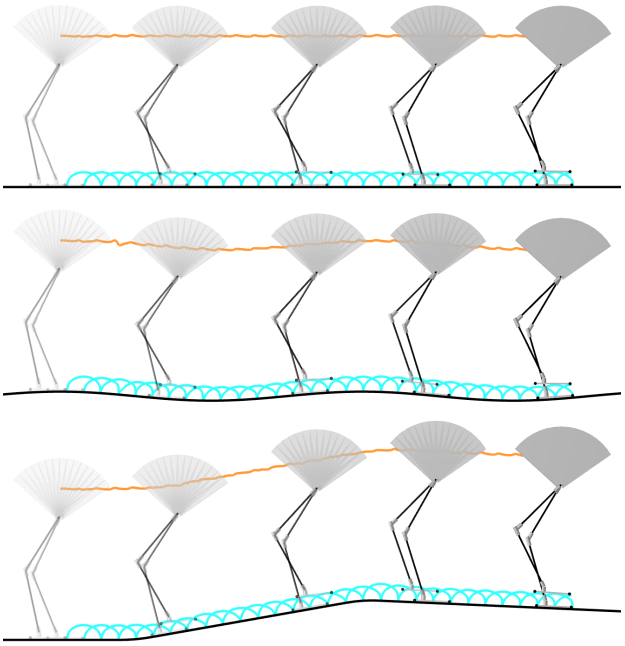


Fig. 6: Biped walking from left to right across flat (top), sinusoidal (middle), and piecewise linear (bottom) terrain using the same policy.

The primary reason for these modeling simplifications is to reduce the online computational requirement when evaluating the dynamics and their derivatives while still being able to reason about new contact sequences online. Importantly, unlike traditional convex quadratic programming policies [54], which assume a fixed contact sequence for the feet, our model is contact-implicit and enables new foot-step sequences to be generated online.

Experimental setup: A reference trajectory generated offline is tracked online using CI-MPC. A pre-tuned low-level controller generates joint torques that aim to match the forces specified by the policy that should be applied by each leg at the foot. The CI-MPC policy is written in Julia and is pre-compiled in order to interface with an existing C++ low-level controller running at 1000Hz. State feedback to the CI-MPC policy and the low-level controller is provided by a 1000Hz Kalman filter which utilizes joint encoders, onboard IMU, and external motion-capture tracking to estimate the robot state. The policy, state estimator, and low-level controller run on a computer equipped with an Intel i9-12900KS CPU and 64GB of memory. The joint torque commands are sent to the quadruped via an ethernet connection. Additional information is provided in Table V.

Trotting: In this example, the specified behavior is trotting in place (Fig. 1). The reference trajectory is 0.8 seconds and is repeated to form a continuous gait. The policy planning horizon is 0.10 seconds and the model uses a time discretization of 0.05 seconds. The policy runs at an average rate 100 Hz.

Wall stand: In this example, the specified behavior is transitioning from four feet on the ground to standing against the wall with two feet (Fig. 1). The reference trajectory is 19.75 seconds. The policy planning horizon is 0.15 seconds

TABLE III: Comparison between CI-MPC and Pratt state-machine [52] policies for flamingo system on flat and inclined terrains. We report the number of steps taken by the robot on the flat terrain and compare the maximum incline traversed by our policy in simulation with reported results† [53].

Policy	Flat	Incline
CI-MPC	$+100$	**10 deg.**
Pratt	$+100^{\dagger}$	$5^{\dagger}$ deg.

and the model uses a time discretization of 0.05 seconds. The policy runs at an average rate of 100 Hz.

Step: In this example, the specified behavior has the quadruped place its right foot onto a step, followed by its left foot (Fig. 1). The entire reference trajectory is 7.0. The policy planning horizon is 0.1 seconds and the model uses a time discretization of 0.05 seconds. The policy runs at an average rate of 100 Hz.

VI. Conclusion

CI-MPC is capable of generating dynamic behaviors by robustly tracking reference trajectories through contact despite disturbances, model mismatch, and uncertain environments. In this section we conclude with a discussion of limitations and directions for future research.

A. Limitations

We highlight important limitations including: approximations, contact model, and reliability, that should be considered before deploying CI-MPC policies.

Approximations: For the online trajectory optimization problem, strategically approximated contact dynamics are utilized for planning. This enables expensive gradient computations and partial matrix factorizations to be performed in an offline stage, in order to substantially reduce online computation. In the examples, we find that short planning horizons, typically between 0.1 and 0.25 seconds, are sufficient. Despite the approximations introduced by these simplifications, in practice, the controls optimized for the fast contact dynamics work well in simulation and on hardware. However, it remains to be seen how well these approximations work for control of highly dynamic behaviours or scenarios that require longer planning horizons, particularly when deployed on hardware.

Contact model: The physics of hard contact produces non-smooth and discontinuous gradients. With our custom interior-point method for the contact dynamics solver, we can efficiently compute smooth gradients in a principled way by exploiting intermediate results, parameterized by the central-path parameter. Hard contact is simulated by returning results from the contact dynamics solver with a central-path value $\kappa_{\text{sim}} = 1\text{e-}6$, whereas gradients are computed using intermediate results from the solve parameterized by $\kappa_{\text{grad}} \approx 1\text{e-}4$.

During online optimization, we prioritize fast updates by solving the trajectory-tracking problem to coarse tolerances. In this context, imposing highly accurate contact physics would be wasteful in terms of computational resources. As a result, the central-path value for the planning dynamics is fixed to the gradient central-path value in order to reduce online computation. This selection was made to balance capturing accurate physics with producing usefully smooth gradients. Empirically, we observe that using these dynamics with slightly soften contact dynamics enhanced the convergence of the trajectory-tracking solver and likely enables the policy to more easily discover new contact sequences. Importantly, the allowed interpenetration with these tolerances is sub-millimeter—much less than allowed by MuJoCo's default settings—but, this raises the question, is simulation of perfectly hard contact actually necessary, or useful, for reliable planning and control of non-smooth systems?

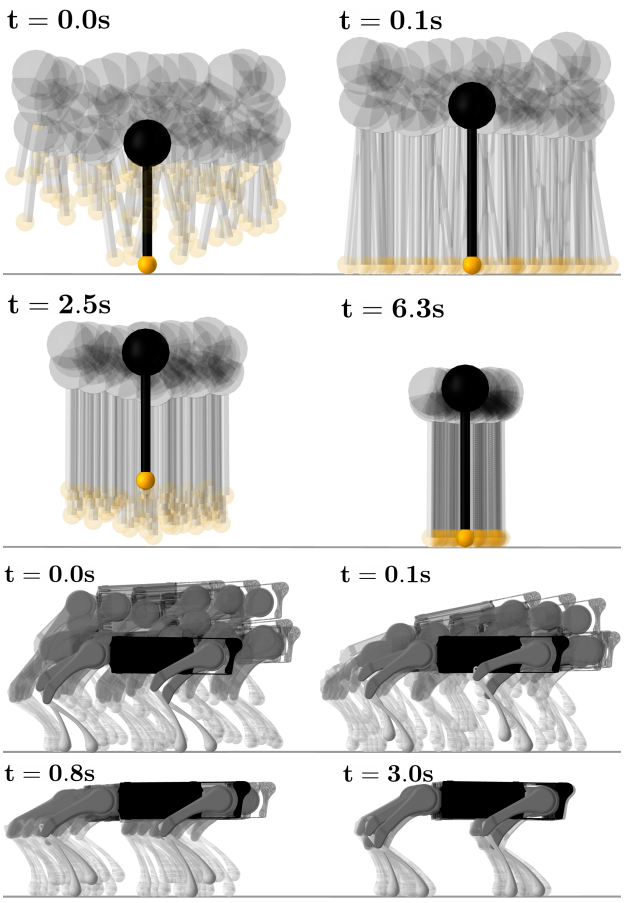


Fig. 7: Monte Carlo simulations of initial conditions for systems tracking a reference trajectory. 100 initial configurations are randomly sampled for a hopper (top) and quadruped (bottom). Perturbations from the reference initial configuration include large translations, tilts, and joint angles offsets. For all the samples, and both systems, the policy successfully recovers to the reference gait.

Reliability: Generating high-quality reference trajectories is crucial for CI-MPC. Contact-implicit trajectory optimization [34, 35] is a powerful tool for generating these trajectories. However, it is notorious for poor convergence properties, despite relying on robust large-scale constrained solvers for non-convex problems and even good warm starting. This unreliability makes online optimization generally impractical—motivating this work. Ultimately, for CI-MPC to be of prac-

TABLE IV: The CI-MPC policy runs at real-time rates meaning that the time required to compute the control is always smaller than the reference time step and the policy is able to successfully track the specified trajectory. Experiments are run on a computer equipped with an Intel Core i9-9900 CPU and 32GB of memory.

System	Planning Horizon	Time Step	Real-Time
pushbot	1.60 s	0.04 s	✓
hopper (2D)	0.10 s	0.01 s	✓
hopper (3D)	0.20 s	0.01 s	✓
quadruped	0.16 s	0.016 s	✓
biped	0.23 s	0.016 s	✓

TABLE V: Hardware experiments with CI-MPC tracking different reference trajectories on a Unitree Go1 quadruped.

Settings	Trotting	Wall	Step
reference trajectory length	0.8 s	19.75 s	7.0 s
reference time step	0.05 s	0.05 s	0.05 s
policy planning horizon	0.1 s	0.15 s	0.1 s
policy rate	100 Hz	100 Hz	100 Hz

tical value, generation of references trajectories, even in the offline setting, must be improved. This may be possible with specialized solvers for contact-implicit trajectory optimization [55], alternative rollout-based methods that leverage the reliability of one-step contact dynamics [56], or learning-based approaches [5].

Additionally, our hardware experiments demonstrate the real-time capabilities of CI-MPC and its ability to be robust in many scenarios. However, in practice, we find that the classic convex MPC policy [57] is significantly more robust for general locomotion because that policy has an additional online foothold selection strategy (the Raibert strategy). Adding this strategy to CI-MPC will likely increase its speed and reliability to match the performance of the baseline convex MPC policy. Performance will likely be improved further with a more efficient C/C++ implementation that utilizes multi-threading for parallel evaluations of the fast contact dynamics.

B. Future Work

In summary, we have presented fast differentiable contact dynamics that can be utilized in an MPC framework that performs robust tracking for robotic systems that make and break contact with their environments. There remain many exciting avenues to explore in future work. First, it should be possible to perform higher-fidelity convex approximations of the contact dynamics that utilize second-order friction cones instead of its linearized approximation. This could enable tracking of highly dynamic behaviors that leverage accurate sliding contacts. Second, a natural extension of this work, which was focused on locomotion, is to the manipulation domain, potentially with quasi-static models, where control through contact is similarly an open problem, but where dynamics are slower and more amenable to online optimization. Third, in our hardware experiments, we utilized a simplified planning model for the quadruped. Similar point-contact models should extend to bipeds and humanoid systems, potentially even dexterous hands. Lastly, a library of template behaviors, comprising CI-MPC policies, could be composed to enable more diverse behavior online with a high-level agent composing templates in a task-and-motion-planning framework in order to generate complex long-horizon plans.

ACKNOWLEDGMENTS

This work was supported in part by Frontier Robotics, Innovative Research Excellence, Honda R&D Co., Ltd, ONR award N00014-18-1-2830, NSF NRI award 1830402, and DARPA YFA award D18AP00064. Toyota Research Institute ("TRI") provided funds to assist the authors with their research but this article solely reflects the opinions and conclusions of its authors and not TRI or any other Toyota entity.

REFERENCES

[1] E. R. Westervelt, J. W. Grizzle, and D. E. Koditschek, "Hybrid zero dynamics of planar biped walkers," *IEEE Transactions on Automatic Control*, vol. 48, no. 1, pp. 42–56, 2003.

[2] A. D. Ames, K. Galloway, K. Sreenath, and J. W. Grizzle, "Rapidly exponentially stabilizing control Lyapunov functions and hybrid zero dynamics," *IEEE Transactions on Automatic Control*, vol. 59, no. 4, pp. 876–891, 2014.

[3] Z. Li, X. Cheng, X. B. Peng, P. Abbeel, S. Levine, G. Berseth, and K. Sreenath, "Reinforcement learning for robust parameterized locomotion control of bipedal robots," in *IEEE International Conference on Robotics and Automation*, 2021, pp. 2811–2817.

[4] A. Aydinoglu, V. M. Preciado, and M. Posa, "Contact-aware controller design for complementarity systems," in *IEEE International Conference on Robotics and Automation*, 2020, pp. 1525–1531.

[5] N. Heess, D. TB, S. Sriram, J. Lemmon, J. Merel, G. Wayne, Y. Tassa, T. Erez, Z. Wang, S. Eslami *et al.*, "Emergence of locomotion behaviours in rich environments," *arXiv:1707.02286*, 2017.

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

[7] 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.

[8] J. 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.

[9] 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

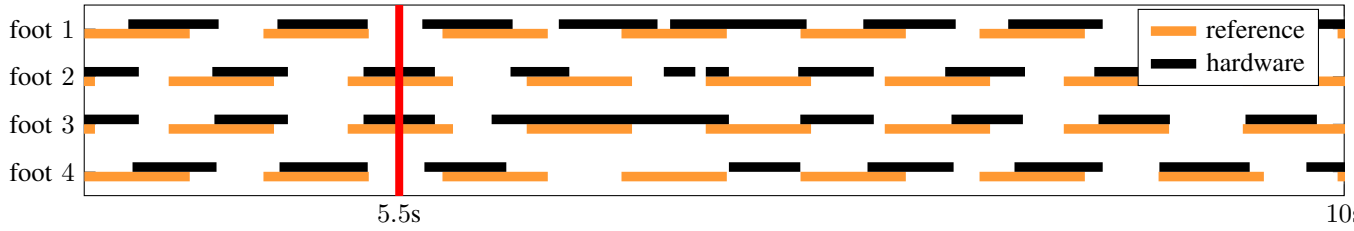


Fig. 8: Contact sequence for quadruped trotting. An external disturbance is applied to the system (red) and the policy is able to generate a new contact sequence online (black) that differs significantly from the reference plan (orange).

robot," *Autonomous Robots*, vol. 40, no. 3, pp. 429–455, 2016.

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

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

[12] 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 *IEEE/RSJ International Conference on Intelligent Robots and Systems*, 2015, pp. 3346–3351.

[13] K. Hirai, M. Hirose, Y. Haikawa, and T. Takenaka, "The development of Honda humanoid robot," in *IEEE International Conference on Robotics and Automation*, vol. 2, 1998, pp. 1321–1326.

[14] G. Bledt, "Regularized predictive control framework for robust dynamic legged locomotion," Ph.D. dissertation, Massachusetts Institute of Technology, 2020.

[15] J. Lee, J. Hwangbo, L. Wellhausen, V. Koltun, and M. Hutter, "Learning quadrupedal locomotion over challenging terrain," *Science Robotics*, vol. 5, no. 47, 2020.

[16] "Unitree Go1 Quadruped," https://m.unitree.com/products/go1/, 2022.

[17] M. H. Raibert, H. B. Brown Jr., M. Chepponis, J. Koechling, J. K. Hodgins, D. Dustman, W. K. Brennan, D. S. Barrett, C. M. Thompson, J. D. Hebert, W. Lee, and B. Lance, "Dynamically stable legged locomotion," Massachusetts Institute of Technology Cambridge Artificial Intelligence Lab, Tech. Rep., 1989.

[18] H. Dai, A. Valenzuela, and R. Tedrake, "Whole-body motion planning with centroidal dynamics and full kinematics," in *IEEE-RAS International Conference on Humanoid Robots*, 2014, pp. 295–302.

[19] Boston Dynamics. (2019) More Parkour Atlas. [Online]. Available: https://www.youtube.com/watch?v=_sBBaNYex3E

[20] M. Chignoli, D. Kim, E. Stanger-Jones, and S. Kim, "The MIT humanoid robot: Design, motion planning, and control for acrobatic behaviors," *IEEE-RAS International Conference on Humanoid Robots (Humanoids)*, pp. 1–8, 2021.

[21] M. Neunert, M. Stäuble, M. Giftthaler, C. D. Bellicoso, J. Carius, C. Gehring, M. Hutter, and J. Buchli, "Whole-body nonlinear model predictive control through contacts for quadrupeds," *IEEE Robotics and Automation Letters*, vol. 3, no. 3, pp. 1458–1465, 2018.

[22] D. Kim, J. Di Carlo, B. Katz, G. Bledt, and S. Kim, "Highly dynamic quadruped locomotion via whole-body impulse control and model predictive control," *arXiv:1909.06586*, 2019.

[23] E. Coumans and Y. Bai, "PyBullet, a Python module for physics simulation for games, robotics and machine learning," 2016–2019. [Online]. Available: http://pybullet.org

[24] J. Lee, M. X. Grey, S. Ha, T. Kunz, S. Jain, Y. Ye, S. S. Srinivasa, M. Stilman, and C. K. Liu, "DART: Dynamic animation and robotics toolkit," *Journal of Open Source Software*, vol. 3, no. 22, p. 500, 2018.

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

[26] E. Drumwright, "Rapidly computable viscous friction and no-slip rigid contact models," *arXiv:1504.00719*, 2015.

[27] R. W. Cottle, J. Pang, and R. E. Stone, *The Linear Complementarity Problem*. SIAM, 2009.

[28] M. Kojima, N. Megiddo, T. Noma, and A. Yoshise, "A unified approach to interior point algorithms for linear complementarity problems: A summary," *Operations Research Letters*, vol. 10, no. 5, pp. 247–254, 1991.

[29] 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.

[30] U. Dini, *Lezioni di analisi infinitesimale*. Fratelli Nistri, 1907, vol. 1.

[31] T. A. Howell, S. Le Cleac'h, J. Z. Kolter, M. Schwager, and Z. Manchester, "Dojo: A differentiable physics engine for robotics," *arXiv:2203.00806*, 2022.

[32] K. Werling, D. Omens, J. Lee, I. Exarchos, and C. K. Liu, "Fast and feature-complete differentiable physics for articulated rigid bodies with contact," *arXiv:2103.16021*, 2021.

[33] O. Von Stryk, "Numerical solution of optimal control problems by direct collocation," in *Optimal Control*. Springer, 1993, pp. 129–143.

[34] 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.

[35] Z. Manchester and S. Kuindersma, "Variational contact-implicit trajectory optimization," in *Robotics Research*. Springer, 2020, pp. 985–1000.

[36] F. de Avila Belbute-Peres, K. Smith, K. Allen, J. Tenenbaum, and J. Z. Kolter, "End-to-end differentiable physics for learning and control," *Advances in Neural Information Processing Systems*, vol. 31, pp. 7178–7189, 2018.

[37] M. Anitescu, "Optimization-based simulation of nonsmooth rigid multibody dynamics," *Mathematical Programming*, vol. 105, no. 1, pp. 113–143, 2006.

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

[39] J. Richalet, A. Rault, J. L. Testud, and J. Papon, "Model predictive heuristic control: Applications to industrial processes," *Automatica*, vol. 14, no. 5, pp. 413–428, 1978.

[40] R. E. Kalman, "When Is a Linear Control System Optimal?" *Journal of Basic Engineering*, vol. 86, no. 1, pp. 51–60, 1964.

[41] Y. Wang and S. Boyd, "Fast model predictive control using online optimization," *IEEE Transactions on Control Systems Technology*, vol. 18, no. 2, pp. 267–278, 2009.

[42] D. E. Stewart and J. C. Trinkle, "An implicit time-stepping scheme for rigid body dynamics with inelastic collisions and Coulomb friction," *International Journal for Numerical Methods in Engineering*, vol. 39, no. 15, pp. 2673–2691, 1996.

[43] J. E. Marsden and M. West, "Discrete mechanics and variational integrators," *Acta Numerica*, vol. 10, pp. 357–514, 2001.

[44] J. J. Moreau, "On unilateral constraints, friction and plasticity," in *New Variational Techniques in Mathematical Physics*. Springer, 2011, pp. 171–322.

[45] J. Nocedal and S. J. Wright, *Numerical Optimization*, 2nd ed. Springer, 2006.

[46] S. Mehrotra, "On the implementation of a primal-dual interior point method," *SIAM Journal on Optimization*, vol. 2, no. 4, pp. 575–601, 1992.

[47] O. Von Stryk and R. Bulirsch, "Direct and indirect methods for trajectory optimization," *Annals of Operations Research*, vol. 37, no. 1, pp. 357–373, 1992.

[48] I. Yamazaki, S. Nooshabadi, S. Tomov, and J. Dongarra, "Structure-aware linear solver for realtime convex optimization for embedded systems," *IEEE Embedded Systems Letters*, vol. 9, no. 3, pp. 61–64, 2017.

[49] T. A. Davis, "Algorithm 849: A concise sparse Cholesky factorization package," *ACM Transactions on Mathematical Software*, vol. 31, no. 4, pp. 587–591, 2005.

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

[51] T. Marcucci and R. Tedrake, "Warm start of mixed-integer programs for model predictive control of hybrid systems," *Transactions on Automatic Control*, vol. 66, no. 6, pp. 2433–2448, 2020.

[52] J. E. Pratt, "Exploiting Inherent Robustness and Natural Dynamics in the Control of Bipedal Walking Robots," Ph.D. dissertation, Massachusetts Institute of Technology, 2000.

[53] J. Pratt and G. Pratt, "Intuitive control of a planar bipedal walking robot," in *IEEE International Conference on Robotics and Automation*, vol. 3, 1998, pp. 2014–2021.

[54] G. Bledt, P. M. Wensing, and S. Kim, "Policy-regularized model predictive control to stabilize diverse quadrupedal gaits for the MIT cheetah," in *IEEE/RSJ International Conference on Intelligent Robots and Systems*, 2017, pp. 4102–4109.

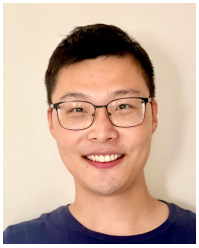
[55] T. A. Howell, S. Le Cleac'h, K. Tracy, and Z. Manchester, "CALIPSO: A differentiable solver for trajectory optimization with conic and complementarity constraints," *arXiv:2205.09255*, 2022.

[56] T. A. Howell, S. Le Cleac'h, S. Singh, P. Florence, Z. Manchester, and V. Sindhwani, "Trajectory optimization with optimization-based dynamics," *IEEE Robotics and Automation Letters*, vol. 7, no. 3, pp. 6750–6757, 2022.

[57] J. Di Carlo, P. M. Wensing, B. Katz, G. Bledt, and S. Kim, "Dynamic locomotion in the MIT cheetah 3 through convex model-predictive control," in *International Conference on Intelligent Robots and Systems*, 2018, pp. 1–9.

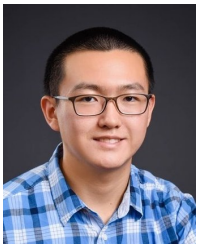
Simon Le Cleac'h is a graduate student with the Robotic Exploration Lab at Carnegie Mellon University and with the Multi-Robot Systems Lab at Stanford University. He received his BS in Engineering from Ecole Centrale Paris in 2016 and his MS in Mechanical Engineering from Stanford University in 2019. His research interests include differentiable optimization for control and simulation of robotics systems.

Taylor Howell is a graduate student at Stanford University and with the Robotic Exploration Lab. He received his BS in mechanical engineering in 2016 from the University of Utah and his MS in mechanical engineering from Stanford University in 2019. His research interests include numerical optimization and control for robotic systems.



Shuo Yang is a graduate student at the Robotics Exploration Lab at Carnegie Mellon University. He received his BEng degree in Computer Engineering in 2012 and MPhil degree in Electrical and Computer Engineering in 2015, both from the Hong Kong University of Science and Technology. His research interests include trajectory optimization and state estimation for robotic systems.

Chi-Yen Lee is a recent graduate from the Robotics Exploration Lab at Carnegie Mellon University. He received his BS in Engineering from Harvey Mudd College in 2019 and his MS in Robotics from Carnegie Mellon University in 2022. His research interests include optimal control and state estimation for robotics system.



John Zhang is a graduate student at the Robotic Exploration Lab at Carnegie Mellon University. He received his B.S. in Mechanical Engineering and a minor in Computer Science from the Georgia Institute of Technology. His research interests include optimization and control for robotic systems.

Arun Bishop is a Phd Student in the Robotics Institute at Carnegie Mellon University. He received his B.S.E. in Mechanical Engineering and his M.S. in Robotics from the University of Michigan working on robust bipedal walking. His research interests include optimal control and design of legged robotic systems for applications in exploration and search and rescue.

Mac Schwager is an assistant professor with the Aeronautics and Astronautics Department at Stanford University. He obtained his BS degree in 2000 from Stanford University, his MS degree from MIT in 2005, and his PhD degree from MIT in 2009. He was a postdoctoral researcher working jointly in the GRASP lab at the University of Pennsylvania and CSAIL at MIT from 2010 to 2012, and was an assistant professor at Boston University from 2012 to 2015. He received the NSF CAREER award in 2014, the DARPA YFA in 2018, and a Google faculty research award in 2018, and the IROS Toshio Fukuda Young Professional Award in 2019. His research interests are in distributed algorithms for control, perception, and learning in groups of robots, and models of cooperation and competition in groups of engineered and natural agents.

Zachary Manchester is an assistant professor in the Robotics Institute at Carnegie Mellon University and founder of the Robotic Exploration Lab. He received a PhD in aerospace engineering in 2015 and a BS in applied physics in 2009, both from Cornell University. He was a postdoctoral fellow in the Agile Robotics Lab at Harvard from 2015 to 2017 and an assistant professor at Stanford from 2018 to 2020. He received the NASA Early Career Faculty Award in 2018 and a Google Faculty Research Award in 2020. His research interest include numerical optimization, control and estimation with applications to aerospace and robotic systems with challenging nonlinear dynamics.