Fast Contact-Implicit Model-Predictive Control

Simon Le Cleac'h[1*], Taylor A. Howell[1*], Mac Schwager[2], and Zachary Manchester[3]

***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 relying on linear complementarity problems (LCP) computed using strategic Taylor approximations about a reference trajectory and retaining non-smooth impact and friction dynamics, allowing the policy to not only reason about contact forces and timing, but also generate entirely new contact mode sequences online. To achieve reliable and fast numerical convergence, we devise a structure-exploiting, path-following solver for the LCP contact dynamics and a custom trajectory optimizer for trajectory-tracking MPC problems. We demonstrate CI-MPC at real-time rates in simulation, and show that it is robust to model mismatch and can respond to disturbances by discovering and exploiting new contact modes across a variety of robotic systems, including a pushbot, hopper, and planar quadruped and biped.**

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

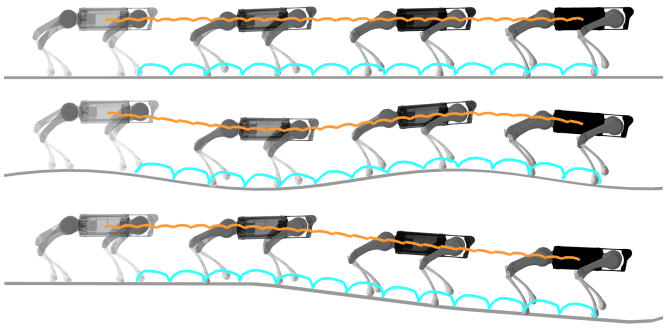

Fig. 1: Planar quadruped walking over uneven terrain. The reference gait is optimized for flat ground. Our CI-MPC policy, with orange center-of-mass and blue foot position trajectories, is able to adapt online to the unmodeled variation in terrain and track the reference trajectory.

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 [39, 1, 24], to complementarity controllers [3], to neural-network policies [15, 16], and model-predictive control (MPC) [40, 33]. There have also been numerous successes deploying such approaches on complex systems in recent years: direct trajectory optimization and LQR on Atlas [21], smooth-contact models and differential dynamic programming on HRP-2 [36, 19], zero-moment point and feedback linearization on ASIMO [17], and MPC with simplified dynamics models on Cheetah [5] and ANYmal [23]. 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.

In this work, we focus on the problem of local tracking control for systems that experience contact interactions with their environments. Our approach combines a differentiable "hard-contact" rigid-body dynamics formulation with strategic linearizations, exploitation of the trajectory-optimization problem structure, and specialized numerical optimization techniques. The result is a 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 on modest computing hardware.

We formulate dynamics with contact as a complementarity problem that simultaneously satisfies impact and friction constraints. By employing a path-following method to optimize this problem, we naturally and reliably converge from "soft" to "hard" contact as the central-path parameter is decreased to zero. At a solution point, the implicit-function theorem is then utilized to efficiently compute dynamics derivatives for use in the policy. To enable real-time performance for model-predictive 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). An upper-level trajectory-optimization problem is then optimized using an efficient structure-exploiting solver. We refer to this algorithm as *contact-implicit model-predictive control* (CI-MPC).

Finally, we demonstrate that our CI-MPC policy can generate new contact sequences online and reliably track reference trajectories while subject to significant model mismatch and large disturbances for a number of qualitatively different robotic systems, including a pushbot, hopper, and planar

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

[3] Zachary Manchester is with The Robotics Institute, Carnegie Mellon University, Pittsburgh, PA 15213, USA. zacm@cmu.edu

(Corresponding author: Taylor A. Howell)

[*] These authors contributed equally to this work.

quadruped and biped.

Our specific contributions are:

- A contact-dynamics formulation that can be reliably evaluated and efficiently differentiated with a custom path-following solver
- Fast, 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 examples demonstrating the performance of the CI-MPC algorithm on a variety of robotic systems across a range of highly dynamic balancing and locomotion tasks

In the remainder of this paper, we first review related work on controlling systems that experience contact events with MPC and LCP contact dynamics in Section II. Next, we present a brief overview of linear MPC, outline the classic LCP for contact dynamics, and provide background on path-following methods and how to differentiate through their solutions via the implicit-function theorem in Section III. Then, we present our differentiable contact-dynamics implementation in Section IV and our CI-MPC algorithm in Section V. In Section VI, we provide simulation results to demonstrate the policy's performance. Finally, we discuss our results in Section VII and conclude with directions for future work in Section VIII.

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 methods for solving LCP 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 [32]. 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 obstacles scenarios [8] and perform parkour [13]. Integrating hardware design and controller synthesis has also recently enabled small humanoids to perform agile acrobatic maneuvers in simulation [6].

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, [5]. 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 [27]. Approaches that utilize both force-based MPC and whole-body control have also demonstrated agile locomotion [18].

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 our CI-MPC approach 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 while achieving comparable or better performance.

B. LCP Contact Dynamics

Our CI-MPC framework relies on efficient evaluations of contact dynamics. The classic approach for simulating rigid-body systems that experience contact is a velocity-based time-stepping scheme that optimizes an LCP. The LCP's solution produces a one-step simulation of rigid-body systems that experience hard contact. This problem can be efficiently optimized using pivoting-based techniques [12], such as Lemke's algorithm [7], or path-following methods [20]. Implementations of pivoting methods can be found in general LCP solvers such as PATH [11], or contact-dynamics simulators including DART [22]. Pivoting approaches enforce strict complementarity at each iteration, frequently resulting in non-differentiable solution points. While it is possible to compute subgradients of the solution in these situations, optimization using these subgradients becomes much more difficult. In contrast, path-following methods relax the complementarity constraints, and can return smooth gradients.

For trajectory generation, the LCP problem's contraints have been directly used to encode contact dynamics with collocation techniques in large nonlinear programs in order to optimize trajectories without pre-specified mode sequences for locomotion and simple manipulation tasks [29]. Subsequent work improved this approach by introducing higher-order integrators for the dynamics (5) and a numerically robust, exact ℓ_1-penalty for handling the complementarity constraints [25]. Alternative gradient-based approaches that utilize simulator rollouts differentiate through the solution of a one-step LCP [9, 38].

Another popular contact-dynamics formulation is MuJoCo's [36] soft-contact model, which solves a convex optimization problem and trades physical realism for fast and reliable performance. It is possible to recover finite-differenced gradients from the simulator. Similarly, the LCP complementarity constraints can be relaxed, resulting in a soft-contact model that exhibits improved numerical properties [14].

III. Background

In this section, we provide technical background on linear MPC, the LCP formulation for contact dynamics, introduce path-following methods for optimizings LCPs and an approach for differentiating through their solutions.

A. Model-Predictive Control

CI-MPC is a receding-horizon control policy; in this section, we provide background on this type of algorithm, which

optimizes the following trajectory-optimization 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 system with state $x_t \in \mathbf{R}^n$, control inputs $u_t \in \mathbf{R}^m$, time index t, initial state x_1, and discrete-time dynamics $f_t : \mathbf{R}^n \times \mathbf{R}^m \to \mathbf{R}^n$, we aim to minimize an objective with stage-cost functions, $g_t : \mathbf{R}^n \times \mathbf{R}^m \to \mathbf{R}$, and terminal-cost function, $g_T : \mathbf{R}^n \to \mathbf{R}$, over a planning horizon T.

To reduce computational complexity for online performance, the actual problem we would like to optimize is often replaced with a proxy that has linear dynamics,

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

and quadratic cost functions,

$$g_t(x_t, u_t) = x_t^T Q_t x_t + q_t^T x_t + u_t^T R_t u_t + r_t^T u_t, \tag{3}$$

$$g_T(x_T) = x_T^T Q_T x_T + q_T^T x_T. \tag{4}$$

This formulation, with additional affine state and control constraints, is commonly referred to as *linear* MPC. Without such constraints, (1) is solved efficiently with a backward Riccati recursion and is commonly referred to as the LQR problem.

In MPC, after optimizing an instantiation of (1), the optimized control at the first time step, u_1^*, is applied to the system. After the system evolves, x_1 is updated and (1) is re-optimized in order to compute a new control input for the system. By repeating this procedure, feedback control is achieved [37]. 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.

In order to optimize (1) efficiently using Newton or quasi-Newton methods, MPC requires differentiable dynamics constraints. In the case of linear MPC, these derivatives, i.e., A_t and B_t, resulting from a linearization about a reference trajectory that is being tracked, can be pre-computed offline for additional efficiency. Quadratic objectives are often designed to track a reference trajectory, and they can similarly be pre-computed offline. Similar to linear MPC, CI-MPC strategically linearizes components of the dynamics about a reference trajectory in order to leverage offline pre-computation and matrix pre-factorization. However, unlike linear MPC, CI-MPC does not result in a convex quadratic program, but in a nonlinear program since the contact dynamics are only partially linearized and retain complementarity constraints.

B. LCP Contact Dynamics

Our CI-MPC framework relies on a velocity-based time-stepping scheme that optimizes an LCP in order to find the systems's next configuration, $q_{t+1} \in \mathbf{R}^n$, and velocity, $v_{t+1} \in \mathbf{R}^n$ [34]. The LCP is comprised of impact and friction subproblems.

The impact problem has the following structure,

$$M(q_t)v_{t+1} - M(q_{t-1})v_t + hC(q_t, v_{t+1}) = J(q_{t+1})^T \lambda_t + B(q_{t+1})u_t, \tag{5}$$

$$q_{t+1} = q_t + hv_{t+1}, \tag{6}$$

$$\gamma_t \circ \phi(q_{t+1}) = 0, \tag{7}$$

$$\gamma_t,\ \phi(q_{t+1}) \geq 0, \tag{8}$$

with: mass matrix $M : \mathbf{R}^n \to \mathbf{S}^n_{++}$; dynamics bias $C : \mathbf{R}^n \times \mathbf{R}^n \to \mathbf{R}^n$ that includes Coriolis and gravitational terms; contact Jacobian $J : \mathbf{R}^l \to \mathbf{R}^n$ that maps contact forces in the local surface frame into the generalized coordinates; input Jacobian $B : \mathbf{R}^n \to \mathbf{R}^{n \times m}$ that maps control inputs, typically joint torques, into the generalized coordinates; time step $h \in \mathbf{R}_+$; contact forces $\lambda_t = (\gamma_t^{(1)}, \beta_t^{(1)}, \ldots, \gamma_t^{(c)}, \beta_t^{(c)}) \in \mathbf{R}^l$ defined in the local surface frame, with normal force $\gamma_t^{(i)} \in \mathbf{R}$ and friction forces $\beta_t^{(i)} \in \mathbf{R}^{2(d-1)}$ which have been double parameterized for the linearized friction cone; signed-distance function, $\phi : \mathbf{R}^n \to \mathbf{R}^c$, that returns distances between specified contact points on the robot (e.g., feet) and the closest surface in the environment (e.g., the floor); and $\circ$ is an element-wise (Hadamard) vector product. We use c to denote the number of contact points, and the environment dimension $d = 2$ for planar systems and $d = 3$ otherwise. The manipulator dynamics (5) are discretized with a semi-implicit Euler scheme, and the complementarity constraints (7)–(8) encode the fact that impact forces can only be imparted to the system while in contact, and can only be repulsive.

For the integration scheme, we note that J, B, C, and ϕ should be evaluated at q_{t+1} but in practice the approximation $\tilde{q}_{t+1} = q_t + hv_t$ is utilized in order for the constraints to be linear in the decision variables. Instead of this approximation, CI-MPC evaluates these terms at configurations along a known reference trajectory.

Coulomb friction is modeled at each contact point using the maximum-dissipation principle [26] and a linearized friction cone, and is encoded with the following constraints:

$$\eta_t - \begin{bmatrix}\nu_{t+1}^T & -\nu_{t+1}^T\end{bmatrix}^T - \psi_t \mathbf{1} = 0 \tag{9}$$

$$\psi_t \cdot (\mu\gamma_t - \mathbf{1}^T \beta_t) = 0 \tag{10}$$

$$\beta_t \circ \eta_t = 0 \tag{11}$$

$$\beta_t, \eta_t \geq 0, \tag{12}$$

where $\nu_t \in \mathbf{R}^{d-1}$ is the tangential velocity at a contact point, $\mu \in \mathbf{R}_+$ is the coefficient of friction, $\psi_t \in \mathbf{R}$ is the dual variable (Lagrange multiplier) associated with a linearized friction-cone constraint, and $\eta_t \in \mathbf{R}^{2(d-1)}$ is the dual variable associated with the nonnegative friction-force constraint. The tangential velocity is computed by projecting the configuration velocity through the contact Jacobian and taking the appropriate components.

The impact and friction problems (5)–(12) are coupled through the contact forces and the system's configuration and velocity, forming an LCP; its solution produces a one-step simulation of rigid-body systems that experience hard contact. The following section presents a path-following method to optimize such problems.

C. Path-Following Method

Path-following methods can efficiently and reliably optimize problems with inequality and complementarity constraints [20, 28]. In this section, we focus on optimizing LCPs of the form[1]:

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

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

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

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

The equality and relaxed bilinear constraints form a residual vector or solution map, $r : \mathbf{R}^{n+2m} \times \mathbf{R}^p \times \mathbf{R}_+ \to \mathbf{R}^{n+2m}$, 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 $\rho \to 0$ until the central-path parameter, also referred to as complementary slackness, is below a desired tolerance.

D. Implicit-Function Theorem

An implicit function, $r : \mathbf{R}^k \times \mathbf{R}^p \to \mathbf{R}^k$, 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., $\frac{\partial z}{\partial \theta}$, can be computed by utilizing the implicit-function theorem [10]. 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. We will use this approach to differentiate through the solution from a path-following method in order to compute gradients of the contact dynamics for CI-MPC. A differentiable path-following method is summarized in Algorithm 1.

[1]Technically, a *mixed* linear complementarity problem (MLCP) since the optimization variable x is not part of the complementarity constraint.

IV. Differentiable Contact Dynamics

In this section we present a differentiable contact dynamics formulation that can return smooth gradients. First, we formulate a parameterized complementarity problem, that jointly solves the impact and friction problems, for a path-following method. Solving this problem with a path-following method successively reduces the complementary slackness, helping to avoid numerical issues inherent to non-smooth and discontinuous impact and friction dynamics, and directly corresponds to converging from "soft" to "hard" contact. Then, we formulate an LCP by selectively linearizing this formulation about a reference trajectory. Next, we leverage the implicit-function theorem from Section III-D to differentiate through the solution to this LCP in order to provide gradients to the CI-MPC policy. Finally, we present a custom linear solver for the LCP that leverages offline pre-computation and partial factorizations for efficient online computation within the CI-MPC policy.

A. Contact Dynamics

To evaluate our contact dynamics, we need to jointly optimize the coupled impact and friction problems. We formulate a parameterized joint feasibility problem that simultaneously satisfies (5–12) for a fixed central-path parameter:

$$\text{find} \quad q_{t+1}, \lambda_t, \psi_t, \eta_t^{(1)}, \ldots, \eta_t^{(c)}, s_\phi, s_\psi \tag{18}$$

$$\begin{aligned} \text{s.t.} \quad & \Big(M(q_{t-1})(q_t - q_{t-1}) \\ & \quad - M(q_t)(q_{t+1} - q_t)\Big)/h \\ & \quad - hC(q_t, (q_{t+1} - q_t)/h) \\ & \quad + J(q_{t+1})^T \lambda_t + B(q_{t+1}) u_t = 0, \end{aligned} \tag{19}$$

$$s_\phi - \phi(q_{t+1}) = 0, \tag{20}$$

$$s_\psi^{(i)} - (\mu^{(i)} \gamma_t^{(i)} - \mathbf{1}^T \beta_t^{(i)}) = 0, \forall i, \tag{21}$$

$$\begin{aligned} & \eta_t^{(i)} - P^{(i)}(q_{t+1})(q_{t+1} - q_t)/h \\ & \quad - \psi_t^{(i)} \mathbf{1} = 0, \forall i, \end{aligned} \tag{22}$$

$$\gamma_t \circ s_\phi = \rho \mathbf{1}, \tag{23}$$

$$\psi_t \circ s_\psi = \rho \mathbf{1}, \tag{24}$$

$$\beta_t^{(i)} \circ \eta_t^{(i)} = \rho \mathbf{1}, \forall i, \tag{25}$$

$$\gamma_t, s_\phi, \psi_t, s_\psi \geq 0, \tag{26}$$

$$\beta_t^{(i)}, \eta_t^{(i)} \geq 0, \forall i. \tag{27}$$

The problem data include previous configurations, time step and control inputs. This feasibility problem (18) is closely related to prior work [34]. There, complementarity is strictly enforced at each iteration, i.e., $\rho = 0$, as a result of the underlying pivoting-based solver, whereas our formulation has relaxed bilinear constraints at each iteration that are successively decreased in order to achieve hard contact in the limit as the central-path parameter goes to zero. We introduce additional slack variables s_ϕ, $s_\psi \in \mathbf{R}^c$, a standard technique used with path-following methods and implicitly encode velocities via finite-difference approximations.

Algorithm 1 Differentiable Path-Following Method

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

B. Linearized Contact-Implicit Dynamics

The CI-MPC policy aims to track a reference trajectory comprising configurations $\bar{Q} = (\bar{q}_0, \ldots, \bar{q}_T)$, control inputs $\bar{U} = (\bar{u}_1, \ldots, \bar{u}_{T-1})$, and the contact forces $\bar{\Lambda} = (\bar{\lambda}_1, \ldots, \bar{\lambda}_{T-1})$.

Similar to linear MPC, we reduce the online computational burden by utilizing simplified dynamics. Specifically, we formulate time-varying contact-implicit dynamics which comprise the manipulator dynamics, signed-distance function, and maximum-dissipation-principle terms linearized about the reference trajectory, while the key contact dynamics modeled with complementarity and inequality constraints are retained. At each time step along the trajectory, we formulate the following LCP:

$$\begin{aligned} \text{find} \quad & w \\ \text{subject to} \quad & C(w - \bar{w}) + D(\theta - \bar{\theta}) = 0 \\ & \gamma \circ s_\phi = \rho\mathbf{1}, \\ & \psi \circ s_\psi = \rho\mathbf{1} \\ & \beta^{(i)} \circ \eta^{(i)} = \rho\mathbf{1}, \forall i \\ & \gamma, s_\phi, \psi, s_\psi \geq 0 \\ & \beta^{(i)}, \eta^{(i)} \geq 0, \forall i. \end{aligned} \tag{28}$$

with decision variables $w = (q_{t+1}, \lambda_t, \psi_t, \eta_t^{(1)}, \ldots, \eta_t^{(c)}, s_\phi, s_\psi)$. Here, $\bar{w}$ and $\bar{\theta}$ are reference decision variables and problem data, respectively, and C and D are matrices that define a linear system of equations and are pre-computed offline. These strategically linearized contact-implicit dynamics,

$$q_{t+1} = s_t(q_{t-1}, q_t, u_t), \tag{29}$$

$s_t : \mathbf{R}^n \times \mathbf{R}^n \times \mathbf{R}^m \to \mathbf{R}^n$, optimize (28) and return the configuration at the next time step. The contact forces at the current time step can also be returned.

This problem (28) is an LCP, however, it differs from the classic LCP formulation for contact dynamics [34]. In prior work, linearization is performed at some approximation of the next configuration. In the trajectory tracking setting, where the current and following configurations are known, we can linearize about the reference trajectory and not simply about the current configuration or a first-order approximation of the next configuration.

C. Differentiable Contact Dynamics

CI-MPC employs a bilevel-optimization scheme which evaluates the contact dynamics and their gradients for an upper-level trajectory-optimization problem by solving lower-level optimization problems. Evaluation of the contact dynamics is done by optimizing an LCP (28); gradients of the contact dynamics are then obtained by differentiating the solution to this problem using the implicit-function theorem [2, 35].

We define the problem data for (28) as $\theta = (q_{t-1}, q_t, u_t)$, but could also include the time step, friction coefficients, central-path parameter, and other system values like masses or inertia terms. At a solution point, we can compute the gradients of the next configuration or contact forces with respect to the problem data, e.g., $\frac{\partial q_{t+1}}{\partial u_t}$, using the approach presented in Section III-D. The residual comprises the equality constraints of the LCP (28).

The Jacobian $\frac{\partial r}{\partial w}$ can be computed and factorized (e.g., generally with LU decomposition) in the process of optimizing (28). Thus, differentiating through the solution w^* requires additionally computing $\frac{\partial r}{\partial \theta}$ and performing p additional linear-system solves.

We note that this approach depends upon finding a local minimizer for the residual, which may not exist in general. Additionally, if $\frac{\partial r}{\partial w^*}$ is singular, we can at best find an approximate solution to (17), e.g., a least-squares solution. Finally, an alternative approach to differentiating through an optimization problem is to fix the number of update steps and then directly differentiate through an unrolled sequence of solver steps. We do not explore such an approach in this work.

Lastly, the central-path parameter determines the complementary slackness of the solution, i.e., this is a tuneable parameter for controlling the smoothness of the returned gradients. Gradients computed with large values of ρ will be smoother than those computed with small values of ρ, which more closely approximates a true subgradient at nondifferentiable points. These subgradients often fail to provide useful information through contact events, so this parameter will be used to provide CI-MPC with smooth gradients that provide information through contact events.

D. Linearized Contact-Implicit Dynamics Solver

The most expensive steps in evaluating the LCP and computing gradients of a solution are computing the solution to a linear system,

$$R_w \Delta w = r, \tag{30}$$

required by the path-following method in order to compute a new search direction, where we define r as the equality constraints from (28), $R_w = \frac{\partial r}{\partial w}$, and Δw as the step direction for this linear system.

To reduce the computational cost of this optimization, we exploit both the sparsity pattern and the property that most of R_w remains constant across iterations [41] and, therefore, can be pre-computed offline.

For clarity, we use the same notations as in Problem 13. We work with: $x = q_{t+1}$, $y = (\gamma_t, \psi_t, \beta_t^{(1)}, \ldots, \beta_t^{(c)})$,

and $z = (s_\phi, s_\psi, \eta_t^{(1)}, \ldots, \eta_t^{(c)})$, where $w = (x, y, z)$, and similarly, split the residual: $r = (r_x, r_y, r_z)$. The Jacobian's sparsity pattern is:

$$R_w = \begin{bmatrix} E & F & 0 \\ G & H & I \\ 0 & \mathbf{diag}(z) & \mathbf{diag}(y) \end{bmatrix}, \tag{31}$$

where I denotes the identity matrix.

By exploiting sparsity in the third row of (31), we can form the following condensed system:

$$\begin{bmatrix} E & F \\ G & \tilde{H} \end{bmatrix} \begin{bmatrix} \Delta x \\ \Delta y \end{bmatrix} = \begin{bmatrix} r_x \\ \tilde{r}_y \end{bmatrix} \Leftrightarrow \tilde{R}_w \, \Delta\tilde{w} = \tilde{r}, \tag{32}$$

where,

$$\tilde{H} = H - \mathbf{diag}(y^{-1} \circ z), \tag{33}$$

$$\tilde{r}_y = r_y - y^{-1} \circ r_z, \tag{34}$$

$$\Delta z = y^{-1} \circ (r_z - z \circ \Delta y), \tag{35}$$

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 optimize 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 (32),

$$Q, R \leftarrow \mathbf{qr}(\tilde{H} - GE^{-1}F), \tag{36}$$

and then solve for the search directions,

$$\Delta y = -R^{-1}Q^T(GE^{-1}r_x - \tilde{r}_y), \tag{37}$$

$$\Delta x = E^{-1}(r_x - F\Delta y). \tag{38}$$

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 (35).

For a system with configuration dimension n and c contact points, the computational complexity of solving (30) with a naive approach is $O\Big((n + 2cd)^3\Big)$. Our structure-exploiting approach is $O(8c^3d^3)$ during the online phase. In practice, this provides a factor of 15 speed-up for evaluating the linearized contact-implicit dynamics across all the robotic systems presented in this paper and, in turn, results in a factor of 2.5 overall speed-up for the CI-MPC policy.

V. Contact-Implicit Model-Predictive Control

In this section, we present the CI-MPC algorithm. Online, this algorithm efficiently optimizes an instantiation of (1) that utilizes differentiable contact dynamics. First, we formulate discrete-time dynamics for this problem which utilize our linearized contact dynamics (28) for tracking a reference trajectory. Next, we present a structure-exploiting linear-system solver for efficiently solving the upper-level trajectory-optimization problem. Finally, we summarize the algorithm and present a simple heuristic that enables the policy to robustly handle unmodeled terrain.

Algorithm 2 Contact-Implicit MPC

procedure MPC($\bar{Q}, \bar{U}, \bar{\Lambda}, H, \rho$)
  Offline Stage
  $(C_{1:T-1}, D_{1:T-1}) \leftarrow$ LINEARIZE($\bar{Q}, \bar{U}, \bar{\Lambda}, \rho$)
  Online Stage
  For $\tau = 1, \ldots, \infty$
    $u_1^* \leftarrow$ TRAJOPT(q_0, q_1, τ, H)
    $q_2 \leftarrow$ SYSTEM(q_0, q_1, u_1^*)
    $(q_0, q_1) \leftarrow (q_1, q_2)$
  End

A. Trajectory Optimization

Similar to linear MPC, we optimize a version of (1) with the time-varying linearized contact-implicit dynamics (28). Importantly, we employ a state representation, $x_t = (q_{t-1}^{(t)}, q_t^{(t)})$, comprising two configurations. This exposes Markov structure, enabling a custom linear-system solver to reduce the overall complexity of solving the trajectory-optimization problem.

The first state, x_1, is fixed and we optimize decision variables: $z = (u_1, x_2, \cdots, u_{T-1}, x_T) \in \mathbf{R}^{m(T-1)+2n(T-1)}$. The resulting dynamics constraints,

$$d_t(x_t, u_t, x_{t+1}) = \begin{bmatrix} q_t^{(t+1)} \\ q_{t+1}^{(t+1)} \end{bmatrix} - \begin{bmatrix} q_t^{(t)} \\ s_t(q_{t-1}^{(t)}, q_t^{(t)}, u_t) \end{bmatrix} = 0, \tag{39}$$

effectively propagate configurations across one time step and encode the contact-implicit dynamics. The dynamics constraints Jacobian,

$$C = \nabla d = \begin{bmatrix} -B_1 & I & 0 & 0 \\ 0 & -A_2 & -B_2 & I \\ 0 & 0 & 0 & \ddots \end{bmatrix}, \tag{40}$$

has associated one-step dynamics Jacobians,

$$A_t = \begin{bmatrix} 0 & I \\ \frac{\partial s_t}{\partial q_{t-1}} & \frac{\partial s_t}{\partial q_t} \end{bmatrix}, \; B_t = \begin{bmatrix} 0 \\ \frac{\partial s_t}{\partial u_t} \end{bmatrix}, \tag{41}$$

that have additional sparsity, where $d = (d_1, \cdots, d_{T-1}) \in \mathbf{R}^{2n(T-1)}$.

In order to track a reference trajectory, the following objective, $J : \mathbf{R}^{m(T-1)+2n(T-1)} \to \mathbf{R}$, with time-varying quadratic cost functions is used:

$$g_t(x_t, u_t) = (x_t - \bar{x}_t)^T Q_t (x_t - \bar{x}_t) + (u_t - \bar{u}_t)^T R_t (u_t - \bar{u}_t) \tag{42}$$

$$g_T(x_T) = (x_T - \bar{x}_T)^T Q_T (x_T - \bar{x}_T), \tag{43}$$

where $Q_t \in \mathbf{S}_{++}^{2n}$ and $R_t \in \mathbf{S}_{++}^{m}$ ensure that the Hessian of the objective is positive definite. Velocities are penalized using finite-difference approximations and because the problem is lifted by using states with two configurations, these costs do not introduce state coupling across time steps. The resulting Hessian of the objective,

$$\nabla^2 J = \begin{bmatrix} R_1 & 0 & 0 & 0 \\ 0 & Q_2 & 0 & 0 \\ 0 & 0 & R_2 & 0 \\ 0 & 0 & 0 & \ddots \end{bmatrix}, \tag{44}$$

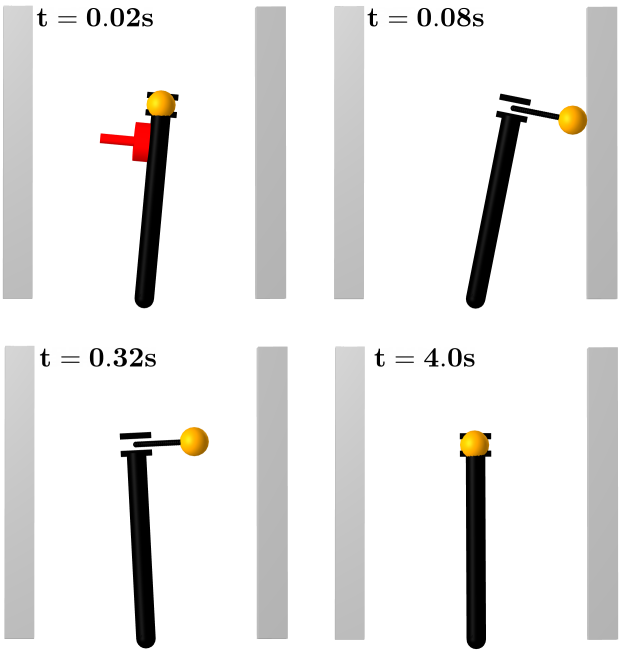

Fig. 2: PushBot performing push recovery. A disturbance (red) creates an impulse on the system and the policy generates a new contact sequence that extends the prismatic joint toward the right wall in order to make contact. After stabilizing, PushBot pushes against the wall, eventually breaking contact,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

and its inverse,

$$(\nabla^2 J)^{-1} = \begin{bmatrix} R_1^{-1} & 0 & 0 & 0 \\ 0 & Q_2^{-1} & 0 & 0 \\ 0 & 0 & R_2^{-1} & 0 \\ 0 & 0 & 0 & \ddots \end{bmatrix}, \tag{45}$$

are block diagonal and can be pre-computed offline.

The resulting system,

$$\begin{bmatrix} \nabla^2 J & C^T \\ C & 0 \end{bmatrix} \begin{bmatrix} \Delta z \\ \Delta \nu \end{bmatrix} = \begin{bmatrix} \nabla J + C^T \nu \\ d \end{bmatrix}, \tag{46}$$

which uses a Gauss-Newton approximation of the constraints when computing the Hessian of the Lagrangian, is solved using a block-elimination approach [37]. First, the Schur complement,

$$Y = C(\nabla^2 J)^{-1} C^T = \begin{bmatrix} Y_{11} & Y_{12} & 0 & 0 \\ Y_{21} & Y_{22} & Y_{23} & 0 \\ 0 & Y_{32} & Y_{33} & \dots \\ 0 & 0 & \vdots & \ddots \end{bmatrix}, \tag{47}$$

is formed blockwise,

$$Y_{11} = B_1 R_1^{-1} B_1^T, \tag{48}$$

$$Y_{tt} = B_t R_t^{-1} B_t^T + A_t P_t^{-1} A_t^T + P_{t+1}^{-1}, \quad t = 2, \cdots, T-1, \tag{49}$$

$$Y_{t,t+1} = Y_{t+1,t}{}^T = -P_{t+1}^{-1} A_{t+1}^T, \quad t = 1, \cdots, T-2. \tag{50}$$

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.

	replanning time	**real-time**
CI-MPC	0.014 ± 0.027 s	success
MIQP	0.18 ± 0.09 s	failure

Next, a Cholesky factorization, $Y = L^T L$, is similarly performed blockwise,

$$L_{11} L_{11}^T = Y_{11} \tag{51}$$

$$L_{tt} L_{t+1,t}^T = Y_{t,t+1}, \quad t = 1, \cdots, T-1, \tag{52}$$

$$L_{tt} L_{tt}^T = Y_{tt} - L_{t,t-1} L_{t,t-1}^T, \quad t = 2, \cdots, T-1. \tag{53}$$

Finally, block elimination is utilized to solve,

$$\Delta \nu = Y^{-1} \Big(C(\nabla^2 J)^{-1} (\nabla J + C^T \nu) - d \Big), \tag{54}$$

$$\Delta z = (\nabla^2 J)^{-1} \Big(\nabla J + C^T \nu - C^T \Delta \nu \Big). \tag{55}$$

Without exploiting the sparsity structure of the trajectory-optimization problem, the complexity of solving this instantiation of (1) with a generic LDL^T factorization is $O\Big(\frac{1}{3} T^3 (4n+m)^3\Big)$, while this specialized solver has an overall complexity: $O\Big(T(8n^3 + n^2 m)\Big)$.

B. Model-Predictive Control

The CI-MPC policy is comprised of offline and online stages and requires a reference trajectory, $(\bar{Q}, \bar{U}, \bar{\Lambda})$, and a planning horizon, H. During the offline stage, the necessary terms for (28) are computed for the given reference trajectory. Then, during the online stage, for a given initial state, comprising the current and previous configurations, (q_0, q_1), a control u_1^* is optimized by solving (1) over the MPC planning horizon, and this input is applied to the system. After the system evolves, the configurations are updated using the latest state information.

Additionally, like prior work that utilizes path-following methods for MPC [37], we employ a fixed central-path parameter for computational efficiency. In practice, we find that $\rho \approx$ 1e-4 is a good balance between computation time, physical accuracy, and gradient smoothness. Note that, when verifying the performance of the policy in simulation, we utilize the nonlinear contact dynamics (18) and tight tolerance for the central-path parameter, e.g., $\rho =$ 1e-6, to enforce hard contact, see Section VII for more details. The CI-MPC policy is summarized in Algorithm 2.

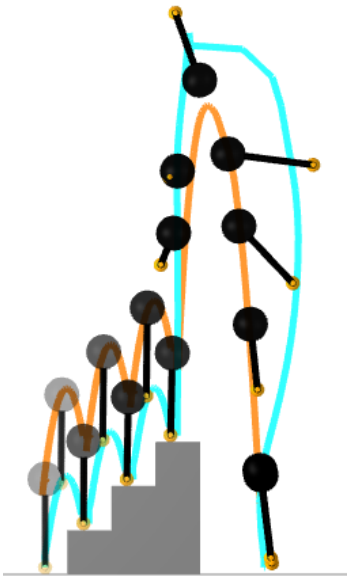

Fig. 3: Hopper in 2D performing parkour. The system tracks the body (orange) and foot (blue) reference trajectories while ascending three stairs before performing a front flip.

C. 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,

$$\phi_{\text{MPC}}(q) = \phi(q) + a, \tag{56}$$

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 (31) and only requires c more addition operations to compute r when optimizing the linearized contact-implicit dynamics problem (30). In our experiments, we find the heuristic to be effective and reliable across unknown terrain for the systems tested.

VI. Examples

We demonstrate the CI-MPC algorithm in simulation 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 IV.

For all the examples presented in this section, we provide a reference trajectory to the controller. These reference trajectories are designed using contact-implicit trajectory optimization [25] to generate gaits for the biped and quadruped, and templates for the hopper performing parkour. In the former case, the gait forms a limit cycle and is repeatedly for tracking, while in the latter case, two templates are generated and combined to form a composite reference trajectory. It is also possible to manually design trajectories, as was done for the hopper in 3D.

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.

	flat	**sinusoidal**	**piecewise**	**parkour**
CI-MPC	+100	+100	+100	success
Raibert	+100	+100	+100	failure

We verify the policy performance in simulation using the nonlinear contact dynamics (18). 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.

A. PushBot

In this example, we demonstrate that our policy can generate 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 [4] 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.

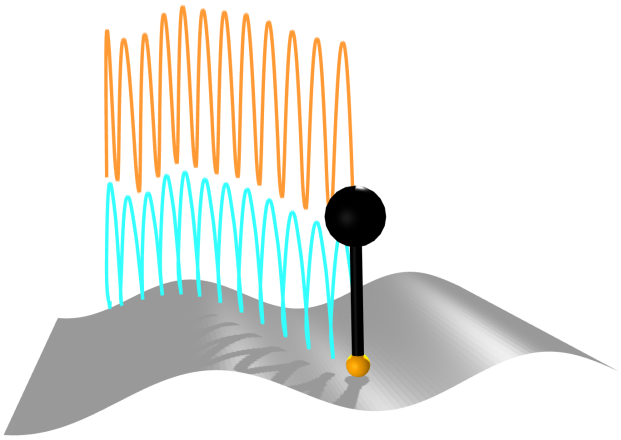

Fig. 4: Hopper in 3D tracks a hopping gait with body (orange) and foot (blue) trajectories over unmodeled and uneven terrain.

B. Hopper

Inspired by the Raibert Hopper [32], we model a 2D hopping robot with $n = 4$ 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.

Finally, we extend the Raibert hopper to 3D, for a system with $n = 7$ configuration variables: 3 position, 3 modified Rodrigues parameters for orientation, and 1 leg length; the $m = 3$ controls are: 2 body moments about the system's roll and pitch angles, and leg force. Again, foot and leg masses are assumed to be located at the body. We demonstrate that a policy tuned for flat ground can be employed, without retuning, for the hopper to locomote over an unknown terrain (Fig. 4).

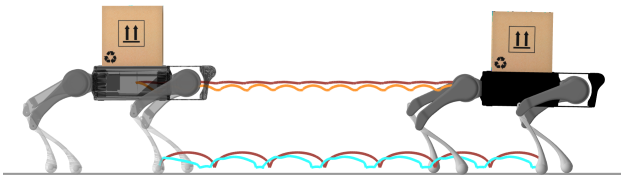

Fig. 5: Quadruped tracking a reference trajectory (red) while carrying an unmodeled 3-kg payload. We depict the torso (orange) and front-left foot (blue) trajectories.

C. Quadruped

We model a planar quadruped with $n = 11$ 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. 1). Additionally, we test the robustness of the MPC policy by introducing model mismatch. We provide the MPC 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 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.

D. Biped

We model a planar biped based on Spring Flamingo [31] with $n = 9$ 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 [31], which relies on a state-machine architecture and a number of proportional-derivative controllers. Our 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 [30].

E. Monte Carlo Initial Conditions

In order to assess the robustness of CI-MPC, we perform Monte Carlo analysis on two systems: the hopper (2D) and the 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 that the quadruped is robust to large drops (Fig. 7).

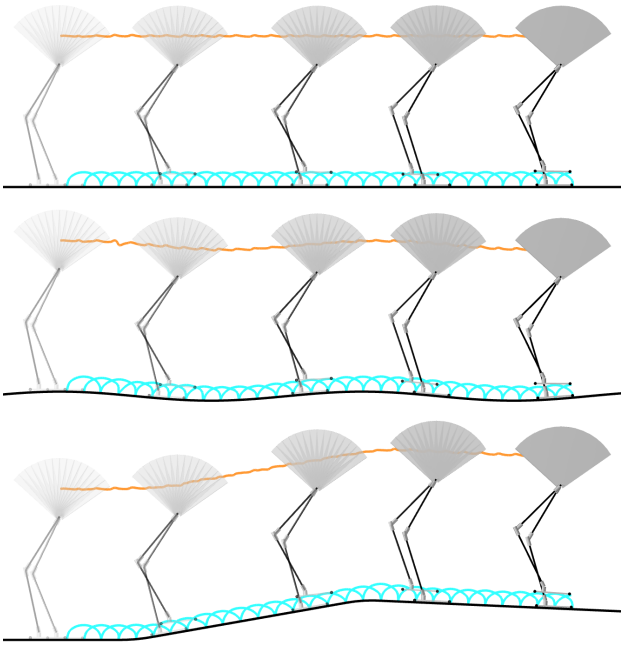

Fig. 6: Biped walking from left to right across flat (top), sinusoidal (middle), and piecewise linear (bottom) terrain using the same policy.

VII. Discussion

CI-MPC is capable of robustly tracking reference trajectories through contact despite disturbances, model mismatch, and uncertain environments. Additionally, we demonstrate that these policies are capable of generating new contact sequences online that are qualitatively distinct from their reference trajectory.

An important approximation we make for the online trajectory-optimization problem is using the strategically linearized contact-implicit dynamics in the policy. This enables expensive gradient computations and partial matrix factorizations to be performed in an offline stage, in order to substantially reduce online computation. Despite the approximations introduced by these simplifications, in practice, the controls optimized for the linearized contact-implicit dynamics work well in simulation for the nonlinear contact dynamics and we have a demonstrably robust tracking controller that works through contact events for a collection of different robotic systems.

The physics of hard contact produces non-smooth and discontinuous gradients. With our path-following method, we can directly control the smoothness of the gradients in a principled way via the central-path parameter. For simulation, this tolerance is set to $\rho_{\text{sim}} = 10^{-6}$ in order to produce realistic hard contact. For the MPC policy, the parameter is fixed to reduce online computation and the value, set to $\rho_{\text{MPC}} = 10^{-4}$ for all of the examples, was selected empirically to balance capturing accurate physics with producing usefully smooth gradients. Indeed, we prioritize fast MPC updates by solving the trajectory-tracking problem to coarse tolerances. In this context, imposing very accurate physics would be wasteful in terms of computational resources. Additionally, we observed that using smoother gradients enhanced the convergence of the trajectory-tracking solver and likely enables the policy to more easily discover new contact sequences.

TABLE III: Comparison between CI-MPC and Pratt state-machine [31] 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† [30].

	flat	**incline**
CI-MPC	$+100$	10 deg.
Pratt	$+100^{\dagger}$	$5^{\dagger}$ deg.

In the examples, we compare our policy to a number of system-specific control strategies. We note that these heuristics work quite well on the particular system for which they were designed, but that the approaches do not generalize to other systems. For example, the Raibert hopping controller does not transfer to the PushBot push-recovery scenario and the Spring Flamingo walking controller is not directly applicable to the quadruped. In contrast to these system-specific approaches, our CI-MPC approach is successfully deployed to all the systems in this work. Similar to the abilitiy of linear MPC to track a reference trajectory for a wide range of systems with smooth dynamics, CI-MPC is able to track reference trajectories for a variety of systems with nonsmooth dynamics that make and break contact with their environments.

Contact-implicit trajectory optimization [29, 25] is notorious for poor convergence properties, despite typically relying on robust large-scale constrained solvers for nonlinear programming and even good warm starting. As a result, MPC with *nonlinear* contact dynamics is generally impractical. In this work, we demonstrate that contact-implicit MPC can be reliable, by utilizing selectively linearized contact dynamics; converging despite disturbances and initial conditions with large offsets. In practice, we find that the one-step contact-dynamics problem always converges, largely enabled by a numerically robust path-following method. Additionally, our bilevel-optimization approach for MPC abstracts the lower-level contact-dynamics inequality and complementarity constraints away from the upper-level trajectory optimization problem, significantly improving convergence behavior.

VIII. Conclusions and Future Work

In summary, we have presented 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, with a more efficient implementation, and potential use of parallelization and hardware acceleration, we expect CI-MPC can run in real-time onboard quadruped and biped hardware. Second, instead of linearization, it should be possible to perform higher-fidelity convex MPC online by utilizing the nonlinear second-order friction cone instead of its linearized approximation. Next, an interesting comparison for CI-MPC would be against a nonlin-

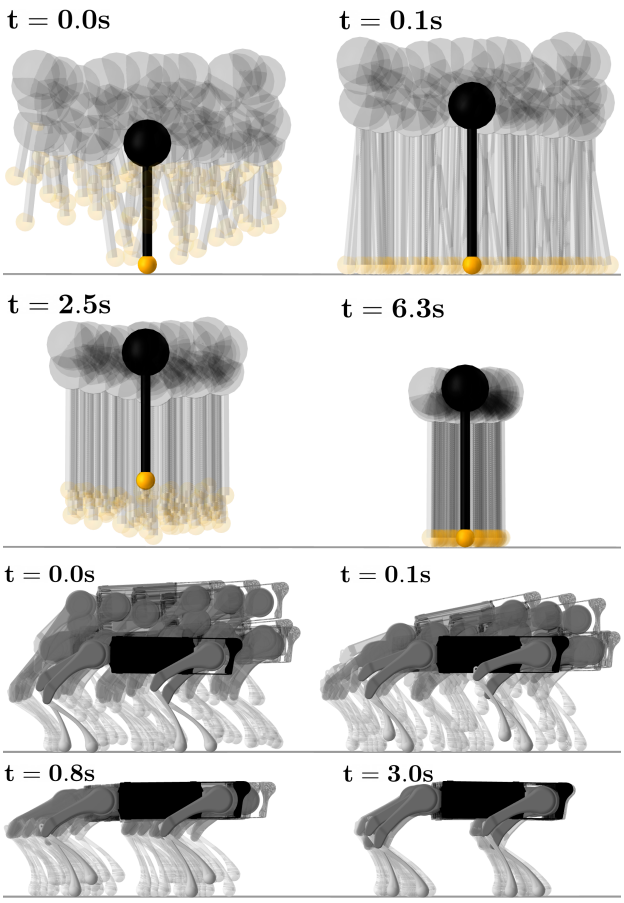

Fig. 7: Monte Carlo simulations of initial conditions for systems tracking a reference trajectory. 100 initial configurations are randomly sampled for a hopper (top) and quadruped (bottom). Perturbations from the reference initial configuration include large translations, tilts, and joint angles offsets.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

ear MPC implementation in terms of reliability and ability to generate dynamic behaviors. A natural extension of this work, which was focused on locomotion, is to the manipulation domain where control through contact is similarly an open problem. Finally, a library of templates, comprising reference trajectories and associated MPC policies, could be assembled to enable more diverse behavior online and Markov decision processes could be optimized 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.

TABLE IV: CI-MPC runs at a fixed rate corresponding to the reference trajectory's rate f_{ref}. For all robotic systems, the policy runs in real-time, meaning that the time required to compute the control is always smaller than the time budget $1/f_{\text{ref}}$. All experiments are run on a computer equipped with an Intel Core i7-8750H processor.

system	planning horizon	rate
pushbot	1.60 s	25 Hz
hopper (2D)	0.10 s	100 Hz
hopper (3D)	0.20 s	100 Hz
quadruped	0.16 s	64 Hz
biped	0.23 s	64 Hz

REFERENCES

[1] Aaron D. Ames, Kevin Galloway, Koushil Sreenath, and Jessy W. Grizzle. Rapidly exponentially stabilizing control Lyapunov functions and hybrid zero dynamics. *IEEE Transactions on Automatic Control*, 59(4):876–891, 2014.

[2] Brandon Amos and J. Zico Kolter. Optnet: Differentiable optimization as a layer in neural networks. In *International Conference on Machine Learning*, pages 136–145. PMLR, 2017.

[3] Alp Aydinoglu, Victor M. Preciado, and Michael Posa. Contact-aware controller design for complementarity systems. In *2020 IEEE International Conference on Robotics and Automation (ICRA)*, pages 1525–1531. IEEE, 2020.

[4] Alberto Bemporad and Manfred Morari. Control of systems integrating logic, dynamics, and constraints. *Automatica*, 35(3):407–427, 1999.

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

[6] Matthew Chignoli, Donghyun Kim, Elijah Stanger-Jones, and Sangbae Kim. The MIT humanoid robot: Design, motion planning, and control for acrobatic behaviors. *arXiv preprint arXiv:2104.09025*, 2021.

[7] Richard W. Cottle, Jong-Shi Pang, and Richard E. Stone. *The linear complementarity problem*. SIAM, 2009.

[8] Hongkai Dai, Andrés Valenzuela, and Russ Tedrake. Whole-body motion planning with centroidal dynamics and full kinematics. In *2014 IEEE-RAS International Conference on Humanoid Robots*, pages 295–302. IEEE, 2014.

[9] Filipe de Avila Belbute-Peres, Kevin Smith, Kelsey Allen, Josh Tenenbaum, and J. Zico Kolter. End-to-end differentiable physics for learning and control. *Advances in Neural Information Processing Systems*, 31:7178–7189, 2018.

[10] Ulisse Dini. *Lezioni di analisi infinitesimale*, volume 1. Fratelli Nistri, 1907.

[11] Steven P. Dirkse and Michael C. Ferris. The PATH Solver: A nommonotone stabilization scheme for mixed complementarity problems. *Optimization Methods and Software*, 5(2):123–156, 1995.

[12] Evan Drumwright. Rapidly computable viscous fric-

tion and no-slip rigid contact models. *arXiv preprint arXiv:1504.00719*, 2015.

[13] Boston Dynamics. More Parkour Atlas, 2019. URL https://www.youtube.com/watch?v=_sBBaNYex3E.

[14] Moritz Geilinger, David Hahn, Jonas Zehnder, Moritz Bächer, Bernhard Thomaszewski, and Stelian Coros. ADD: Analytically differentiable dynamics for multi-body systems with frictional contact. *ACM Transactions on Graphics (TOG)*, 39(6):1–15, 2020.

[15] Nicolas Heess, Dhruva TB, Srinivasan Sriram, Jay Lemmon, Josh Merel, Greg Wayne, Yuval Tassa, Tom Erez, Ziyu Wang, S. M. Eslami, et al. Emergence of locomotion behaviours in rich environments. *arXiv preprint arXiv:1707.02286*, 2017.

[16] Eric Heiden, David Millard, Erwin Coumans, Yizhou Sheng, and Gaurav S. Sukhatme. Neuralsim: Augmenting differentiable simulators with neural networks. *arXiv preprint arXiv:2011.04217*, 2020.

[17] Kazuo Hirai, Masato Hirose, Yuji Haikawa, and Toru Takenaka. The development of Honda humanoid robot. In *Proceedings. 1998 IEEE International Conference on Robotics and Automation*, volume 2, pages 1321–1326. IEEE, 1998.

[18] Donghyun Kim, Jared Di Carlo, Benjamin Katz, Gerardo Bledt, and Sangbae Kim. Highly dynamic quadruped locomotion via whole-body impulse control and model predictive control. *arXiv preprint arXiv:1909.06586*, 2019.

[19] Jonas Koenemann, Andrea Del Prete, Yuval Tassa, Emanuel Todorov, Olivier Stasse, Maren Bennewitz, and Nicolas Mansard. Whole-body model-predictive control applied to the HRP-2 humanoid. In *2015 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 3346–3351. IEEE, 2015.

[20] Masakazu Kojima, Nimrod Megiddo, Toshihito Noma, and Akiko Yoshise. A unified approach to interior point algorithms for linear complementarity problems: A summary. *Operations Research Letters*, 10(5):247–254, 1991.

[21] Scott Kuindersma, Robin Deits, Maurice Fallon, Andrés Valenzuela, Hongkai Dai, Frank Permenter, Twan Koolen, Pat Marion, and Russ Tedrake. Optimization-based locomotion planning, estimation, and control design for the Atlas humanoid robot. *Autonomous Robots*, 40(3):429–455, 2016.

[22] Jeongseok Lee, Michael X. Grey, Sehoon Ha, Tobias Kunz, Sumit Jain, Yuting Ye, Siddhartha S. Srinivasa, Mike Stilman, and C. Karen Liu. DART: Dynamic animation and robotics toolkit. *Journal of Open Source Software*, 3(22):500, 2018.

[23] Joonho Lee, Jemin Hwangbo, Lorenz Wellhausen, Vladlen Koltun, and Marco Hutter. Learning quadrupedal locomotion over challenging terrain. *Science Robotics*, 5 (47), 2020.

[24] Zhongyu Li, Xuxin Cheng, Xue Bin Peng, Pieter Abbeel, Sergey Levine, Glen Berseth, and Koushil Sreenath. Reinforcement learning for robust parameterized locomotion control of bipedal robots. *arXiv preprint arXiv:2103.14295*, 2021.

[25] Zachary Manchester and Scott Kuindersma. Variational contact-implicit trajectory optimization. In *Robotics Research*, pages 985–1000. Springer, 2020.

[26] Jean Jacques Moreau. On unilateral constraints, friction and plasticity. In *New Variational Techniques in Mathematical Physics*, pages 171–322. Springer, 2011.

[27] Michael Neunert, Markus Stäuble, Markus Giftthaler, Carmine D. Bellicoso, Jan Carius, Christian Gehring, Marco Hutter, and Jonas Buchli. Whole-body nonlinear model predictive control through contacts for quadrupeds. *IEEE Robotics and Automation Letters*, 3 (3):1458–1465, 2018.

[28] Jorge Nocedal and Stephen J. Wright. *Numerical Optimization*. Springer, second edition, 2006.

[29] Michael Posa, Cecilia Cantu, and Russ Tedrake. A direct method for trajectory optimization of rigid bodies through contact. *The International Journal of Robotics Research*, 33(1):69–81, 2014.

[30] Jerry Pratt and Gill Pratt. Intuitive control of a planar bipedal walking robot. In *Proceedings. 1998 IEEE International Conference on Robotics and Automation (Cat. No. 98CH36146)*, volume 3, pages 2014–2021. IEEE, 1998.

[31] Jerry E. Pratt. *Exploiting Inherent Robustness and Natural Dynamics in the Control of Bipedal Walking Robots*. PhD thesis, Massachusetts Institute of Technology, May 2000.

[32] Marc H. Raibert, H. Benjamin Brown Jr., Michael Chepponis, Jeff Koechling, Jessica K. Hodgins, Diane Dustman, W. Kevin Brennan, David S. Barrett, Clay M. Thompson, John Daniell Hebert, Woojin Lee, and Borvansky Lance. Dynamically stable legged locomotion. Technical report, Massachusetts Institute of Technology Cambridge Artificial Intelligence Lab, 1989.

[33] Jean-Pierre Sleiman, Farbod Farshidian, Maria Vittoria Minniti, and Marco Hutter. A unified mpc framework for whole-body dynamic locomotion and manipulation. *IEEE Robotics and Automation Letters*, 6(3):4688–4695, 2021.

[34] David E. Stewart and Jeffrey C. Trinkle. An implicit time-stepping scheme for rigid body dynamics with inelastic collisions and coulomb friction. *International Journal for Numerical Methods in Engineering*, 39(15): 2673–2691, 1996.

[35] Mandar Thombre, Zhou Joyce Yu, Johannes Jäschke, and Lorenz T. Biegler. Sensitivity-assisted multistage nonlinear model predictive control: Robustness, stability and computational efficiency. *Computers & Chemical Engineering*, 148:107269, 2021.

[36] Emanuel Todorov, Tom Erez, and Yuval Tassa. MuJoCo: A physics engine for model-based control. In *2012 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 5026–5033. IEEE, 2012.

[37] Yang Wang and Stephen Boyd. Fast model predictive control using online optimization. *IEEE Transactions on Control Systems Technology*, 18(2):267–278, 2009.

[38] Keenon Werling, Dalton Omens, Jeongseok Lee, Ioannis

Exarchos, and C. Karen Liu. Fast and feature-complete differentiable physics for articulated rigid bodies with contact. *arXiv preprint arXiv:2103.16021*, 2021.

[39] Eric R. Westervelt, Jessy W. Grizzle, and Daniel E. Koditschek. Hybrid zero dynamics of planar biped walkers. *IEEE Transactions on Automatic Control*, 48 (1):42–56, 2003.

[40] Alexander W. Winkler, C. Dario Bellicoso, Marco Hutter, and Jonas Buchli. Gait and Trajectory Optimization for Legged Systems Through Phase-Based End-Effector Parameterization. *IEEE Robotics and Automation Letters*, 3(3):1560–1567, July 2018.

[41] Ichitaro Yamazaki, Saeid Nooshabadi, Stanimire Tomov, and Jack Dongarra. Structure-Aware Linear Solver for Realtime Convex Optimization for Embedded Systems. *IEEE Embedded Systems Letters*, 9(3):61–64, September 2017.

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 optimal control and game-theoretic optimization for robotic 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 optimization and control for robotic systems.

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.