Linear 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: linear contact-implicit model-predictive control (LCI-MPC). Our use of differentiable contact dynamics provides a natural extension of linear model-predictive control to contact-rich settings. The policy leverages precomputed linearizations about a reference state or trajectory while contact modes, encoded via complementarity constraints, are explicitly retained, resulting in policies that can be efficiently evaluated while maintaining robustness to changes in contact timings. In many cases, the algorithm is even capable of generating entirely new contact sequences. To enable real-time performance, we devise a custom structure-exploiting linear solver for the contact dynamics. We demonstrate that the policy can respond to disturbances by discovering and exploiting new contact modes and is robust to model mismatch and unmodeled environments for a collection of simulated robotic systems, including: pushbot, hopper, quadruped, and biped.**

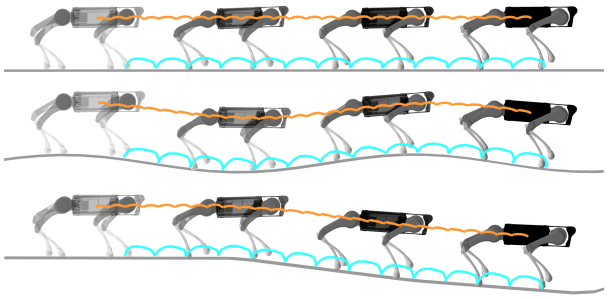

Fig. 1: Planar quadruped walking over uneven terrain. The reference gait is optimized for flat ground and the MPC, with orange center of mass and blue foot position trajectories, is able to adapt online to the unmodeled variation in height.

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 [38, 2, 19], to complementarity controllers [4], to neural-network policies [11, 12], and model-predictive control (MPC) [39, 32]. There have also been numerous successes deploying such approaches on complex systems in recent years: direct trajectory optimization and LQR on Atlas [16], smooth-contact models and differential dynamic programming on HRP-2 [35, 15], zero-moment point and feedback linearization on ASIMO [13], and MPC with simplified dynamics models on Cheetah [5] and ANYmal [18]. 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 unilateral contact interactions with their environments. Our approach combines a differentiable "hard-contact" rigid-body dynamics formulation with strategic linearizations and specialized numerical optimization techniques to develop 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 feasibility problem that simultaneously satisfies impact and friction constraints. By employing a primal-dual interior-point method to solve this problem, we naturally and reliably converge from "soft" to "hard" contact. At a solution, the implicit function theorem is utilized to efficiently compute dynamics derivatives for use in the controller. For online model-predictive control, we then use key precomputed linearizations about a reference state or trajectory while maintaining nonlinear complementarity constraints that encode information about contacts. We

[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

[*] These authors contributed equally to this work.

refer to this algorithm as *linear contact-implicit model-predictive control* (LCI-MPC).

Finally, we demonstrate that our LCI-MPC controller 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, quadruped, and biped.

Our specific contributions are:

- A contact dynamics formulation that can be reliably evaluated with a primal-dual interior-point method, and can be efficiently differentiated
- A model-predictive-control framework that utilizes strategic linearization of dynamics and constraints for computational efficiency while preserving information about contacts via nonlinear complementarity constraints
- An efficient structure-exploiting linear-system solver for linear contact-implicit dynamics that enables real-time performance of the LCI-MPC algorithm
- A collection of simulation examples demonstrating the performance of the LCI-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 prior work related to controlling systems that experience contact events with MPC; outline a classic contact-dynamics formulation; present a brief overview of linear MPC; and provide background on interior-point methods and how to differentiate through their solutions in Section II. Next, we present our differentiable contact dynamics in Section III. In Section IV, we present our LCI-MPC framework including: a feasibility problem for linear contact-implicit dynamics, an efficient trajectory-optimization formulation that utilizes these dynamics, and key implementation details. In Section V, we provide simulation results using the nonlinear contact dynamics to demonstrate the performance of the LCI-MPC policy. Finally, we discuss our results in Section VI and conclude with directions for future work in Section VII.

II. Background

In this section, we review related work on MPC and contact dynamics, provide background on linear MPC, and present an overview of interior-point methods and an approach for differentiating through their solutions.

A. Related Work

Today, most successful approaches for controlling legged robots utilize MPC in combination with simplified models and ideas originally developed by Raibert for hopping robots [30]. The key insight of this work is that the control problem can be decoupled into a high-level controller that plans body motions based on centroidal dynamics that ignore the details of the leg dynamics, and a low-level controller that applies joint torques to the legs to generate the desired body forces from 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 [7] and perform parkour [9]. 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 also been achieved at real-time rates on hardware [26]. Approaches that utilize both force-based MPC and whole-body control have also demonstrated agile locomotion [14].

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 LCI-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. Contact Dynamics

The classic approach for simulating rigid-body systems that experience contact is a velocity-based time-stepping scheme that solves a linear complementarity problem (LCP) in order to find the systems's next configuration, $q_{t+1} \in \mathbf{R}^n$, and velocity, $v_{t+1} \in \mathbf{R}^n$ [33]. The LCP is comprised of impact and friction subproblems.

A system is defined 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 $J : \mathbf{R}^n \to \mathbf{R}^l$ 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 controls inputs, often torques, into the generatlized coordinates; time step $h \in \mathbf{R}_+$; contact forces $\lambda_t = (\gamma_t^{(1)}, \beta_t^{(1)}, \ldots, \gamma_t^{(c)}, \beta_t^{(c)}) \in \mathbb{R}^l$ defined in the local surface frame, where $l = c(2d-1)$, with impact magnitude, $\gamma_t^{(i)} \in \mathbf{R}$, and a double parameterized friction-force vector, $\beta_t^{(i)} \in \mathbf{R}^{2(d-1)}$; signed-

distance function, $\phi : \mathbf{R}^n \to \mathbf{R}^c$, that returns distances between between specified contact points on the robot, e.g., feet, and the closest surface in the environment, e.g., the floor, where c is the number of contact points; and the environment dimension is $d = 2$ for planar systems and $d = 3$ otherwise.

The impact problem,

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

$$q_{t+1} = q_t + hv_{t+1}, \tag{2}$$

$$\gamma_t^T \phi(q_{t+1}) = 0, \tag{3}$$

$$\gamma_t,\ \phi(q_{t+1}) \geq 0, \tag{4}$$

includes: manipulator dynamics discretized with a semi-implicit Euler scheme (1), finite-difference velocities (2), complementarity constraints encoding physical behavior that impact impulses can only be imparted to the system while in contact (22), and inequality constraints modeling repulsive impact impulses and non-penetration of rigid bodies (23).

The friction problem models Coulomb friction at each contact point using the maximum-dissipation principle [24] and is encoded with the following constraints,

$$\eta - \begin{bmatrix} v^T & -v^T \end{bmatrix}^T - \psi \mathbf{1} = 0 \tag{5}$$

$$\psi \cdot (\mu\gamma - \mathbf{1}^T \beta) = 0 \tag{6}$$

$$\beta^T \eta = 0 \tag{7}$$

$$\beta, \eta \geq 0, \tag{8}$$

where $v \in \mathbf{R}^{d-1}$ is the tangential velocity at a contact point, $\mu \in \mathbf{R}_+$ is the coefficient of friction, $\psi \in \mathbf{R}$ is the dual variable associated with a linearized friction-cone constraint, and $\eta \in \mathbf{R}^{2(d-1)}$ is the dual variable associated with the nonnegative friction-force constraint.

The impact and friction problems are coupled through the contact forces and the system's configuration and velocity, which is mapped through the contact Jacobian in order to compute the tangential velocities. The LCP jointly solves the impact and friction problems, enabling one-step simulation of a rigid-body systems that experience hard contact.

For trajectory generation, these constraints have been directly used to encode contact dynamics with collocation techniques in a large nonlinear programs in order to optimize trajectories without pre-specified mode sequences for locomotion and simple manipulation tasks [28]. Subsequent work improved this approach by introducing higher-order integrators for the dynamics (1) and a numerically robust, exact ℓ_1-penalty for handling the complementarity constraints [21]. Alternative gradient-based approaches that utilize simulator rollouts differentiate through the solution of a one-step LCP [8, 37].

Another popular contact-dynamics formulation is MuJoCo's [35] soft-contact model, which trades physical realism for fast and reliable performance, and 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 [10]. We do not explore soft-contact models in this work.

C. Model-Predictive Control

MPC aims to solve the following trajectory optimization problem,

$$\begin{array}{ll} \underset{x_{1:T}, u_{1:T-1}}{\text{minimize}} & g_T(x_T) + \sum_{t=1}^{T-1} g_t(x_t, u_t) \\ \text{subject to} & x_{t+1} = f_t(x_t, u_t), \quad t = 1, \ldots, T-1, \\ & (x_1 \text{ given}). \end{array} \tag{9}$$

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 solve is often replaced with a proxy that has linear dynamics,

$$x_{t+1} = A_t x_t + B_t u_t, \tag{10}$$

and quadratic cost functions,

$$g_T(x_T) = x_T^T Q_T x_T + q_T^T x_T, \tag{11}$$

$$g_t(x_t, u_t) = x_t^T Q_t x_t + q_t^T x_t + u_t^T R_t u_t + r_t^T u_t. \tag{12}$$

This formulation is commonly referred to as *linear* MPC.

After optimizing our instantiation of (9), the first control, u_1^*, is applied to the system. After the system evolves, x_1 is updated and (9) is resolved in order to compute a new control input for the system. By repeating this procedure, feedback control is achieved [36]. In practice, applying the controls optimized for our proxy problem, with linear dynamics and quadratics costs, to the actual nonlinear system is extremely effective, especially for applications that track a reference trajectory.

In order to solve (9) 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.

D. Interior-Point Methods

Interior-point methods can efficiently and reliably solve optimization problems with inequality constraints [27]. A problem,

$$\begin{aligned} \underset{x}{\text{minimize}} \quad & f(x;\theta) \\ \text{subject to} \quad & g(x;\theta) = 0, \\ & x \geq 0, \end{aligned} \tag{13}$$

with decision variables $x \in \mathbf{R}^n$, problem data $\theta \in \mathbf{R}^p$, objective $f : \mathbf{R}^n \times \mathbf{R}^p \to \mathbf{R}$, and equality constraints $g : \mathbf{R}^n \times \mathbf{R}^p \to \mathbf{R}^m$, can be rewritten with a logarithmic barrier in order to handle the inequality constraints as follows:

$$\begin{aligned} \underset{x}{\text{minimize}} \quad & f(x;\theta) - \rho \sum_{i=1}^{n} \log(x^{(i)}) \\ \text{subject to} \quad & g(x;\theta) = 0, \end{aligned} \tag{14}$$

and then solved by finding solutions to a sequence of barrier subproblems as the central-path parameter $\rho \to 0$. The optimality conditions for (14) are:

$$\nabla_x f(x;\theta) + \nabla_x g(x;\theta)^T y - z = 0, \tag{15}$$

$$g(x;\theta) = 0, \tag{16}$$

$$x \circ z = \rho \mathbf{1}, \tag{17}$$

$$x, z \geq 0, \tag{18}$$

where $y \in \mathbf{R}^m, z \in \mathbf{R}^n$ are dual variables associated with the equality and inequality constraints, respectively, $\circ$ is an element-wise (Hadamard) vector product, and $\mathbf{1}$ is a vector of ones.

The equality constraints (15-17) form a residual vector or solution map, $r : \mathbf{R}^{n+m+n} \times \mathbf{R}^p \to \mathbf{R}^{n+m+n}$, that takes $w = (x, y, z) \in \mathbf{R}^{n+m+n}$ and the problem data as inputs. These problem data are fixed parameters during optimization, e.g., bounds on decision variables or weights in the objective. 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 (18) are strictly satisfied for candidate points at each iteration. Once the optimality conditions (15-18) are solved to a desired tolerance, the central-path parameter is decreased and the new subproblem is warm-started with the current solution and then solved. This procedure is repeated until the central-path parameter, also referred to as complementary slackness, is below a desired tolerance.

Algorithm 1 Differentiable Interior-Point Method

procedure INTERIORPOINT(x, θ)
  Parameters: $\beta, \gamma, \epsilon_p, \epsilon_r$
  Initialize: y, z, ρ
  Until $\rho < \epsilon_\rho$ **do**
    $\Delta w = (\Delta x, \Delta y, \Delta z) = (\frac{\partial r}{\partial w})^{-1} r(w;\theta)$
    $\alpha \leftarrow 1$
    Until $x - \alpha \Delta x > 0$ **do** $\alpha \leftarrow \beta\alpha$
    Until $\|r(w - \alpha\Delta w;\theta)\| < \|r(w;\theta)\|$ **do**
      $\alpha \leftarrow \beta\alpha$
    $w \leftarrow w - \alpha \Delta w$
    If $\|r(w;\theta)\| < \epsilon_r$ **do** $\rho \leftarrow \gamma\rho$
  $\frac{\partial w}{\partial \theta} \leftarrow$ DIFFERENTIATE(w, θ) ▷ Eq. 19
  Return $w, \frac{\partial w}{\partial \theta}$

E. Differentiating Solutions of Optimization Problems

At a solution point $w^* = (x^*, y^*, z^*)$ of (14), the residual $r(w^*;\theta) = 0$ and it is possible to differentiate through the solution map with respect to the problem data [1, 3]. We follow the approach from [3, 34] and directly differentiate through the residual using the implicit-function theorem:

$$\frac{\partial w^*}{\partial \theta} = -\Big(\frac{\partial r}{\partial w^*}\Big)^{-1} \frac{\partial r}{\partial \theta}. \tag{19}$$

The Jacobian $\frac{\partial r}{\partial w^*}$ can be computed and factorized (e.g., generally with LU decomposition) in the process of solving (13). Thus, differentiating through the solution w^* requires additionally computing $\frac{\partial r}{\partial \theta}$ and performing p additional linear-system solves. The differentiable interior-point method is summarized in Algorithm 1.

We note that this approach depends upon finding a local minimizer for (13), which may not exist in general. Additionally, if $\frac{\partial r}{\partial w^*}$ is singular, we can at best find an approximate solution to (19), 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.

III. Differentiable Contact Dynamics

In order to perform fast MPC with Newton or quasi-Newton methods, we need dynamics that can be efficiently evaluated and differentiated. For many robotic systems, e.g., serial-link manipulators or quadrotors, dynamics are typically derived using the Euler-Lagrange equations; evaluation of the resulting equality constraints and computation of their derivatives can be efficiently performed analytically.

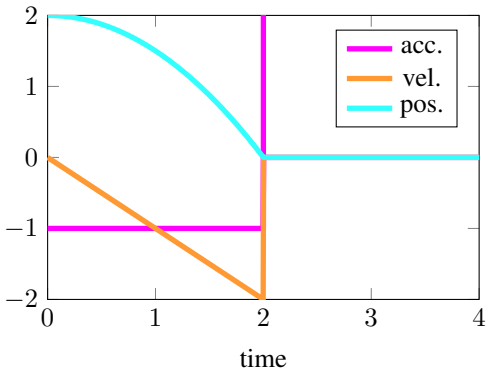

Fig. 2: Acceleration (magenta), velocity (orange), and position (blue) trajectories from simulation dropping a particle on a flat surface. The acceleration has a large impulse, velocity a discontinuous jump, while position is continuous but nonsmooth.

Contact dynamics are more complex; to capture non-smooth behavior like friction and impacts, dynamics evaluations require solving an optimization problem, and differentiation is performed via operations on the corresponding solution map.

In this work, we formulate contact dynamics as a joint feasibility problem over impact and friction optimization problems [33, 21]. Both problems are first converted to an interior-point form as described in Section II-D. A joint feasiblility problem is then solved by an interior-point method that 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.

A. Impact

Rigid-body systems experience acceleration impulses due to impact, as shown in Fig. 2. As a result, simulation of such systems has typically been performed at the velocity level [33]. However, velocity trajectories will exhibit jump discontinuities, an undesirable behavior for control applications. We instead work exclusively at the configuration level, utilizing a state representation comprising two configurations, and implicitly representing velocities using finite-difference approximations. While still nonsmooth, our system trajectories no longer have large, potentially infinite, impulses or discontinuities.

Just as the dynamics of robotic systems with smooth dynamics are derived from the Euler-Lagrange equations using the principle of least action [17], we can formulate a *constrained* least-action problem,

$$\begin{array}{ll} \underset{q_{t+1}}{\text{minimize}} & \mathcal{A}(q_{t+1}) \\ \text{subject to} & \phi(q_{t+1}) \geq 0, \end{array} \tag{20}$$

with inequality constraints in order to model impact dynamics [23], where the objective, $\mathcal{A} : \mathbf{R}^n \to \mathbf{R}$, depends on the Lagrangian of the system and a numerical integration scheme. Unlike prior work [21], which used the optimality conditions of the Lagrangian form of (20) to model impact dynamics, we utilize the interior-point form.

The interior-point optimality conditions for (20) are:

$$\begin{aligned} &\Big(M(q_{t-1})(q_t - q_{t-1}) \\ &\quad - M(q_t)(q_{t+1} - q_t)\Big)/h \\ &\quad - hC(q_t, (q_{t+1} - q_t)/h) \\ &\quad + J(q_{t+1})^T \lambda_t + B(q_{t+1})u_t = 0, \end{aligned} \tag{21}$$

$$\gamma_t \circ \phi(q_{t+1}) = \rho \mathbf{1}, \tag{22}$$

$$\gamma_t,\ \phi(q_{t+1}) \geq 0, \tag{23}$$

where, for concreteness, we present a first-order semi-implicit Euler scheme from [21]. The problem data include previous configurations, time step, friction forces, and control inputs.

This feasibility problem (21-23) is closely related to prior work on time-stepping methods [33]. However, in our formulation, the complementarity constraints (22), using an element-wise product, are relaxed by the central-path parameter which successively decreases, $\rho \to 0$, in order to achieve hard contact. Additionally, we implicitly encode the velocity finite-difference constraint in the manipulator dynamics.

B. Friction

Friction directly opposes a contact point's tangential velocity and its magnitude is maximized for a given friction coefficient and normal force. These characteristics are formulated into the following optimization problem,

$$\begin{array}{ll} \underset{b}{\text{minimize}} & v^T b \\ \text{subject to} & \|b\|_2, \leq \mu\gamma. \end{array} \tag{24}$$

While (24) is a second-order cone program [20] and can be solved efficiently using tools from convex optimization, the friction-cone constraint is not differentiable when $b = 0$. Physically, this situation corresponds to contact without tangential force; or tangential motion of the contact point while not in contact. Similar to prior work, and for numerical reasons, we employ a linearized friction cone that double parameterizes the friction-force

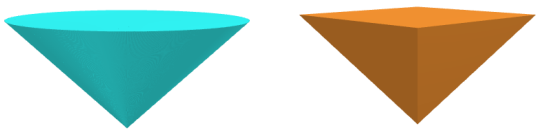

Fig. 3: Friction-cone comparison. Nonlinear, second-order cone (blue) and linearized cone for double parameterized friction force (orange).

vector. The friction-cone reformulation is visualized in Fig. 3. The reformulated optimization problem for the linearized friction cone is,

$$\begin{aligned} \underset{\beta}{\text{minimize}} \quad & \begin{bmatrix} v^T & -v^T \end{bmatrix} \beta \\ \text{subject to} \quad & \mathbf{1}^T \beta \leq \mu\gamma, \\ & \beta \geq 0, \end{aligned} \tag{25}$$

and its optimality conditions in interior-point form are:

$$\eta - \begin{bmatrix} v^T & -v^T \end{bmatrix}^T - \psi\mathbf{1} = 0 \tag{26}$$

$$\psi \cdot (\mu\gamma - \mathbf{1}^T\beta) = \rho \tag{27}$$

$$\beta \circ \eta = \rho\mathbf{1} \tag{28}$$

$$\beta, \eta \geq 0. \tag{29}$$

Problem data include tangential velocity and impact impulse magnitude. Again, this feasibility problem (26-29) is closely related to prior work [33] and our formulation has relaxed complementarity constraints that are successively decreased to achieve hard contact. The friction problem is coupled to the impact problem via the impact magnitude and tangential velocity, which is configuration dependent.

C. Contact Jacobian

For simulations with point contacts and flat surfaces the contact Jacobian associated with ith contact point, $J^{(i)}$, is simply the kinematic Jacobian mapping changes in the system's configuration q into changes in the contact-point position $p^{(i)}$ expressed in the world frame.

For more complex environments or friction parameterizations, the contact Jacobian is computed as,

$$J^{(i)}(q) = D^{[d]} \ {}^{\mathrm{S}}R^{\mathrm{W}}(p^{(i)}(q)) \ \frac{dp^{(i)}(q)}{dq}, \tag{30}$$

with rotation matrix, ${}^{\mathrm{S}}R^{\mathrm{W}} : \mathbf{R}^d \rightarrow \mathbf{SO}(d)$, that maps vectors in the world frame into the surface frame, and projection matrix D that maps contact forces into a parameterized space. For the double-parameterized friction force we use,

$$D^{[2]} = \begin{bmatrix} 1 & 0 \\ -1 & 0 \\ 0 & 1 \end{bmatrix}, \quad D^{[3]} = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ -1 & 0 & 0 \\ 0 & -1 & 0 \\ 0 & 0 & 1 \end{bmatrix}, \tag{31}$$

where the superscript denotes the 2D and 3D cases, respectively. Without double parameterization, D is simply the identity matrix.

We employ Rodrigues' rotation formula [31] to compute the rotation matrix,

$$\begin{aligned} {}^{\mathrm{S}}R^{\mathrm{W}} &= I + [\hat{n}^{\mathrm{S}} \times \hat{n}^{\mathrm{W}}]_\times \\ &\quad + \frac{1}{1 + (\hat{n}^{\mathrm{S}})^T \hat{n}^{\mathrm{W}}} [\hat{n}^{\mathrm{S}} \times \hat{n}^{\mathrm{W}}]_\times [\hat{n}^{\mathrm{S}} \times \hat{n}^{\mathrm{W}}]_\times, \end{aligned} \tag{32}$$

relating the world unit z-axis, $\hat{n}^{\mathrm{W}}$, and the normalized surface normal, $\hat{n}^{\mathrm{S}}$. We use the operator $[\cdot]_\times$ to represent a transformation from a vector into a skew-symmetric matrix.

In 3D for example, $n^{\mathrm{W}} = (0, 0, 1)$, and $\hat{n}^{\mathrm{S}} = (F_x, F_y, F_z)/\sqrt{F_x^2 + F_y^2 + F_z^2}$, where F_j is a partial derivative of an implicit surface, $F : \mathbf{R}^3 \rightarrow \mathbf{R}$. For flat, horizontal surfaces, this rotation matrix is simply the identity matrix.

D. Dynamics

To evaluate our contact dynamics, we need to jointly solve the coupled impact and friction problems. We formulate a joint feasibility problem[1] that simultaneously satisfies (21-23) and (26-29),

[1]Solving this feasibility problem for multiple contact points is equivalent to finding a Nash equilibrium [25] in a multi-player game where one player aims at solving Problem 20 and the others players solve Problem 25 for each contact point.

$$
\begin{aligned}
\text{find} \quad & q_{t+1}, \lambda_t, \psi_t, \eta_t^{(1)}, \ldots, \eta_t^{(c)}, s_\phi, s_\psi && (33)\\
\text{s.t.} \quad & \Big(M(q_{t-1})(q_t - q_{t-1}) && (34)\\
& \quad - M(q_t)(q_{t+1} - q_t) \Big) / h \\
& \quad - hC(q_t, (q_{t+1} - q_t)/h) \\
& \quad + J(q_{t+1})^T \lambda_t + B(q_{t+1}) u_t = 0, \\
& s_\phi - \phi(q_{t+1}) = 0, && (35)\\
& s_\psi^{(i)} - (\mu^{(i)} \gamma_t^{(i)} - \mathbf{1}^T \beta_t^{(i)}) = 0, \forall i, && (36)\\
& \eta_t^{(i)} - P^{(i)}(q_{t+1})(q_{t+1} - q_t)/h && (37)\\
& \quad - \psi_t^{(i)} \mathbf{1} = 0, \forall i, \\
& \gamma_t \circ s_\phi = \rho \mathbf{1}, && (38)\\
& \psi_t \circ s_\psi = \rho \mathbf{1}, && (39)\\
& \beta_t^{(i)} \circ \eta_t^{(i)} = \rho \mathbf{1}, \forall i, && (40)\\
& \gamma_t, s_\phi, \psi_t, s_\psi \geq 0, && (41)\\
& \beta_t^{(i)}, \eta_t^{(i)} \geq 0, \forall i. && (42)
\end{aligned}
$$

We introduce additional slack variables s_ϕ, $s_\psi \in \mathbf{R}^c$, a standard technique used with interior-point methods to simplify the log-barrier term, and replace the tangential velocities with finite-difference approximations of the configuration-space velocities projected through,

$$
P^{(i)}(q) = \begin{bmatrix} \bar{J}^{(i)}(q) \\ -\bar{J}^{(i)}(q) \end{bmatrix}, \tag{43}
$$

into the surface's tangent space, where we use an overbar to denote the tangential components of the contact Jacobian.

To solve (33) with our interior-point framework we define $w_t = (q_{t+1}, \lambda_t, \psi_t, \eta_t^{(1)}, \ldots, \eta_t^{(c)}, s_\phi, s_\psi)$ and set the residual to be comprised of the equality constraints. The interior-point method presented in Section II-D is used to solve (33) via a sequence subproblems that corresponds physically to converging from soft to hard contact. We note that by construction, the Jacobian associated with (33) is always full rank and that regularization is never required. The interior-point method is particularly well suited to handle the increasingly ill-conditioned properties of this Jacobian as hard contact is achieved.

E. Differentiable Dynamics

We define the problem data for (33) as $\theta = (q_{t-1}, q_t, u_t)$. 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 II-E. The problem data could also include the time step, friction coefficients, central path parameter, and other system parameters like masses or inertia terms. Additionally, the central-path parameter ρ can be set to control the gradient's characteristics. Gradients computed with large values of ρ will be smoother than those computed with small values of ρ, which more closely approximates the true non-smooth behavior of contact.

IV. Linear Contact-Implicit Model-Predictive Control

We formulate a model-predictive-control algorithm that utilizes our contact-dynamics formulation (33) for tracking a reference trajectory. First, we present a version of the contact-dynamics feasibility problem that has a subset of terms linearized along a reference trajectory while retaining contact modes via nonlinear complementarity constraints. Then, we employ these dynamics in our LCI-MPC algorithm. Finally, we present a number of key implementation details that enable the policy to be fast and robust.

A. Linear Contact-Implicit Dynamics

The LCI-MPC policy aims to track a reference trajectory comprised of 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 linear contact-implicit dynamics which comprise of the manipulator dynamics, signed-distance function, and maximum dissipation principle terms linearized about the reference trajectory, while key contact dynamics modeled with nonlinear complementarity and inequality constraints are retained. At each time step along the trajectory, we formulate the following feasibility problem,

$$
\begin{aligned}
\text{find} \quad & w \\
\text{subject to} \quad & C(w - \bar{w}) + D(\theta - \bar{\theta}) = 0 \\
& \gamma \circ s_\phi = \rho \mathbf{1}, \\
& \psi \circ s_\psi = \rho \mathbf{1} \\
& \beta^{(i)} \circ \eta^{(i)} = \rho \mathbf{1}, \forall i \\
& \gamma, s_\phi, \psi, s_\psi \geq 0 \\
& \beta^{(i)}, \eta^{(i)} \geq 0, \forall i.
\end{aligned} \tag{44}
$$

Here, $\bar{w}$ and $\bar{\theta}$ are reference decision variables and problem data, respectively, for the interior-point method. C and D are matrices that define an underdetermined linear system of equations and are pre-computed offline. The linear contact-implicit dynamics,

$$
q_{t+1} = s_t(q_{t-1}, q_t, u_t), \tag{45}
$$

$s_t : \mathbf{R}^n \times \mathbf{R}^n \times \mathbf{R}^m \to \mathbf{R}^n$, solve (44) and return the configuration at the next time step. The contact forces at the current time step can also be returned.

B. Linear Contact-Implicit Dynamics Solver

The most expensive step in evaluating the linear contact-implicit dynamics and computing their gradients is computing the solution to a linear system,

$$R_w \Delta w = r, \tag{46}$$

required by the interior-point method in order to compute a new search direction, where we define $R_w = \frac{\partial r}{\partial w}$.

To reduce the computational cost of this solve, we exploit both the sparsity pattern and the fact that most of R_w remains constant across solution iterations [40] and, therefore, can be pre-computed offline.

For clarity, we work with: $w_1 = q_{t+1}$, $w_2 = (\gamma_t, \psi_t, \beta_t^{(1)}, \ldots, \beta_t^{(c)})$, and $w_3 = (s_\phi, s_\psi, \eta_t^{(1)}, \ldots, \eta_t^{(c)})$, where $w = (w_1, w_2, w_3)$, and similarly, split the residual: $r = (r_1, r_2, r_3)$. The Jacobian sparsity pattern is:

$$R_w = \begin{bmatrix} E & F & 0 \\ G & H & I \\ 0 & \textbf{diag}(w_3) & \textbf{diag}(w_2) \end{bmatrix}, \tag{47}$$

where I denotes the identity matrix.

By exploiting sparsity in the third row of (47), we can form the following condensed system:

$$\begin{bmatrix} E & F \\ G & \bar{H} \end{bmatrix} \begin{bmatrix} \Delta w_1 \\ \Delta w_2 \end{bmatrix} = \begin{bmatrix} r_1 \\ \bar{r}_2 \end{bmatrix} \Leftrightarrow \bar{R}_w \, \Delta \bar{w} = \bar{r}, \tag{48}$$

where,

$$\bar{H} = H - \textbf{diag}(w_2^{-1} \circ w_3), \tag{49}$$

$$\bar{r}_2 = r_2 - w_2^{-1} \circ r_3, \tag{50}$$

$$\Delta w_3 = w_2^{-1} \circ (r_3 - w_3 \circ \Delta w_2), \tag{51}$$

and w_2^{-1} denotes the element-wise reciprocal of vector w_2. This term is always well-defined because the interior-point line search enforces $w_2 > 0$ at each iteration.

To solve for $\Delta \bar{w}$, we leverage the fact that, apart from the bottom-right block, R_w can be computed offline. We perform a QR decomposition on the Schur complement of (48),

$$Q, R \leftarrow \textbf{qr}(\bar{H} - GE^{-1}F), \tag{52}$$

and then solve for the search directions,

$$\Delta w_2 = -R^{-1} Q^T (GE^{-1} r_1 - \bar{r}_2), \tag{53}$$

$$\Delta w_1 = E^{-1}(r_1 - F \Delta w_2). \tag{54}$$

Additionally, E^{-1}, GE^{-1}, and $GE^{-1}F$ are precomputed offline. Finally, after solving for $\Delta \bar{w}$, we obtain Δw_3 with cheap vector-vector operations (51).

For a system with configuration dimension n and c contact points, the computational complexity of solving (46) with a naive approach is $O(n + c)^3$. Our structure-exploiting approach is $O(c^3)$ during the online phase. In practice, this provides a factor of 15 speed-up for evaluating the linear 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 MPC problem.

C. Trajectory Optimization

Similar to linear MPC, we solve a version of (9) with the linear contact-implicit dynamics (44). Further, we lift the problem in order to exploit Markov structure and employ a custom linear solver to reduce the overall solution complexity.

First, we represent a state, $x_t = (q_{t-1}^{(t)}, q_t^{(t)})$, using two configurations. Lifting the problem increases its overall size, but exposes Markov structure to the problem and enables the system's sparsity to be more effectively exploited. 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,

$$d_t(x_t, u_t, x_{t+1}) = \begin{bmatrix} q_t^{(t+1)} \\ q_{t+1}^{(t+1)} \end{bmatrix} - \begin{bmatrix} q_t^{(t)} \\ s_t(q_{t-1}^{(t)}, q_t^{(t)}, u_t) \end{bmatrix} = 0, \tag{55}$$

where $d = (d_1, \cdots, d_{T-1}) \in \mathbf{R}^{2n(T-1)}$, effectively propagate configurations across one time step and encode the linear contact-implicit dynamics. The dynamics Jacobian,

$$C = \nabla d = \begin{bmatrix} -B_1 & I & 0 & 0 \\ 0 & -A_2 & -B_2 & I \\ 0 & 0 & 0 & \ddots \end{bmatrix}, \tag{56}$$

has associated one-step dynamics Jacobians,

$$A_t = \begin{bmatrix} 0 & I \\ \frac{\partial s_t}{\partial q_{t-1}} & \frac{\partial s_t}{\partial q_t} \end{bmatrix}, \, B_t = \begin{bmatrix} 0 \\ \frac{\partial s_t}{\partial u_t} \end{bmatrix}, \tag{57}$$

that have additional sparsity that can be exploited.

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:

$$
\begin{aligned}
&g_t(q_{t-1}, q_t, u_t) = \qquad (58)\\
&\quad (q_t - \bar{q}_t)^T Q_t (q_t - \bar{q}_t)\\
&\quad + (u_t - \bar{u}_t)^T R_t (u_t - \bar{u}_t)\\
&\quad + \frac{1}{h^2}(q_t - q_{t-1})^T V_t (q_t - q_{t-1}),\\
&\qquad t = 1, \cdots, T-1, \qquad (59)\\
&g_T(q_{T-1}, q_T) = \qquad (60)\\
&\quad (q_T - \bar{q}_T)^T Q_T (q_T - \bar{q}_T)\\
&\quad + \frac{1}{h^2}(q_T - q_{T-1})^T V_T (q_T - q_{T-1}),
\end{aligned}
$$

where $Q_t, V_t \in \mathbf{S}^n_{++}$ and $R_t \in \mathbf{S}^m_{++}$ are chosen to be diagonal matrices to further reduce complexity. Velocities are penalized using finite-difference approximations and because the problem is lifted, these costs do not introduce state coupling across time steps. The resulting Hessian of the objective,

$$
\nabla^2 J = \begin{bmatrix} R_1 & 0 & 0 & 0 \\ 0 & P_2 & 0 & 0 \\ 0 & 0 & R_2 & 0 \\ 0 & 0 & 0 & \ddots \end{bmatrix}, \qquad (61)
$$

and its inverse,

$$
(\nabla^2 J)^{-1} = \begin{bmatrix} R_1^{-1} & 0 & 0 & 0 \\ 0 & P_2^{-1} & 0 & 0 \\ 0 & 0 & R_2^{-1} & 0 \\ 0 & 0 & 0 & \ddots \end{bmatrix}, \qquad (62)
$$

are block diagonal and can be pre-computed offline. The matrix $P_t \in \mathbf{S}^{2n}_{++}$ is comprised of Q_t and V_t.

The resulting system,

$$
\begin{bmatrix} \nabla^2 J & C^T \\ C & 0 \end{bmatrix} \begin{bmatrix} \Delta z \\ \Delta \nu \end{bmatrix} = \begin{bmatrix} \nabla J + C^T \nu \\ d \end{bmatrix}, \qquad (63)
$$

which uses a Gauss-Newton approximation of the constraints when computing the Hessian of the Lagrangian, is solved using a block elimination approach [36]. First, the Schur complement,

$$
Y = C(\nabla^2 J)^{-1} C^T = \begin{bmatrix} Y_{11} & Y_{12} & 0 & 0 \\ Y_{21} & Y_{22} & Y_{23} & 0 \\ 0 & Y_{32} & Y_{33} & \dots \\ 0 & 0 & \vdots & \ddots \end{bmatrix}, \qquad (64)
$$

is formed blockwise,

$$
\begin{aligned}
Y_{11} &= B_1 R_1^{-1} B_1^T, \qquad (65)\\
Y_{tt} &= B_t R_t^{-1} B_t^T + A_t P_t^{-1} A_t^T + P_{t+1}^{-1}, \qquad (66)\\
&\qquad t = 2, \cdots, T-1,\\
Y_{t,t+1} &= Y_{t+1,t}{}^T = -P_{t+1}^{-1} A_{t+1}^T, \qquad (67)\\
&\qquad t = 1, \cdots, T-2.
\end{aligned}
$$

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

$$
\begin{aligned}
L_{11} L_{11}^T &= Y_{11} \qquad (68)\\
L_{tt} L_{t+1,t}^T &= Y_{t,t+1}, \qquad (69)\\
&\qquad t = 1, \cdots, T-1,\\
L_{tt} L_{tt}^T &= Y_{tt} - L_{t,t-1} L_{t,t-1}^T, \qquad (70)\\
&\qquad t = 2, \cdots, T-1.
\end{aligned}
$$

Finally, block elimination is utilized to solve,

$$
\Delta \nu = Y^{-1}\Big(C(\nabla^2 J)^{-1}(\nabla J + C^T \nu) - d\Big), \qquad (71)
$$

$$
\Delta z = (\nabla^2 J)^{-1}\Big(\nabla J + C^T \nu - C^T \Delta \nu\Big). \qquad (72)
$$

Without exploiting the trajectory-optimization problem structure, the complexity of solving this instantiation of (9) with a generic LDL^T factorization is $O\Big(\frac{1}{3}(T-1)^3(4n+m)^3\Big)$, while the custom solver has a reduced overall complexity: $O\Big((T-1)(8n^3 + n^2 m)\Big)$.

D. Model-Predictive Control

The LCI-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 (44) 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 (9) 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 interior-point methods for MPC [36], 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 simulating a system using the nonlinear contact dynamics, we use a tight central-path parameter tolerance, e.g., $\rho =$ 1e-6, to enforce hard contact, see Section VI for more details. The LCI-MPC policy is summarized in Algorithm 2.

Algorithm 2 Linear 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

E. Contact-Height Heuristic

The policy is able to adapt online to unknown variations in terrain by maintaining a height estimate, $a \in \mathbf{R}^c$, for each contact. The policy utilizes a modified signed-distance function,

$$\phi_{\text{MPC}}(q) = \phi(q) + a, \tag{73}$$

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 (47) and only requires c more addition operations to compute r when solving the linear contact-implicit feasibility problem (46). In our experiments, we find the heuristic to be effective and reliable across unknown terrain for all of the systems tested.

V. EXAMPLES

We demonstrate the LCI-MPC algorithm 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 sequence online; is robust to disturbances, model mismatch and unknown terrain; and is faster than real-time (Table I).

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 [21] 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 tracked, while in the latter case, two templates are generated and combined to form a complete reference trajectory. It is also possible to hand design trajectories, as was done for the hopper in 3D.

We verify the policy performance in simulation using the full nonlinear contact dynamics (33). Additionally, all examples are simulated using a different sample rate, typically 5-10$\times$ faster than the reference trajectory, in order to ensure that the policy is robust to sampling rates.

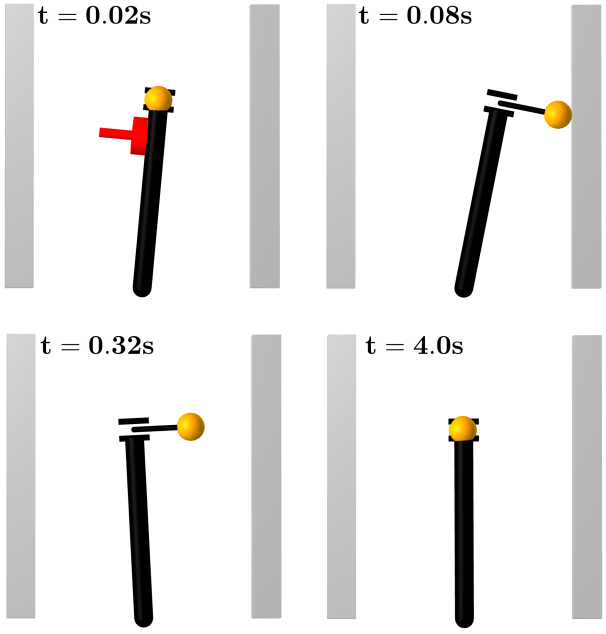

Fig. 4: PushBot performing push recovery. A disturbance (red) creates an impulse on the system and the policy generates a new contact sequence that extends the prismatic joint toward the right wall in order to make contact. After stabilizing, PushBot pushes against the wall, eventually breaking contact, in order to return to the nominal upright configuration.

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. 4). 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

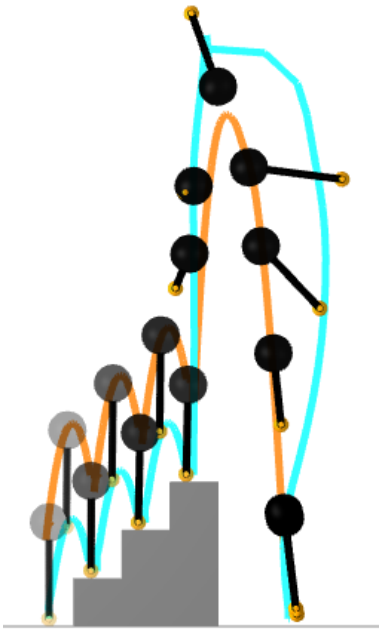

Fig. 5: Hopper in 2D performing parkour. The system tracks the body (orange) and foot (blue) reference trajectories while ascending three stairs before performing a front flip.

to return to the nominal position. The latter behavior is shown in Fig. 4.

An alternative approach for solving similar push-recovery problems using explicit MPC and mixed-integer convex optimization takes seconds, and must be solved offline [22]. Our LCI-MPC approach is fast enough to be run online.

B. Hopper

Inspired by the Raibert Hopper [30], we model a 2D hopping robot with $n = 4$ configuration variables: 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.

In this scenario, the hopper is tasked with moving to the right over unknown terrain. The reference trajectory is optimized with a flat surface and no incline. We

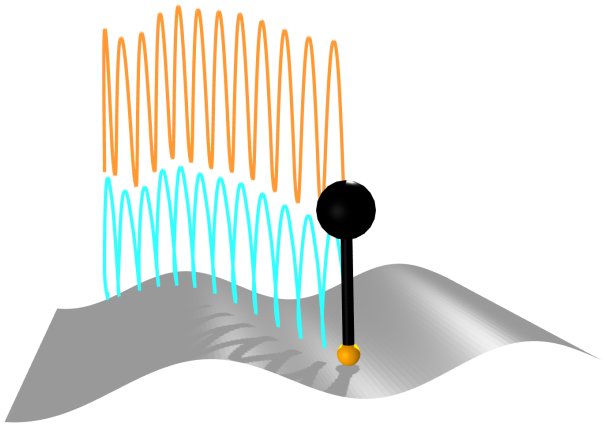

Fig. 6: Hopper in 3D tracks a hopping gait with body (orange) and foot (blue) trajectories over unmodeled and uneven terrain.

compare our policy to the Raibert heuristic, which we tune for flat ground and no incline. Online, our policy is able to adapt to the varying surface height that ranges from 0-24 cm. The robot slips multiple times and is able to recover while traversing steep inclines. We find that, when tuned well, the Raibert heuristic also works very well in this setting.

Additionally, we task the hopper with climbing a staircase and executing a front flip (Fig. 5). 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 LCI-MPC.

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. 5).

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

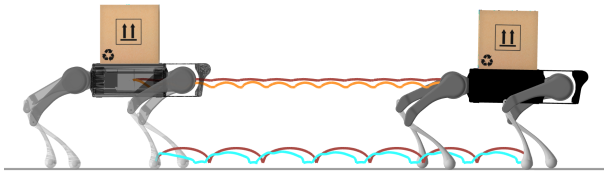

Fig. 7: Quadruped tracking a reference trajectory (red) while carrying an unmodeled 3-kg payload. We depict the torso (orange) and front-left foot (blue) trajectories.

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 executes 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 [29] 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, shown in Fig. 8, using the same MPC policy. We compare this to Pratt's policy [29], 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.

E. Monte Carlo Initial Conditions

In order to assess the robustness of LCI-MPC, we perform Monte Carlo analysis on two systems: the hopper (2D) and the quadruped. We randomly sample the initial configurations of the robots which are tasked with tracking a reference gait (Fig. 9). For each system we use 100 samples; we show that the hopper recovers from significant initial tilts and that the quadruped is robust to large drops.

VI. Discussion

LCI-MPC is capable of robustly tracking reference trajectories through contact despite disturbances, model

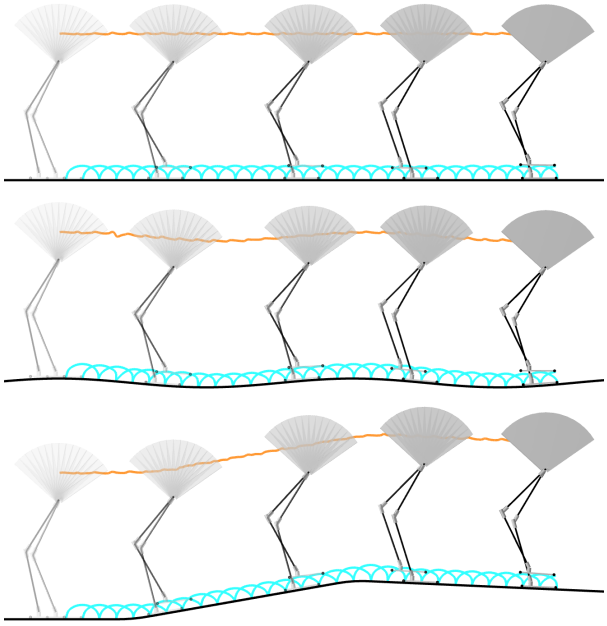

Fig. 8: Biped walking from left to right across flat (top), sinusoidal (middle), and piecewise linear (bottom) terrain using the same policy.

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 linear 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 linear contact-implicit dynamics work well for the full 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 interior-point approach, 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 gradi-

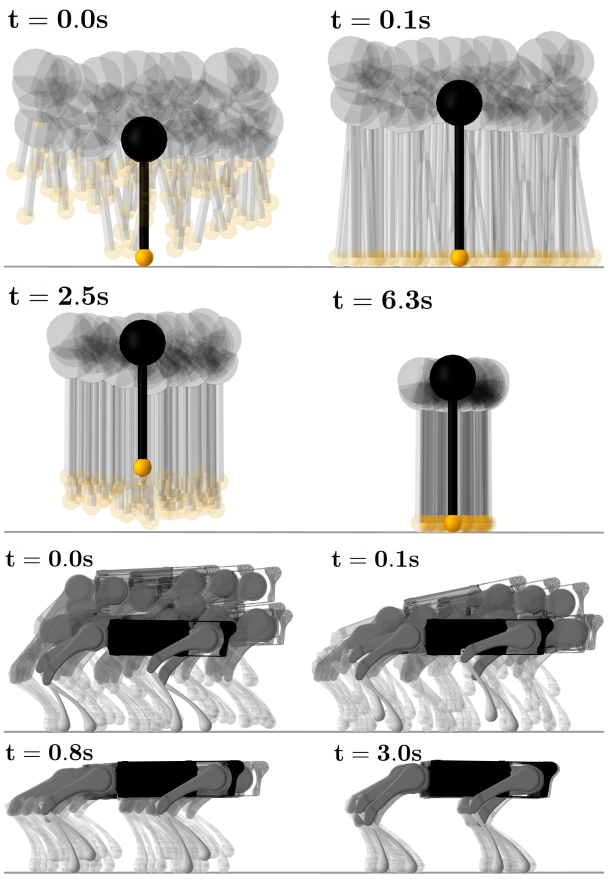

Fig. 9: Monte Carlo simulations of initial conditions for systems tracking a reference trajectory. For a hopper (top) and quadruped (bottom), 100 different initial configurations are tested. The initial disturbances include large translations, tilts, and joint angles offsets compared to the reference initial configuration. For all the samples, and both systems, the policy successfully recovers to the reference gait.

ents. 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.

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 LCI-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, LCI-MPC is able to track reference trajectories for a variety of systems with nonsmooth dynamics that make and break contact with their environments.

TABLE I: The MPC policy runs at a fixed rate corresponding to the reference trajectory's rate f_{ref}. For all robotic systems, the MPC 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 laptop 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

Contact-implicit trajectory optimization [28, 21] 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, *nonlinear* contact-implicit MPC is generally impractical. In this work, we demonstrate that *linear* contact-implicit MPC can be extremely reliable, converging despite disturbance and initial conditions with large offests. In practice, we find that the one-step contact-dynamics problem always converges, largely enabled by a numerically robust interior-point 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.

VII. 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 LCI-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 LCI-MPC would be against a nonlinear 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 solved 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] Akshay Agrawal, Brandon Amos, Shane Barratt, Stephen Boyd, Steven Diamond, and J. Zico Kolter. Differentiable convex optimization layers. In *Advances in Neural Information Processing Systems*, pages 9562–9574, 2019.

[2] 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.

[3] 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.

[4] 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.

[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] 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.

[8] 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.

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

[10] 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.

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

[12] 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.

[13] 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.

[14] 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.

[15] 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.

[16] 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.

[17] Lev Davidovich Landau and Evgenii Mikhailovich Lifshitz. *Mechanics*, volume 1. Butterworth-Heinemann, 1976.

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

[19] 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.

[20] Miguel Sousa Lobo, Lieven Vandenberghe, Stephen Boyd, and Hervé Lebret. Applications of second-order cone programming. *Linear Algebra and its Applications*, 284(1-3):193–228, 1998.

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

[22] Tobia Marcucci, Robin Deits, Marco Gabiccini, Antonio Bicchi, and Russ Tedrake. Approximate hybrid model predictive control for multi-contact push recovery in complex environments. In *2017 IEEE-RAS 17th International Conference on Humanoid Robotics (Humanoids)*, pages 31–38. IEEE, 2017.

[23] Jerrold E. Marsden and Matthew West. Discrete mechanics and variational integrators. *Acta Numerica*, 10:357–514, 2001.

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

[25] John F. Nash. Equilibrium Points in n-Person Games. *Proceedings of the National Academy of Sciences of the United States of America*, 36(1):48–49, 1950. URL http://www.jstor.org/stable/88031.

[26] 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.

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

[28] 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.

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

[30] 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.

[31] Olinde Rodrigues. Des lois géométriques qui régissent les déplacements d'un système solide dans l'espace, et de la variation des coordonnées provenant de ces déplacements considérés indépendamment des causes qui peuvent les produire. volume 5(1) of *Journal de Mathématiques Pures et Appliquées*, pages 380–440. 1840.

[32] 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.

[33] 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.

[34] 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.

[35] 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.

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

[37] 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.

[38] 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.

[39] 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.

[40] 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.