Contact-Implicit Differential Dynamic Programming for a Hopping Robot

ROB599-014 Legged Robot Control - Term Project

Kevin Best

***Abstract*—Online contact planning is a desirable feature in a legged-robot control system, as it allows the controller more freedom in planning motions to navigate various terrains. However, the numerical consequences of including decisions regarding contact in optimal control problems often make them difficult to solve and prohibitively slow for real-time use. In this work, I implement the contact-implicit differential dynamic programming-based model predictive controller first presented in [1], which, through clever relaxations of the ground contact complementarity constraints, can select appropriate contact modes at rates suitable for online use. After reviewing the mathematical theory behind the controller, I demonstrate a baseline Differential Dynamic Programming controller for contact-free systems with actuator disturbances in simulation. Then, I demonstrate the controller's ability to control hopping robot with 2 actuators and 4 degrees of freedom, performing a series of increasingly complex motions again in simulation. My results verify the performance reported in [1], and I propose various avenues for future investigation regarding this promising control methodology.**

I. Introduction

Aside from the significant perception and state estimation challenges, real-time contact planning is one of the most difficult barriers preventing legged robots from achieving robust and agile real-world ambulation. In order to effectively navigate unstructured, realistic environments, robots should be able to select where and when to make contact with their environment in real-time. As the robot's contact conditions (*i.e.* which feet/hands are in contact with the environment) determine its equations of motion, the selection of contacts has a strong impact on the possible behaviors that a controller can produce.

Robots with varying contact conditions are often modeled as hybrid systems [2]. In a hybrid system model, impact maps are used to transition the dynamics from one continuous manifold to another when the state enters a switching set defined by the contact surfaces. Between contact events, the system state evolves smoothly until it hits another switching surface.

These hybrid dynamics associated with varying contact conditions can be challenging for traditional trajectory optimization methods to handle due to the lack of differentiability during impact events [3]. Thus, many optimal-control based paradigms use a pre-defined *contact schedule*, in which the sequence of the various contact modes are defined *a priori*. The assumption of a known contact sequence allows the optimal control problem (OCP) to assume when transitions occur and thus to only optimize over the smooth manifolds of the dynamics. However, when disturbances or other environmental factors render the pre-defined contact sequence invalid, the controller may be unable to stabilize the system or may not leverage available contacts to their fullest extent. Therefore, methods that can select contact conditions as needed are desirable.

One method for automatically selecting the best contact condition within the OCP, termed *contact-implicit* optimization, is to enumerate all possible contact modes and treat them as discrete decision variables [4]. However, the resulting OCP becomes a mixed-integer program, which can be notoriously difficult to solve. These methods are thus best suited to systems with small numbers of contact modes or offline planning. However, some formulations are able to structure the problem as a mixed-integer quadratic program, and can solve it fast enough for real-time use [4].

An alternative to a hybrid system model is a complementarity model [5]. Complementarity models implicitly define the hybrid dynamics by adding constraints between the normal contact forces λ_n and the signed distance between the contact point and the surface $\phi(q)$:

$$0 \leq \lambda_n \perp \phi(q) \geq 0. \tag{1}$$

The process of solving for the generalized acceleration $\ddot{q}$ is thus given by a *linear complementarity problem*:

$$\begin{aligned} &\text{find } \ddot{q}, \lambda \\ &\text{s.t. } \ddot{q} = f(q, \dot{q}, u, \lambda) \\ &\quad\ \ \phi(q) \geq 0 \\ &\quad\ \ \lambda_n \geq 0 \\ &\quad\ \ \phi(q)^T \lambda_n = 0 \end{aligned} \tag{2}$$

where the system dynamics are given by $f(q, \dot{q}, u, \lambda)$ and λ is defined via normal and tangential forces as $\lambda = [\lambda_n, \lambda_t]^T$.

While this formulation of the dynamics was originally developed for forward simulation, it can also be used to describe the system dynamics in an OCP. The benefit over hybrid formulations is that a single set of constrained dynamics are used instead of a collection of unconstrained dynamics with jumps between them. Thus, standard direct optimal control approaches can be applied, such as direct collocation in [6].

However, the addition of complementarity constraints to a direct collocation OCP results in a poorly-conditioned nonlinear program, as the orthogonality constraint (1) is numerically stiff. This stiffness makes the OCP often too slow to solve for online use. For example, Posa et al. reported solution times on the order of minutes and hours for small horizon plans [6].

A relaxation to the complementarity constraint was proposed by Manchester et al. in order to improve the convergence properties of the OCP [7]. A slack variable s was introduced to allow violation of the complementarity condition during early iterations of the OCP:

$$s - \phi(q)^T \lambda_n = 0. \tag{3}$$

As the solution converged, the cost weight on $||s||^2$ was iteratively increased, converging back to the original complementarity condition in the limit.

Shooting methods are an alternative to collocation methods, with the benefit of not requiring the system dynamics as explicit constraints in the OCP. One particular shooting method that has gained popularity in recent years is Differential Dynamic Programming (DDP). DDP is a local value iteration method in which the action-value function Q is approximated at each time step as a quadratic function. The optimal control sequence is iteratively calculated by minimizing Q [8], [9]. This minimization has a closed form solution in the case of an unconstrained problem, or a small quadratic program in the case of box constraints [10]. An iterative process alternating between forward simulation and backwards control updates is repeated until convergence.

In order to calculate the Q function, one must calculate the gradients and Hessians of the dynamics and cost functions. As hybrid systems have undefined gradients at impacts, they are not compatible with DDP when considering a contact-implicit approach. Complementarity methods, however, do always have well-defined gradients and are thus can be used in a contact-implicit DDP [11]. However, the finite-differencing method used in [11] to calculate the contact-related gradients was not fast enough for real-time use.

Kim et. al [1], [12] avoid the speed penalties of finite-differencing by utilizing the methods in [13] to analytically calculate the contact impulse gradient, and thus execute the DDP controller at sufficient rates for online-use. Another innovation in [1] is a renaissance of the complementarity relaxation idea [7] applied to this gradient calculation. By relaxing the contact gradient, the authors show that the optimization can more easily discover new contact modes that further reduce the cost function. The use of non-relaxed hard contact in the forward rollout helps ensure that the optimizer arrives at physically realistic solutions.

In this work, I implemented the DDP-based Model Predictive Controller (MPC) presented in [1], [12] in a MATLAB simulation environment. This controller utilizes a contact-implicit formulation that allows for dynamic movements including contact based on simple floating-base reference trajectories. I applied the controller to a hopping robot system, inspired by the robot shown in Fig. 1. I first show that the controller can generate hopping behaviors when given a static desired pose above the ground. I then demonstrate a single sideways hop and a continuous sideways hopping gait. Finally, I discuss the limitations of my implementation and various avenues for future improvements.

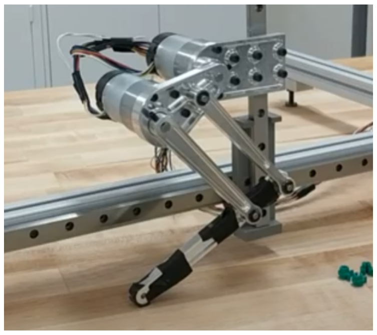
Fig. 1. A photo of the four DoF robot system that was the inspiration of this work. The planar system has two floating base DoF and two configurable joint angles. I built this robot during a semester of undergraduate research with Professor Patrick Wensing at the University of Notre Dame.

II. METHODS

This section details the theory behind the DDP-MPC controller. I first discuss the system dynamics, including the hard-contact calculations and the calculation of the gradients required to use DDP. I then outline the DDP algorithm with details of its specific implementation, and I discuss the simulation used to evaluate the controller. The software accompanying this work can be found in this Github repository.

A. System Dynamics

As an initial study, I used a planar robot with one rotational joint and one prismatic joint for a total of four degrees of freedom (DOF). The floating base was free to move within the plane, but not to rotate, similar to the robot in Fig. 1. This system was selected due to its simplicity (*e.g.*, no kinematic loops), minor nonlinearity, and computational scale. Future studies can build on this work and incorporate systems with more complex dynamics.

1) Rigid Body Dynamics: I implemented the system within the `SpatialV2` framework [14]. This library provides pre-built functions to calculate the system forward dynamics, given by

$$\ddot{q} = f(q, \dot{q}, u, \tilde{\lambda}) = M^{-1}(q)\left(Bu - c(q, \dot{q}) + J^T(q)\tilde{\lambda}\right). \tag{4}$$

The generalized coordinates and their first two time derivatives are given by $q, \dot{q}, \ddot{q}$ respectively. M is the inertia matrix, and c comprises the coriolis, centrifugal, and gravitational terms. The actuation selector matrix B maps control inputs u to the generalized forces. J is the Jacobian that maps forces at the robot foot $\tilde{\lambda}$ in a local contact frame with a normal and tangential direction to the generalized forces. The tilde is used to distinguish forces $\tilde{\lambda}$ from corresponding impulses λ, where $\lambda = \int_0^{\Delta t} \tilde{\lambda} dt$.

DDP is a discrete-time framework and thus it is helpful to also denote the system's discrete time dynamics. Let the state vector x be $[q^T, \dot{q}^T]^T$. Using an explicit Euler approximation with period Δt, we write the discrete dynamics $F(\cdot)$ that map from a current state x_k at timestep k to the state at the next timestep x_{k+1} as

$$\begin{aligned} x_{k+1} &= F(q_k, \dot{q}_k, u_k, \lambda_k), &(5)\\ &= x_k + \begin{bmatrix} \dot{q}_k \\ f(q_k, \dot{q}_k, u_k, \lambda_k) \end{bmatrix} \Delta t. &(6) \end{aligned}$$

For notation simplicity, we will often denote values at the next timestep with an apostrophe (*i.e.,* $x_{k+1} = x'$) and if no index is denoted, we assume it refers to the current timestep (*i.e.,* $x_k = x$).

2) Contact Dynamics: In order to calculate the contact impulse λ, we follow the approach presented in [15]. We limit our analysis to a single possible contact point to simplify the algorithm. However, one could extend to multiple simultaneous contact points using the iteration methods from [15]. If the distance between the ground and the contact point $\phi(q) > 0$, the impulse is trivially zero due to the complementarity condition. Otherwise, the impulse must be determined using the maximum dissipation principle.

In the case of $\phi(q) \leq 0$, we can find λ as the impulse that minimizes the kinetic energy of the contact point. Let $v' = J(q')\dot{q}'$, $\mathcal{M} = \left(J(q)M^{-1}J\right)$, and n and t subscripts denote normal and tangential directions, respectively. The contact impulse is given by the solution to

$$\lambda = \underset{\lambda \in \mathcal{S}}{\arg\min} \quad v'^T \mathcal{M} v' \tag{7}$$

where $\mathcal{S}$ is the feasible set defined by the friction cone[1] and the complementarity condition.

When $\phi(q) \leq 0$, contact can be categorized into three classes: 1) opening contacts, 2) slip contacts, and 3) stick contacts. In opening contacts (*i.e.,* $v_n > 0$), the impulse is trivially zero, as the foot is separating from the contact surface. In the case of slips and sticks, the impulse is given by the solution of (7). In the case of a single stick contact, the optimization problem has a closed form solution of

$$\lambda = -\mathcal{M}h \tag{8}$$

where $h = J\dot{q} + JM^{-1}\Delta t(Bu - c)$. In the event of a slip, we simply clamp the stick solution to the friction cone, as our system is planar. See [15] for more detail on the non-planar case, which is significantly more complicated.

3) Gradients and Hessians of the Dynamics: DDP requires expressions for the gradients and Hessians of the discrete dynamics with respect to the state and control variables. We utilize the derivation presented in [13]. The derivatives of (6) with respect to x and u are

$$\nabla_x F(\cdot) = I + \begin{bmatrix} 0 & I \\ \frac{\partial f}{\partial q} & \frac{\partial f}{\partial \dot{q}} \end{bmatrix} \Delta t, \tag{9}$$

$$\nabla_u F(\cdot) = M^{-1}\left(B + \frac{\partial f}{\partial u}\right)\Delta t. \tag{10}$$

[1]As we are only considering a planar system, the friction cone and the friction pyramid are identical

Thus, we require the derivatives of the continuous dynamics with respect to x and u. Applying the chain rule yields

$$\begin{aligned} \frac{\partial f}{\partial q} &= \frac{\partial M^{-1}}{\partial q}(c - \tau + J^T\tilde{\lambda}) \\ &\quad + M^{-1}\left(\frac{\partial c}{\partial q} + \frac{\partial J^T}{\partial q}\tilde{\lambda} + J^T\frac{\partial \tilde{\lambda}}{\partial q}\right), &(11)\\ \frac{\partial f}{\partial \dot{q}} &= M^{-1}\left(\frac{\partial c}{\partial \dot{q}} + J^T\frac{\partial \tilde{\lambda}}{\partial \dot{q}}\right), &(12)\\ \frac{\partial f}{\partial u} &= J^T\frac{\partial \tilde{\lambda}}{\partial u}. &(13) \end{aligned}$$

Recursive algorithms exist to quickly evaluate the derivatives shown in blue [16], which are already implemented in the extended version of `SpatialV2`.

The terms in red however involve differentiating the contact force $\tilde{\lambda}$. In [13], the authors present a closed form solution to these derivatives based on the exact complementarity condition. However, in [1], [12] the authors suggest a relaxation of this derivative in order to assist the DDP algorithm in finding novel contact conditions. By relaxing the complementarity condition $v_n\lambda_n = 0 \rightarrow v_n\lambda_n = \rho$ for a small ρ, Kim et al. show better convergence properties in the DDP optimization. We therefore use this relaxed gradient in this work.

The velocity of the contact point at the next iteration is given by the linear relationship

$$v' = A\lambda + b, \tag{14}$$

where

$$\begin{aligned} A &= JM^{-1}J^T, &(15)\\ b &= JM^{-1}((-c + Bu)\Delta t + M\dot{q}). &(16) \end{aligned}$$

Note that the expressions for A and B differ slightly for the case of sliding, but I refer the reader to [13] for the sake of space. A is denoted separately from $\mathcal{M}$ for this reason, but are identical in the case of a sticking contact. By relaxing the complementary condition, Kim et al. [1] show that the derivative of λ with respect to an arbitrary vector ξ is

$$\frac{\partial \lambda}{\partial \xi} = -(A + \rho D)^{-1}\left(\frac{\partial A}{\partial \xi}\lambda + \frac{\partial b}{\partial \xi}\right), \tag{17}$$

where D is a diagonal matrix of the form $D = \text{diag}(1/\lambda_n^2, 0)$. In this work, we use $\rho = 0.1$ for the relaxation parameter.

Noting that $\frac{\partial \tilde{\lambda}}{\partial q} = \frac{\partial \lambda}{\partial q}\frac{1}{\Delta t}$, (17) can be used to evaluate each of the red terms in (11) - (13). In my implementation, CasADi was used to evaluate $\frac{\partial A}{\partial \xi}$, etc.

B. DDP Optimal Control

Let us denote the discrete control sequence as $U \equiv [u_0, \ldots, u_{N-1}]$, the discrete state sequence $X \equiv [x_0, \ldots, x_N]$ and the initial state as $x_0 = [q_0^T, \dot{q}_0^T]^T$. We seek to solve the following finite horizon OCP:

$$\begin{aligned} &\underset{U}{\arg\min} && J(x_0, U) = \sum_{k=1}^{N-1} \ell(\tilde{x}_k, u_k) + \frac{1}{2}\tilde{x}_N^T P \tilde{x}_N \\ &\text{s.t.} && u_k \in [\underline{u}, \bar{u}]\forall k \end{aligned}$$

where $\ell(\tilde{x}_k, u_k) = \frac{1}{2}\left(\tilde{x}_k^T P \tilde{x}_k + u_k^T R u_k\right)$ is the stage cost, P and R are diagonal positive definite weighting matrices and $[\underline{u}, \bar{u}]$ is the interval defining the admissible control set. We denote the deviation from a desired trajectory x_r as $\tilde{x} \equiv x - x_r$.

A single iteration[2] of the DDP update consists of three parts: 1) a forward rollout with the current estimate of the optimal control trajectory, 2) a backwards pass in which a step direction is calculated for the control update, and 3) a line search to select the step size. As the forward rollout is trivial given (6), we focus on the backwards pass and the line search below.

1) Backwards Pass: The DDP backwards pass recursively selects optimal changes in the control inputs in order to minimize a quadratic approximation of the cost-to-go. Using Bellman's optimality principle, we define the cost-to-go as

$$V(x,k) = \min_u [\ell(\tilde{x}, u) + V(F(x,u), k+1)]. \tag{18}$$

The action-value function Q_k tells us the change in V for a given perturbation in state and control δx and δu respectively at timestep k:

$$\begin{aligned} Q_k(\delta x, \delta u) \equiv & \ell(x+\delta x, u+\delta u, k) - \ell(x,u,k) \\ & + V(F(x+\delta x, u+\delta u), k+1) \\ & - V(F(x,u), k+1) \end{aligned} \tag{19}$$

Using a second-order Taylor series, we approximate Q_k. Note that we use the notation G_x to denote $\nabla_x G$ and we drop the k subscript on the RHS for notational simplicity:

$$Q_k \approx \frac{1}{2} \begin{bmatrix} 1 \\ \delta x \\ \delta u \end{bmatrix}^T \begin{bmatrix} 0 & Q_x^T & Q_u^T \\ Q_x & Q_{xx} & Q_{ux}^T \\ Q_u & Q_{ux} & Q_{uu} \end{bmatrix} \begin{bmatrix} 1 \\ \delta x \\ \delta u \end{bmatrix}$$

where

$$Q_x = \ell_x + F_x^T V_x \tag{20}$$

$$Q_u = \ell_u + F_u^T V_x \tag{21}$$

$$Q_{xx} = \ell_{xx} + F_x^T V_{xx} F_x \tag{22}$$

$$Q_{ux} = \ell_{ux} + F_u^T V_{xx} F_u \tag{23}$$

$$Q_{uu} = \ell_{uu} + F_u^T V_{xx} F_u \tag{24}$$

Given our simple quadratic definition of $\ell(\tilde{x}, u)$, the derivatives of the stage cost are either affine in $\tilde{x}$ or u, constant matrices, or zero. Note that we neglect the tensor terms that involve the second order derivatives of the dynamics, resulting in a Gauss-Newton approximation of the Hessian [17]. This approximation is common in real-time DDP implementations (often named iLQR), as these second derivatives are computationally expensive to compute and the convergence benefits of a full Newton step are often outweighed by the faster iteration time. Note that we can evaluate the derivatives in parallel to improve performance.

Starting at timestep N, we initialize $V_x = P\tilde{x}$ and $V_{xx} = P$. We then iteratively calculate (20) - (24) for each kth timestep, beginning at $k = N-1$. At each iteration, we also calculate the optimal change in the control to minimize the cost-to-go approximation. Minimizing Q with respect to δu with no constraints results in the closed form expression

$$\delta u^* = \underbrace{-Q_{uu}^{-1} Q_u}_{d} - \underbrace{Q_{uu}^{-1} Q_{ux}}_{-K} \delta x, \tag{25}$$

where d is a feed-forward control update and K is a feedback gain matrix stabilizing the state trajectory. However, this minimization does not respect the control constraints from the original OCP. Therefore, Tassa et al. proposed the following constrained QP to accommodate actuator limits [10]:

$$\begin{aligned} \underset{d, K}{\arg\min} \quad & Q(\delta x, \delta u) \\ \text{s.t.} \quad & \underline{u} \le u + \delta u \le \bar{u} \end{aligned}$$

The rows of K corresponding to the constrained directions are overwritten with zeros, as in [10].

2) Armijo Line Search: In the case of linear dynamics and no actuator limits, the solution given by (25) will converge in a single iteration and the feedback gains K will be the same as the LQR solution. However for nonlinear systems, it is possible that taking this full Netwon step can result in a cost increase. This is because the Q function is only a local approximation and too large of an update can make it diverge. We utilize a backtracking line search with an Armijo acceptance condition to balance between adequate step size and convergence [18]. During the backwards pass, we can calculate the expected cost reduction at each time step [10] as

$$\Delta J_{exp,k} = -\frac{1}{2}(d^T Q_{uu} d). \tag{26}$$

This expected cost comes from substituting the control policy δu back into Q. Let $\Delta J_{exp} = \sum_0^{N-1} \Delta J_{exp,k}$. If J is the cost at the beginning of the iteration and J' is the cost after the control update, we iteratively reduce a line search parameter α until the Armijo condition is met:

$$J' < J - \gamma \alpha \Delta J_{exp}. \tag{27}$$

In my experiments, I set the hyperparameter $\gamma = 0.01$, though the behavior of the controller to this parameter was not overly sensitive. To improve speed, an array of α values was evaluated in parallel and the largest value that satisfied (27) was selected as α^*. Finally, the control policy is updated as

$$u' = u + \alpha^* d + K(x' - x). \tag{28}$$

C. MPC Simulation

To evaluate the controller performance, I set up a simulation environment in which the DDP controller was used in a receding horizon MPC framework, similar to [1], [12]. Let Δt be the update period of the MPC controller. The reference state x_r is calculated by forward integration of the desired coordinate velocity $\dot{q}_d$. The simulation proceeded as follows:

1) Run DDP to convergence with $\dot{q}_d = 0$. This novel step was added with the aim of improving the initial guess for the dynamic references.

[2]The algorithm discussed in this section is implemented in the `run_iteration` method of the `DDP_Controller` class of the software repository.

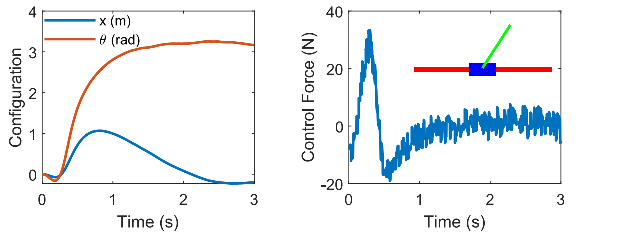

Fig. 2. Configuration and control trajectories of the cart-pole swing up test. Despite the added actuator noise, the MPC controller is able to achieve the desired configuration of a horizontal position $x = 0$ and pole position $\theta = \pi$.

2) Run DDP to convergence with new initial guess and actual $\dot{q}_d$. Store u' from (28).
3) Simulate forward using (6) for Δt seconds using u' as the control policy. Store new state x'. The rate of the simulation is set to 1 kHz to better approximate online use in continuous time.
4) Run DDP for at most two iterations starting at x' and using $u'[2 : \text{end}, \text{end}]$ as the initial guess. Store u' from (28).
5) Return to 3 and repeat.

III. Results

I tested the DDP-MPC controller in various scenarios based on the cases presented in [1]. I first tested a cart-pole system without contact, but with actuator disturbances to validate the DDP-MPC implementation. This step is important, as the value of MPC is being able to re-plan when the actual state differs from the planned state. Then, I considered three desired hopping motions: in place, single hops, and continuous horizontal hopping. These were each realized by commanding various position trajectories of the floating base, with either step or linearly increasing changes.

The main script that implements these results can be found in the Github repository under `./DDP_MPC.m`[3]. The repository README contains GIF videos of various example behaviors, including the ones discussed here.

A. Cart-Pole

To first validate the DDP controller without the contact dynamics, I implemented a cart-pole swing up test, as the cart-pole is the canonical toy system in nonlinear control. The MPC was run at 100 Hz for 100 timesteps (*i.e.,* 1.0 s horizon length). As the dynamics of a cart-pole are quite simple, this longer horizon is achievable without a large slow-down in computation. Uniform random noise ($\tilde{u} \sim U(-5, 5)$ N) was included on the force produced by the actuator to highlight the controller's ability to respond to disturbances. Fig. 2 shows how the controller is able to achieve the desired configuration of a horizontal position $x = 0$ and pole position $\theta = \pi$ despite the actuator noise.

[3]When running this script, one may need to uncomment lines defining the gradients of the dynamics. These can be commented out to avoid recompiling the gradient functions if they are unchanged in order to speed up execution.

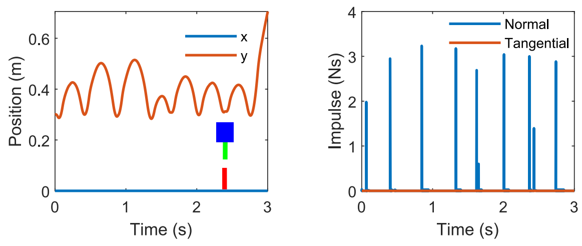

Fig. 3. Position and ground impulse trajectories of the robot performing a vertical hopping. The controller identifies the hopping gait as optimal when simply given a constant reference height 0.6 m above the ground.

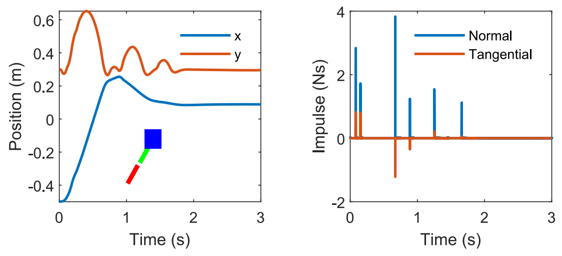

Fig. 4. Position and ground impulse trajectories of the robot performing a sideways hop from an initial x position of -0.5 to 0.0. The robot slightly overshoots the desired position and hops back to the left to recover.

B. RP-Hopper

Next, I tested the controller on the RP hopping robot described in Section II-A. The MPC was run at 200 Hz for 50 timesteps (*i.e.,* 0.25 s horizon length). To parallel the results in [1], I began with a simple hopping-in-place task. I commanded a constant desired floating base height of 0.6 m above the ground, with zeros for the rest of the coordinates. The normal height of the robot is 0.3 m, so the robot must jump to reach this configuration. The controller is able to identify that the best way to minimize the cost is to repeatedly jump from the ground to get close to the desired position, shown in Fig. 3. The robot does not actually achieve the desired height, as the cost function is balancing the height objective with other state and control objectives. However, if height were more important, this penalty weight could be further tuned.

Next, I tested the system's ability to perform a single hop to the right, starting at an x position of -0.5 m with a desired x position of 0.0 m. It is important to note that this reference position was a constant, resulting in a step input to the system. The controller is able to create a series of hops that get the robot close to the desired position. It initially overshoots the desired position and hops back to the left to recover (Fig. 4, note the direction of the tangential impulse). The speed and aggressiveness of this hop was again sensitive to the penalty matrix P, where too large of values would cause significant overshoot and an under-damped convergence to the desired position, similar to what is observed in linear system step responses with insufficient damping.

Finally, I commanded a constant horizontal velocity of the floating base at 0.5 m/s at a nominal base height of 0.3 m. The controller develops a sideways hopping motion, that tracks the

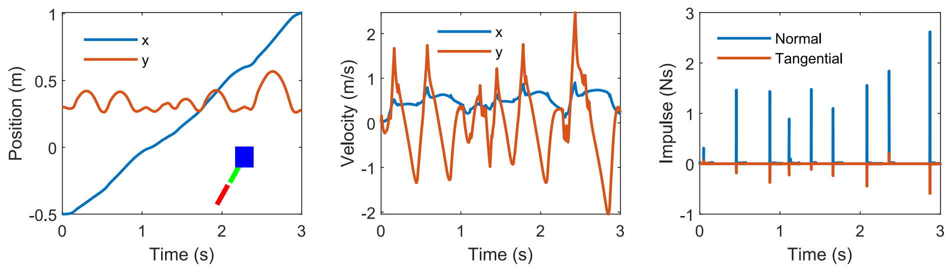

Fig. 5. State, control, and impulse trajectories of the robot tracking a constant horizontal velocity reference of 0.5 m/s. The controller implicitly identifies the hopping gait and is able to maintain the desired velocity with reasonable accuracy.

desired velocity with reasonable accuracy (Fig. 5). Depending on the desired speed and the cost weights, different kinds of hopping gaits were produced, some with smaller and more uniform jumps and others with larger, less frequent jumps.

IV. Discussion

The results highlight the DDP-based MPC's ability to implicitly discover the best contact conditions to achieve a variety of desired behaviors. The online nature of the controller is able to reject the disturbances from actuator noise and mismatches in dynamics (*i.e.* different simulation rates). My re-implementation of the methods presented in [1], [12] confirms that the concept of combining a relaxed gradient with true hard contact in the simulation is a promising combination for contact-implicit DDP.

A. Limitations

Various limitations of this work should be noted. First, the implementation is rather slow, currently running at rates much slower than real-time. However, other work [12], [19], [20] has demonstrated the ability to perform online DDP for MPC applications, and it is likely that more code optimization and efficiency is possible. Further, the contact model requires sufficient update rates in order to avoid constraint drift due to the velocity-based contact model [15]. Improvements may be seen by using higher order discretization methods [7] that prevent this ground penetration and allow larger Δt and thus increase speed.

Like any shooting method, the identified optimum is dependent on the initial guess. I proposed a two-step solution procedure where we first solve the static reference condition before trying to solve the dynamic reference condition. However, I did not systematically test the efficacy of this approach. Another way to improve the dependence on the initial guess is to move from single shooting to multiple shooting DDP, as was done in the journal extension from [1] to [12]. Multiple shooting allows initial conditions for select points in the state trajectory, allowing a better conditioned optimization problem that is less likely to land in a bad local minimum [21], [3].

Finally, I observed significant sensitivity between the system performance and the chosen P and R cost matrices. Changing the weights on different state and control variables would result in different selected contact patterns. As suggested in [1], a more informative cost function could be used to better elicit the desired behavior and potentially lessen the importance of rigorous gain tuning. For instance, a term that encourages periodic leg swinging could make the gait more regular.

B. Open Questions and Future Work

There are various questions within contact-implicit DDP that warrant further investigation. The first is investigating the mismatch between the forward dynamics and the gradients used in the DDP due to the relaxation and its effects on convergence. In my experience, the DDP iteration never converges after a certain point, likely due to this mismatch. While the gradient calculation says that cost should decrease in a given direction, the forward simulation disagrees due to this mismatch, and the optimization ends up stalling as $\alpha \to 0$ in the line search. Future work should be performed to better understand this effect and perhaps a method involving iteratively tightening the relaxation parameter as the optimization converges should be proposed.

This convergence issue may perhaps explain why the system does not seem to reach a steady state, even with steady state references (*e.g.* Fig. 3). I would expect that after the initial transients dissipate, the robot would settle into a consistent gait. However, the gait continues to have slight variation. Perhaps slight numerical differences and the lack of full convergence of the DDP iterations are to blame. A more rigorous understanding of this phenomenon may be beneficial.

Finally, it is still unclear how to best combine contact-implicit approaches with contact-explicit approaches. Perhaps a framework combining contact-explicit approaches in the near horizon with contact-implicit in the distant horizon could be beneficial, similar to [20]. This may allow for more consistent planning in the near term, while still preserving the flexibility of contact-implicit planning in the far term.

V. Conclusion

In this work, I implemented the contact-implicit DDP MPC approach presented in [1], [12]. The addition of a relaxed complementarity condition in the calculation of the contact gradient allows for the controller to discover new contact modes and better minimize the cost function. I demonstrated the controller's efficacy in various simulation experiments, highlighting its ability to perform a diverse class of behaviors involving making and breaking contact. Finally, I discussed various limitations of my implementation and posed multiple avenues for future work to investigate this promising methodology.

References

[1] G. Kim, D. Kang, J.-H. Kim, and H.-W. Park, “Contact-Implicit Differential Dynamic Programming for Model Predictive Control with Relaxed Complementarity Constraints,” *2022 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, vol. 00, pp. 11 978–11 985, 2022.

[2] E. R. Westervelt, J. W. Grizzle, C. Chevallereau, J. H. Choi, and B. Morris, *Feedback Control of Dynamic Bipedal Robot Locomotion*. CRC Press, 10 2018.

[3] P. M. Wensing, M. Posa, Y. Hu, A. Escande, N. Mansard, and A. D. Prete, “Optimization-Based Control for Dynamic Legged Robots,” *IEEE Transactions on Robotics*, vol. 40, pp. 43–63, 2023.

[4] A. Ibanez, P. Bidaud, and V. Padois, “Emergence of humanoid walking behaviors from mixed-integer model predictive control,” in *2014 IEEE/RSJ International Conference on Intelligent Robots and Systems*, 2014, pp. 4014–4021.

[5] D. E. Stewart and J. C. Trinkle, “An Implicit Time-stepping Scheme For Rigid Body Dynamics With Inelastic Collisions And Coulomb Friction,” *International Journal for Numerical Methods in Engineering*, vol. 39, no. 15, pp. 2673–2691, 1996.

[6] M. Posa, C. Cantu, and R. Tedrake, “A direct method for trajectory optimization of rigid bodies through contact,” *The International Journal of Robotics Research*, vol. 33, no. 1, pp. 69–81, 2014.

[7] Z. Manchester, N. Doshi, R. J. Wood, and S. Kuindersma, “Contact-implicit trajectory optimization using variational integrators,” *The International Journal of Robotics Research*, vol. 38, no. 12-13, pp. 1463–1476, 2019.

[8] D. Mayne, “A Second-order Gradient Method for Determining Optimal Trajectories of Non-linear Discrete-time Systems,” *International Journal of Control*, vol. 3, no. 1, pp. 85–95, 1965.

[9] D. Ruxton, “Differential dynamic programming and optimal control of inequality constrained continuous dynamic systems,” *Master’s Thesis, CQUniversity*, 12 1991.

[10] Y. Tassa, N. Mansard, and E. Todorov, “Control-Limited Differential Dynamic Programming,” *2014 IEEE International Conference on Robotics and Automation (ICRA)*, pp. 1168–1175, 2014.

[11] Y. Tassa, T. Erez, and E. Todorov, “Synthesis and Stabilization of Complex Behaviors through Online Trajectory Optimization,” 2012.

[12] G. Kim, D. Kang, J.-H. Kim, S. Hong, and H.-W. Park, “Contact-implicit mpc: Controlling diverse quadruped motions without pre-planned contact modes or trajectories,” 2023.

[13] K. Werling, D. Omens, J. Lee, I. Exarchos, and C. K. Liu, “Fast and feature-complete differentiable physics engine for articulated rigid bodies with contact constraints,” in *Robotics: Science and Systems*, 2021.

[14] R. Featherstone, *Rigid body dynamics algorithms*. Springer, 2007.

[15] J. Hwangbo, J. Lee, and M. Hutter, “Per-Contact Iteration Method for Solving Contact Dynamics,” *IEEE Robotics and Automation Letters*, vol. 3, no. 2, pp. 895–902, 2018.

[16] J. Carpentier and N. Mansard, “Analytical Derivatives of Rigid Body Dynamics Algorithms,” in *Robotics: Science and Systems (RSS 2018)*, Pittsburgh, United States, Jun. 2018.

[17] H. Li, W. Yu, T. Zhang, and P. M. Wensing, “A Unified Perspective on Multiple Shooting In Differential Dynamic Programming,” *2023 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, vol. 00, pp. 9978–9985, 2023.

[18] L. Armijo, “Minimization of functions having Lipschitz continuous first partial derivatives.” *Pacific Journal of Mathematics*, vol. 16, no. 1, pp. 1 – 3, 1966.

[19] C. Mastalli, R. Budhiraja, W. Merkt, G. Saurel, B. Hammoud, M. Naveau, J. Carpentier, L. Righetti, S. Vijayakumar, and N. Mansard, “Crocoddyl: An Efficient and Versatile Framework for Multi-Contact Optimal Control,” in *IEEE International Conference on Robotics and Automation (ICRA)*, 2020.

[20] H. Li and P. M. Wensing, “Cafe-mpc: A cascaded-fidelity model predictive control framework with tuning-free whole-body control,” *arXiv preprint*, 2024.

[21] H. Li, W. Yu, T. Zhang, and P. M. Wensing, “A Unified Perspective on Multiple Shooting In Differential Dynamic Programming,” *2023 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, vol. 00, pp. 9978–9985, 2023.