Bilevel Optimization for Planning through Contact: A Semidirect Method

Benoit Landry[1], Joseph Lorenzetti[1], Zachary Manchester[1], and Marco Pavone[1]

Stanford University, Stanford CA 94305, USA,
{blandry,jlorenze,zacmanchester,pavone}@stanford.edu

Abstract. Many robotics applications, from object manipulation to locomotion, require planning methods that are capable of handling the dynamics of contact. Trajectory optimization has been shown to be a viable approach that can be made to support contact dynamics. However, the current state-of-the art methods remain slow and are often difficult to get to converge. In this work, we leverage recent advances in bilevel optimization to design an algorithm capable of efficiently generating trajectories that involve making and breaking contact. We demonstrate our method's efficiency by outperforming an alternative state-of-the-art method on a benchmark problem. We moreover demonstrate the method's ability to design a simple periodic gait for a quadruped with 15 degrees of freedom and four contact points.

Keywords: bilevel optimization, hard contact, trajectory optimization

1 Introduction

Trajectory optimization is a well-established method to generate motion plans for dynamical systems. One important application for such methods is motion planning and control for systems with contact dynamics, which are ubiquitous in real-world robotic systems. However, trajectory optimization for systems undergoing contact has proven to be inherently difficult due to the drastically changing dynamical behavior arising from making and breaking contact. For this reason, methods have been proposed which generally fall under several categories. Those using continuous models, such as spring-damper models, have been proposed to eliminate the discontinuities that make the problem so challenging. However in these methods approximation error is induced and numerical problems can arise due to stiffness in the resulting differential equations. Additionally, approaches using hybrid models [1] have been introduced which can richly capture the true discontinuous dynamics. However it is well known that trajectory optimization formulations using such models, primarily mixed-integer programs, suffer from computational limitations resulting from a combinatorially large number of mode switching possibilities.

An alternative class of approaches, which is the focus of this work, includes the contact forces as decision variables in a trajectory optimization framework. These methods, referred to as planning "through contact", utilize techniques

made popular for simulation of contact-driven dynamical systems. The primary advantage of this category of methods is that rich contact behavior can be captured without the combinatorial explosions in complexity seen in hybrid model based methods. However, even though the formulation of such optimization problems is generally straightforward and the resulting problem noncombinatorial, they remain numerically challenging for optimization algorithms to handle. Therefore, the exploration of different formulations of the embedded contact dynamics problem and exploration of different solution techniques are both tasks of high importance.

In this work, we propose a variation of the "through contact" method first put forward in [2]. Our approach leverages recent advances in bilevel and differentiable optimization in which a lower level optimization problem is embedded within an upper level optimization. A key aspect of our algorithm is that it gets rid of the complementarity constraints usually needed to model *friction* forces (which tend to be the cause of many numerical difficulties) and only requires complementarity constraints relating to non-penetration constraints (i.e. for the normal forces). Instead, we compute friction forces as an embedded optimization problem. Note that variations on the idea of modeling contact through bilevel optimization have been explored in the past [3]. However, our formulation posses several appealing characteristics such as being able to leverage state-of-the-art off-the-shelf solvers like SNOPT and OSQP in order to outperform alternative approaches. Moreover our method avoids linearizing the non-penetration constraint which can easily lead to infeasible trajectories. Finally, unlike approaches relying on iLQR, our direct transcription based method can easily and effectively be parallelized, which is of particular relevance when modeling contact forces as solutions to embedded optimization problems.

An important note the reader: in this work when discussing approaches to trajectory optimization through contact we use the term "direct", "indirect", and "semidirect" to refer to the way *the contact force is resolved* within the trajectory optimization problem, and not the way the problem itself is solved *overall* (for which we do in fact use a direct method). Specifically, by a "direct" method we mean a method which defines the contact forces in the trajectory optimization problem completely as the result of an embedded optimization problem (i.e. via the maximum dissipation principle), and by an "indirect" method we mean a method which completely resolves the contact forces through the use of complementarity constraints (which can be interpreted as KKT conditions) [2]. We refer to our proposed method as "semidirect" because it handles *only the friction forces* using an embedded optimization problem, and the normal forces through complementarity constraints.

1.1 Statement of Contributions

The contributions of this paper are as follows. First, by combining different modeling approaches to contact forces, we formulate a "semidirect" trajectory optimization method for planning through contact. Second, we demonstrate how our

formulation, a specific bilevel optimization problem, is well suited to leverage recent advances in differentiable optimization as well as state-of-the-art quadratic programming and nonlinear programming solvers. Next, we demonstrate how our method, based on direct transcription, can easily and effectively leverage parallelization to offset the cost incurred by its bilevel structure. Finally we provide evidence that the method offers a promising direction of research by demonstrating it is capable of outperforming a well accepted alternative method on a benchmark problem, and also generate contact rich interactions between a complex robot and its environment.

2   Previous Work

Here we survey methods related to "through contact" trajectory optimization and how they differ from our proposed algorithm.

One of the first instance of such methods [2] leverages work on rigid body simulation with contact [4] in order to formulate a direct transcription problem capable of generating trajectories that make and break contact. We differ from this approach mainly in that instead of using complemetarity constraint to enforce contact dynamics, we use the maximum dissipation principle to model the friction forces as an embedded quadratic program, and leverage advances in bilevel optimization to handle the newly formed problem. Note that modeling contact forces as optimization problems using the maximum dissipation principle is not an entirely new idea and has been explored in the past. Notably [5] uses this approach to simulate forward contact dynamics and provides a survey of difference instances of this modeling approach.

Perhaps closest to our method, [3] demonstrated a method for planning through contact that also leverages a contact dynamics formulation derived from the principle of maximum dissipation. However, our method differentiates itself from this work on a few points. First, we use a direct collocation method instead of iLQR (a shooting method), which allows us to parallelize the computationally more demanding aspects of the method. Second, instead of solving the lower problem with a projected gradient descent algorithm, we use an off-the-shelf quadratic program solver OSQP which provides faster convergence through the Alternating Direction Method of Multipliers (ADMM). Third, we recover the gradients of the embedded problem analytically, instead of autodifferentiating the solver itself, which again is more computationaly efficient. Lastly, we preserve the normal force as decision variables in the upper-level problem.

Finally, there has also been some work on using the analytical derivatives of quadratic programs in the context of contact. Notably [6] embeds a non-convex quadratic program corresponding to the Linear Complementarity Problem (LCP) described in [4] to learn dynamics involving contact. Our method differs from this work in that we perform nonlinear optimization with an off-the-shelf constrained nonlinear programming solver to solve the upper problem instead of stochastic gradient descent. Moreover, the formulations of our lower problem differ as we do not use the non-convex LCP contact formulation, but

rather the convex maximum dissipation optimization problem (from which the LCP formulation can in fact be derived from as KKT conditions).

3   Contact Dynamics as Optimization Problems

A powerful way of understanding the nature of contact forces is to reason about them using the principle of maximum dissipation. As we show in this section, by performing a first-order discretization of the system dynamics using a backward Euler integration scheme, it is possible to then model the contact forces as the solution of a quadratic program.

3.1   Maximum Dissipation

We start with the rigid body dynamics written in manipulator form

$$M(q)\dot{v} + c(q, v) + \lambda = \tau, \tag{1}$$

where q, v are the configuration and joint velocities of the system. The matrix $M(q)$ is the positive-definite inertia matrix, the function $c(q, v)$ is the dynamics bias (i.e. the Coriolis and potential terms of the manipulator equation combined), $\lambda(q, v)$ denotes the contact forces and τ represents the control input (usually joint torques for robotic systems). This dynamics equation is then discretized in time using a backward Euler integration scheme to yield

$$M(q_{i+1})(v_{i+1} - v_i) + hc(q_{i+1}, v_{i+1}) - h\tau_{i+1} + h\lambda_{i+1} = 0,$$
$$q_{i+1} = q_i + hv_{i+1}, \tag{2}$$

where h is the discretization time step.

Next, we define the contact force acting at each contact point as a vector in a frame that is centered at the contact location. The normal force component of the contact force vector at a given point is denoted by c_n, and the (tangential) friction forces by f_x and f_y. Thus the vector $x = [c_n, f_x, f_y]^T$ defines the contact force in the associated contact point frame. With this definition, the dynamics can be written as

$$M(q_{i+1})(v_{i+1} - v_i) + hc(q_{i+1}, v_{i+1}) - h\tau_{i+1} + hJ(q_{i+1})^T x_{i+1} = 0.$$
$$q_{i+1} = q_i + hv_{i+1}, \tag{3}$$

where $J(q)$ is the Jacobian that maps the contact forces from the contact frame to the joint space.

In order for us to be able to use the principle of maximum dissipation when determining the contact forces, we now derive an expression for the change in kinetic energy from one time step to the next

$$
\begin{aligned}
dT &= T_{i+1} - T_i, \\
&= \frac{1}{2} v_{i+1}^T M_{i+1} v_{i+1} - \frac{1}{2} v_i^T M_i v_i, \\
&= \frac{1}{2} (v_i - h M_{i+1}^{-1} (J_{i+1}^T x_{i+1} + c_{i+1} - \tau_{i+1}))^T M_{i+1} \\
&\qquad (v_i - h M_{i+1}^{-1} (J_{i+1}^T x_{i+1} + c_{i+1} - \tau_{i+1})) - \frac{1}{2} v_i^T M_i v_i, \\
&= \frac{1}{2} h^2 (J_{i+1}^T x_{i+1} + c_{i+1} - \tau_{i+1})^T M_{i+1}^{-1} (J_{i+1}^T x_{i+1} + c_{i+1} - \tau_{i+1}) \\
&\qquad - h v_i^T (J_{i+1}^T x_{i+1} + c_{i+1} - \tau_{i+1}) + \frac{1}{2} v_i^T (M_{i+1} - M_i) v_i.
\end{aligned}
\tag{4}
$$

where dT is the change in kinetic energy, and for notational simplicity we use $M_i = M(q_i)$, $c_i = c(q_i, v_i)$, and $J_i = J(q_i)$. Note that dT can be expressed quadratically in terms of the contact force x. This now allows us to formulate a quadratic program to find the contact force x that maximizes the energy dissipation.

3.2 Friction Forces as Quadratic Program

We can treat q_i, v_i, τ_{i+1}, $M(q_i)$, $M(q_{i+1})$, $c(q_{i+1}, v_{i+1})$ and $J(q_{i+1})$ as known quantities, because they are *for the lower solver* in the context of bilevel optimization. Therefore, to find the contact force that provides maximal energy dissipation we seek to minimize the quantity dT in (4). This is equivalent to minimizing the following quadratic function.

$$
\frac{1}{2} x_{i+1}^T Q_{i+1}^d x_{i+1} + (r_{i+1}^d)^T x_{i+1},
\tag{5}
$$

where

$$
\begin{aligned}
Q_{i+1}^d &= h J_{i+1} M_{i+1}^{-1} J_{i+1}^T, \\
r_{i+1}^d &= \left(h (c_{i+1} - \tau_{i+1})^T M_{i+1}^{-1} - v_i^T \right) J_{i+1}^T.
\end{aligned}
\tag{6}
$$

Additionally we want to ensure that the contact forces satisfy constraints imposed by a Coulomb friction model. To accomplish this, we first linearize the friction cone, as described in [4], such that $D\beta = [0, f_x, f_y]^T$. We also define $z = [c_n, \beta^T]^T$, and therefore can express the contact force vector x as

$$
x = Fz, \quad F := \left[\hat{n} | D \right],
\tag{7}
$$

where $\hat{n}$ is the unit length normal vector at the contact point. Then, we impose upon the energy dissipation maximization problem the Coulomb friction model constraint $\mu c_n - e^T \beta \geq 0$, where μ is the friction coefficient and $e = [1, 1, \ldots, 1]^T$.

The resulting computation of the friction force is therefore given by the following quadratic program

$$\beta = \operatorname*{argmin}_{\beta} \quad \frac{1}{2}z^T F^T Q^d F z + (r^d)^T F z$$

$$\mu c_n - e^T \beta \geq 0,$$
$$\beta \geq 0,$$

(8)

where the time index notation has been dropped for clarity. Note that the formulation of our "semidirect" method allows us to make the assumption here that the normal force c_n is known and is therefore not a decision variable in (8). This will be discussed in further detail next in Section 4.

4   Trajectory Optimization Through Contact

We now show how to formulate the trajectory optimization problem as a bilevel optimization problem using the results from Section 3. As mentioned before, this formulation is "semidirect" in the sense that part of the overall contact force (normal forces) are handled via complementarity constraints, and the other part (friction forces) are handled via the maximum dissipation optimization problem given by (8).

Thus in this formulation the decision variables of the trajectory optimization problem are the control inputs τ, configuration variables q, joint velocities v, and the normal force component c_n, but the friction force vector β will *not* be a decision variable, as it will be implicitly encoded by the embedded optimization problem.

Note that handling the non-penetration constraints, and hence the normal force, in a lower problem is also possible. However since it is a constraint that is a function of position that would be imposed on decision variables relating to forces, keeping the embedded constraint linear would introduce compounding approximations (double integration). This is not the case for the problem of friction since this later set of constraints are functions of velocity (not position) imposed on decision variables relating to forces.

4.1   Dynamics Constraints

For a planning horizon $i = 1, \ldots, m$, from the backward Euler discretized manipulator equation (3) with the contact forces in contact space represented by $x = Fz$ where $z = [c_n, \beta]^T$, we have the dynamics constraints

$$M_{i+1}(v_{i+1} - v_i) + hc_{i+1} + hJ_{i+1}^T F z_{i+1} = h\tau_{i+1},$$
$$q_{i+1} = q_i + hv_{i+1}.$$

(9)

The normal force c_n is then constrained by the set of complementarity conditions

$$
\begin{aligned}
\phi(q_{i+1}) &\geq 0, \\
c_{n,i} &\geq 0, \\
c_{n,i}\phi(q_{i+1}) &= 0,
\end{aligned}
\tag{10}
$$

where $\phi(q)$ is a distance function such that $\phi(q) \geq 0$ implies non-penetration of the rigid body with its environment. Finally, the friction force vector β is given by (8).

4.2 Trajectory Optimization

Given an integrating stage cost function $J(q, v, \tau)$, and problem-specific constraints on the configuration and control defined by $g(q, v, \tau) \leq 0$ and $h(q, v, \tau) = 0$ (i.e. desired initial or final configurations, or actuation limits), the trajectory optimization problem is defined as

$$
\begin{aligned}
\underset{q_i, v_i, \tau_i, c_{n,i}; i=1 \ldots m}{\text{minimize}} \quad & \sum_{i=1}^{m} J(q_i, v_i, \tau_i) \\
\text{subject to} \quad & M_{k+1}(v_{k+1} - v_k) + hc_{k+1} + hJ_{k+1}^T F z_{k+1} = h\tau_{k+1}, \\
& q_{k+1} = q_k + hv_{k+1}, \\
& z_{k+1} = [c_{n,k+1}, \beta_{k+1}]^T, \\
& \phi(q_i) \geq 0, \\
& c_{n_i} \geq 0, \\
& c_{n_i}\phi(q_{i+1}) = 0, \\
& \beta_{k+1} = \underset{\beta_{k+1}}{\operatorname{argmin}} \frac{1}{2} z_{k+1}^T F^T Q_{k+1}^d F z_{k+1} + (r_{k+1}^d)^T F z_{k+1} \\
& \qquad\qquad \mu c_{n,k+1} - e^T \beta_{k+1} \geq 0, \\
& \qquad\qquad \beta_{k+1} \geq 0. \\
& g(q_i, v_i, \tau_i) \leq 0, \\
& h(q_i, v_i, \tau_i) = 0,
\end{aligned}
\tag{11}
$$

where $k = 1, \ldots, m - 1$.

The above problem forms a nonlinear bilevel trajectory optimization problem that can be solved by off-the-shelf nonlinear solvers as long as special care is taken to handle the friction force embedded problem, as described next in Section 5.

5   Solving the Friction Force Lower Problem

Now that we have defined the lower problem of our bilevel optimization, namely the quadratic program (8) whose solution corresponds to the friction forces at each contact point, we discuss our solution method for it.

There exists many solutions methods capable of handling convex quadratic programs similar to the one described in (8). However, in the context of bilevel optimization problems, additional requirements are placed on the solvers for the embedded mathematical program. First, solutions are needed very quickly and with little overhead since the embedded solver is called frequently by the primary solver working on the upper problem (i.e. for every evaluation of the constraints of the upper problem by the primary solver). In order to address this requirement, we leverage a state-of-the-art quadratic program solver OSQP [7], that implements the popular Alternating Direction Method of Multipliers (ADMM) algorithm [8].

Second and perhaps most importantly, gradients of the solution with respect to the parameters of the problem need to be available.

5.1   Gradients

In order to provide gradients of the solution, we leverage work in sensitivity analysis [9,10,11], which has regained traction more recently [12,13,14]. The approach relies on using first order optimality conditions and the implicit function theorem in order to define a system that can be solved to recover the gradient of the solution with respect to the problem's parameters. The details of this derivation are outside the scope of this paper and we refer the reader to [13] for a through treatment of the problem. But for the reader's benefit we include the system that must be solved along with the quadratic program in order to recover the gradients (taken from [14]). Given the following quadratic program:

$$
\begin{aligned}
\underset{x}{\text{minimize}} \quad & \frac{1}{2}x^T Q(\theta)x + q(\theta)^T x \\
\text{subject to} \quad & G(\theta)x \preceq h(\theta), \\
& A(\theta)x = b(\theta),
\end{aligned}
\tag{12}
$$

where θ is the parameter vector for which we are interested in getting the gradients with respect to. We can then compute the desired gradient $D_\theta x^*$ by solving

$$
\Pi \begin{bmatrix} D_\theta x^* \\ D_\theta \lambda^* \\ D_\theta \nu^* \end{bmatrix} = z,
\tag{13}
$$

where

$$
\Pi = \begin{bmatrix} Q & G^T & A^T \\ \mathbf{diag}(\lambda^*)G & \mathbf{diag}(Gx^* - h) & 0 \\ A & 0 & 0 \end{bmatrix}, \quad z = \begin{bmatrix} dQx^* + D_\theta q + dG^T\lambda^* + dA^T\nu^* \\ \mathbf{diag}(\lambda)(dGx^* - D_\theta h) \\ dAx^* - D_\theta b \end{bmatrix},
$$

and where x^*, λ^*, ν^* are the optimal solution for the primal, and dual variables (λ is the dual for the inequality constraint and ν is the dual for the equality constraints). Note that the system must often be solved using a least-squares method.

6   Parallelization

One of the main drawback of our approach is that even though the size of the problem (in number of variables and constraints) for the upper problem is reduced, constraint evaluation is now significantly more expensive. Indeed, instead of evaluating a series of inequalities like we would normally be required, constraint evaluation with our approach requires us to solve a mathematical program. However, because we perform our trajectory optimization using a direct transcription method (and not say iLQR as in [3]), we can trivially run our constraint evaluation in parallel. In our implementation, each dynamic constraint (one per successive knot point) is evaluated in parallel as its own thread. The performance gain is therefore dependent on the number of knot points used. When using a computer with sufficiently many cores, this parallelization helps offset the cost incurred by the more computationally expensive constraint evaluation.

7   Results

We now present some results to empirically demonstrate and validate our proposed approach. First, we demonstrate our algorithm on a smaller problem and compare its efficiency with an indirect method which is considered state-of-the-art [2]. Next, we use our algorithm to design a 'stepping forward' motion for a quadruped that can then be used in repetition as a gait.

7.1   Implementation Details

We implement our approach using the Julia programming language [15]. Specifically, the rigid body dynamics (excluding contact dynamics) are computed using the package RigidBodyDynamics.jl [16] and we implement the analytical gradient of the embedded quadratic program inside the ForwardDiff.jl framework [17] (allowing for easy integration with other constraints). Additionally, for the upper problem we use the popular sequential quadratic programming (SQP) solver SNOPT [18], which is often regarded as the most performant solver for trajectory optimization. For the embedded problem, we use the solver OSQP (as described in Section 5). Our experiments were run on a 16-core 3.0 GHz CPU with 32Gb of memory. All of of our code is made available at https://github.com/blandry/Bilevel.jl.

7.2   Performance Benchmark

First we solve a simple planning problem involving sliding a single rigid body on a surface to a target location, where the rigid body is modeled as having a single point of contact with the surface. The initial position of the object is chosen such that it is not in contact with the surface, so that the solver must also determine the contact time. Additionally, a final position constraint is defined, the final velocity is constrained to be zero, and there is no control input along

the trajectory. The problem therefore corresponds to finding the right initial velocity to throw the object such that it slides across a surface to the target position. Note that this task is more challenging than it might appear at first glance, since it requires the trajectory to include transitions from non-contact to sliding friction, and then to sticking friction.

Here we compare the result of solving this trajectory optimization problem using our proposed semidirect method against the indirect method presented in [2]. The solver SNOPT was also used for the approach in [2], and in both methods was allowed to run until optimality (with a tolerance of 10^{-3}). Note that even for this simple example, the indirect method required us to introduce slack variables on the complementarity constraints related to friction (a common trick to handle these constraints). Notably however, since our semidirect method handles the friction contact constraints via the embedded problem given by (8) no slack variables were necessary.

From the trajectory shown in Figure 1 and the results presented in Table 1, we can see that our method not only recovers a solution of comparable quality, but it does so in less time. Our semidirect method is benchmarked at 100ms, while the indirect method required more computation time, at 224ms. Note also that the semidirect method only required SNOPT to solve a problem with 215 variables while the indirect one contained 367. Note that we also perform the constraint evaluation in parallel in our implementation of the indirect method in order to avoid giving our approach a trivial advantage.

	# Variables	Mean solve time (s)
Indirect	367	0.22
Semidirect (ours)	215	0.10

Table 1: Comparison of optimization problem size and solve time between our proposed semidirect method and the indirect method presented in [2], when applied to the problem described in Section 7.2.

7.3   Quadruped Gait Design

Next, we applied our approach to perform trajectory optimization for Boston Dynamics's "little dog". The system and the contact points are modeled as 3-dimensional, but the system is constrained such that its center of mass moves in a vertical plane. The system has 15 degrees of freedom (from the position of the center of mass, orientation in the plane, and leg joint angles) and each leg is modeled with a contact point on its tip.

The problem consists of moving little dog forward by 50 centimeters, starting and ending with zero velocity. The objective is to minimize torques in the joints (which also have upper and lower limits). In order to be able to use the resulting trajectory as a gait, we also enforce that the final configuration (minus the
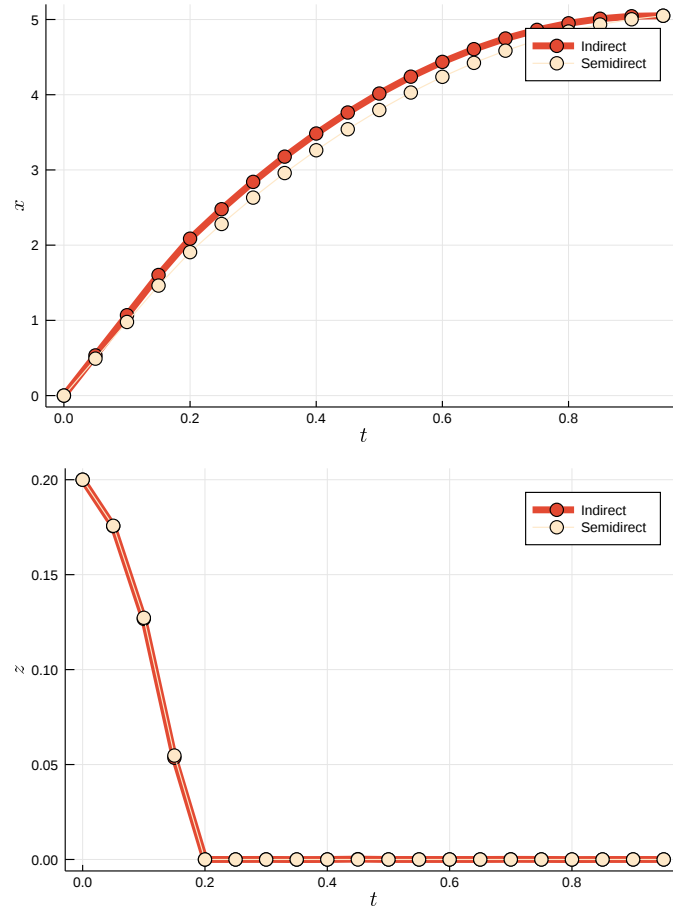
Fig. 1: Comparison of the horizontal (top) and vertical (bottom) position trajectories of the rigid body's center of mass resulting from our proposed semidirect method and the indirect method presented in [2], when applied to the problem described in Section 7.2. Units are in seconds and meters.

forward displacement) matches the initial one. In Figure 2 we show snapshots of the trajectory resulting from our proposed semidirect method, and a video of the resulting gait is also available at https://www.youtube.com/playlist?list=PL8-2mtIlFIJpgmWImauC9rXxgN2wWpcIR.
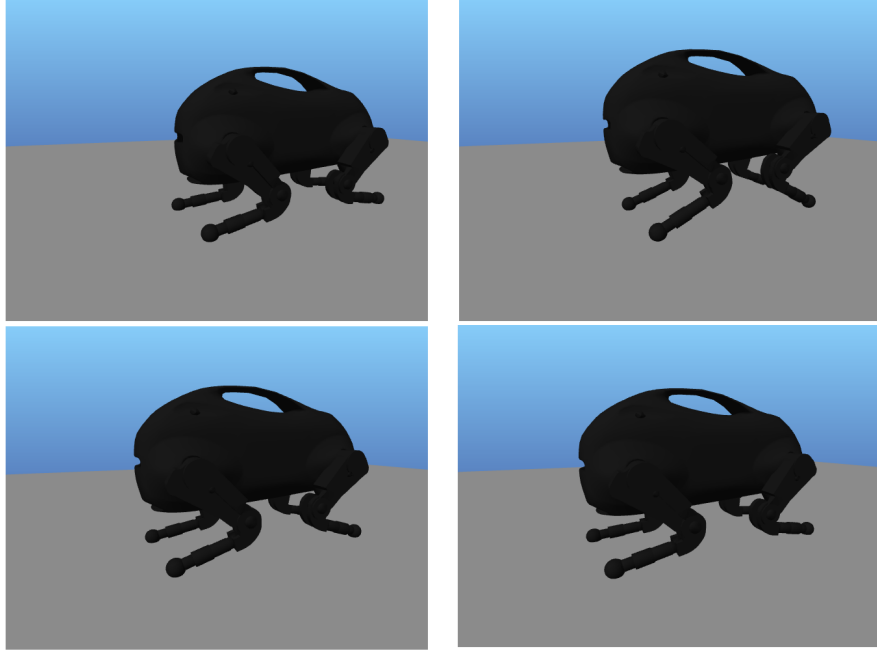

Fig. 2: The gait found by our semidirect method for a quadruped, little dog, to move forwards. The robot first moves its weight on its front legs (upper left), allowing it to slide its back legs forward (upper right). Next, little dog moves its weight on its back legs (bottom left) and slides its front legs forwards (bottom right), returning to its initial configuration. This completes one step of the gait, which interestingly makes effective use of friction forces between the robot and its environment.

Figure 3 shows the total normal force on the front and back legs, clearly demonstrating the non-trivial strategy that the algorithm found to design a gait. For this example, the trajectory optimization solve time is benchmarked at 1.08 seconds. After our best effort, we were unable to get the indirect method from [2] to converge to a good solution for this task and therefore cannot report its performance on it. However an alternative method such as [19], which is specifically tailored to floating rigid bodies, does not report computation times any faster than our own for the same quadruped.
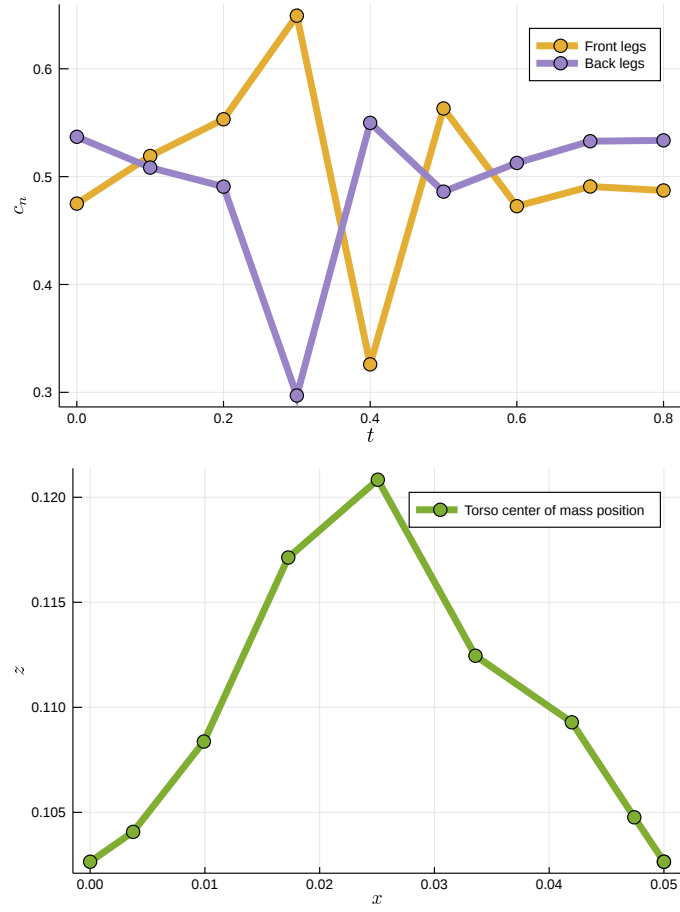
Fig. 3: Normal contact forces at the legs (top) and position of the center of mass (bottom) of little dog during one step of the gait resulting from the trajectory optimization problem described in Section 7.3.

8   Method Shortcomings

We note that even though our approach is capable of efficiently generating a vast array of complex trajectories that involve making and breaking contact, it still leaves many open questions. Notably our strategy (least squares) for choosing a subgradient when the gradient of the lower quadratic problem is not uniquely defined seems to work in practice, but does not rest on a strong theoretical foundation yet, and is perhaps far from optimal for this application. Moreover, in practice, we found that trajectories involving sticking contact seemed to pose a bigger challenge to the upper solver (SNOPT) that would sometimes struggle to improve the trajectory past a certain point, most likely due to the gradients of the lower problem. We believe these are great directions for future research in both theoretical and numerical aspects of our method.

9   Conclusion

In this work we introduced a bilevel optimization approach to robotic trajectory optimization for systems that can make and break contact with their environment. The approach falls under the category of planning "through contact", where the contact constraints are directly resolved within the formulated optimization problem. While similar approaches have been proposed in the past, the novelty of our proposed method is that it formulates the contact constraints in the trajectory optimization in a "semidirect" way. Specifically, the normal force contact is handled indirectly via complementarity constraints in the optimization problem and the friction force is handled directly as the solution to an embedded optimization problem. This allows us to avoid to use of additional complementarity constraints for the friction forces and to avoid linearizing the non-penetration constraints. To demonstrate empirically the advantages of our proposed approach we presented results from two problems: a benchmark involving sticking and sliding friction and a gait optimization for a 15-degrees-of-freedom quadruped.

Aknowledgments

References

1. Bernardo Aceituno-Cabezas, Carlos Mastalli, Hongkai Dai, Michele Focchi, Andreea Radulescu, Darwin G Caldwell, José Cappelletto, Juan C Grieco, Gerardo Fernández-López, and Claudio Semini. Simultaneous contact, gait, and motion planning for robust multilegged locomotion via mixed-integer convex optimization. *IEEE Robotics and Automation Letters*, 3(3):2531–2538, 2017.

2. 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.
3. J. Carius, R. Ranftl, V. Koltun, and M. Hutter. Trajectory optimization with implicit hard contacts. *IEEE Robotics and Automation Letters*, 3(4):3316–3323, 2018.
4. D. Stewart and J. C. Trinkle. An implicit time-stepping scheme for rigid body dynamics with coulomb friction. In *Proc. IEEE Conf. on Robotics and Automation*, 2000.
5. Evan Drumwright and Dylan A Shell. Modeling contact friction and joint friction in dynamic robotic simulation using the principle of maximum dissipation. In *Algorithmic foundations of robotics IX*, pages 249–266. Springer, 2010.
6. F. de Avila Belbute-Peres, K. Smith, K. Allen, J. Tenenbaum, and J. Z. Kolter. End-to-end differentiable physics for learning and control. In *Conf. on Neural Information Processing Systems*, 2018.
7. B. Stellato, G. Banjac, P. Goulart, A. Bemporad, and S. Boyd. OSQP: An operator splitting solver for quadratic programs. *ArXiv e-prints*, November 2017.
8. Stephen Boyd, Neal Parikh, Eric Chu, Borja Peleato, Jonathan Eckstein, et al. Distributed optimization and statistical learning via the alternating direction method of multipliers. *Foundations and Trends® in Machine learning*, 3(1):1–122, 2011.
9. A. B. Levy and R. T. Rockafellar. Sensitivity of solutions in nonlinear programming problems with nonunique multipliers. In *Recent Advances in Nonsmooth Optimization*. World Scientific Publishing, 1995.
10. A. V. Fiacco and Y. Ishizuka. Sensitivity and stability analysis for nonlinear programming. *Annals of Operations Research*, 27(1):215–235, 1990.
11. D. Ralph and S. Dempe. Directional derivatives of the solution of a parametric nonlinear program. *Mathematical Programming*, 70(1-3):159–172, 1995.
12. B. Amos and J. Z. Kolter. Optnet: Differentiable optimization as a layer in neural networks. In *Int. Conf. on Machine Learning*, 2017.
13. Akshay Agrawal, Shane Barratt, Stephen Boyd, Enzo Busseti, and Walaa M Moursi. Differentiating through a conic program. *arXiv preprint arXiv:1904.09043*, 2019.
14. Shane Barratt. On the differentiability of the solution to convex optimization problems. *arXiv preprint arXiv:1804.05098*, 2018.
15. J. Bezanson, S. Karpinski, V. B. Shah, and A. Edelman. Julia: A fast dynamic language for technical computing, 2012. Available at http://arxiv.org/abs/1209.5145.
16. Twan Koolen and contributors. Rigidbodydynamics.jl, 2016.
17. J. Revels, M. Lubin, and T. Papamarkou. Forward-mode automatic differentiation in julia, 2016. Available at https://arxiv.org/abs/1607.07892.
18. P. E. Gill, W. Murray, and M. A. Saunders. SNOPT: An SQP algorithm for large-scale constrained optimization. *SIAM Review*, 47(1):99–131, 2005.
19. 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.