Rapid Convex Optimization of Centroidal Dynamics using Block Coordinate Descent

Paarth Shah*[1], Avadesh Meduri*[2], Wolfgang Merkt[1], Majid Khadiv[3], Ioannis Havoutis[1], Ludovic Righetti[2,3]

***Abstract*— In this paper we explore the use of block coordinate descent (BCD) to optimize the centroidal momentum dynamics for dynamically consistent multi-contact behaviors. The centroidal dynamics have recently received a large amount of attention in order to create physically realizable motions for robots with hands and feet while being computationally more tractable than full rigid body dynamics models. Our contribution lies in exploiting the structure of the dynamics in order to simplify the original non-convex problem into two convex subproblems. We iterate between these two subproblems for a set number of iterations or until a consensus is reached. We explore the properties of the proposed optimization method for the centroidal dynamics and verify in simulation that motions generated by our approach can be tracked by the quadruped Solo12. In addition, we compare our method to a recently proposed convexification using a sequence of convex relaxations as well as a more standard interior point method used in the off-the-shelf solver IPOPT to show that our approach finds similar, if not better, trajectories (in terms of cost), and is more than four times faster than both approaches. Finally, compared to previous approaches, we note its practicality due to the convex nature of each subproblem which allows our method to be used with any off-the-shelf quadratic programming solver.**

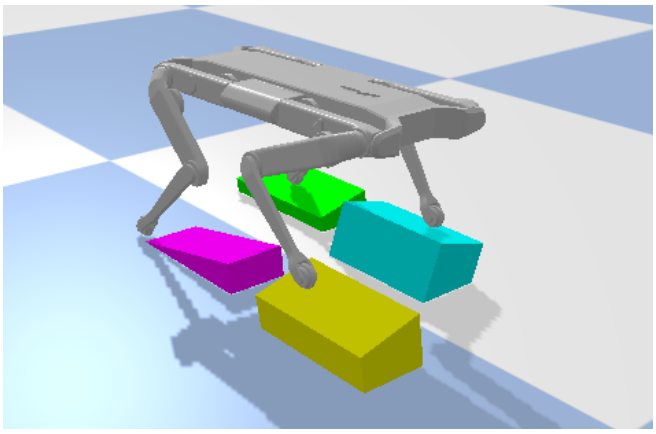


Fig. 1. Simulation of a trajectory computed using our method on complex terrain with the quadruped Solo12.

I. Introduction

The use of optimization to generate movements for robots with hands and feet has been studied extensively over the past years. The problem is inherently complex due to the nonlinear nature of the dynamics, the non-convex cost landscape, and the requirement that computed trajectories must eventually be executable on real robots.

In order to generate trajectories for online control, early research focused on planning using template models [1]. These simplified models are low-dimensional approximations that capture the nature of the dynamics and frequently remove the nonlinearities and non-convexities which allows fast online re-computation due to their linear nature. The most widely studied simplified model in humanoid control has been the linear inverted pendulum and its many variations [2], [3], [4], [5]. In these works, the linearity is exploited to efficiently solve for center of mass trajectories, footstep locations, or both, online. Although these methods have proven to be highly effective, they do not generalize to arbitrary terrains due to the inherent assumptions made in their formulations and focus on legged locomotion rather than the arbitrary multi-contact problem (e.g. using hands) which limits their versatility. Recently, there has been an increasing interest in developing full body motions for floating base robots using the centroidal dynamics model [6]. The centroidal dynamics are a reduced order representation of the full dynamics of the robot that considers the momentum wrench at the center of mass. One of the main benefits to this approach is the ability to utilize the full potential of the entire robot (e.g. arms and legs) to interact with arbitrary environments while also obeying the rigid body dynamics. So far, these methods have shown impressive results [7], [8], [9], [10], with the latter two demonstrating capabilities on hardware itself.

The centroidal dynamics, however, are non-convex, which makes motion planning problems difficult to solve. The authors of [11] use a worst-case ℓ_1 bound on the angular momentum in order to make the problem convex. [9] used an efficient multiple shooting approach, but to the best of our knowledge, this implementation is closed-source due to use of a proprietary solver, MUSCOD-II. In [8], the non-convexity was dealt with by using a difference of quadratic functions which was further exploited and optimized in [10]. In addition to providing the decomposition approach, [8], [7] also proposed a method for creating full-body motions using

*These authors contributed equally

[1]Oxford Robotics Institute, University of Oxford, England. {paarth,wolfgang,ioannis}@robots.ox.ac.uk

[2]Tandon School of Engineering, New York University, Brooklyn, USA. {am9789,ludovic.righetti}@nyu.edu

[3]Max Planck Institute for Intelligent Systems, Tübingen, Germany. mkhadiv@tuebingen.mpg.de

This work was supported in part by New York University, the European Union's Horizon 2020 research and innovation program (grant agreement 780684), the National Science Foundation (grants 1825993, 1932187, 1925079 and 2026479), the UKRI/EPSRC with grants [EP/S002383/1], [EP/R026084/1] and [EP/R026173/1]. This work was part of the Human-Machine Collaboration Programme, supported by a gift from Amazon Web Services.

[1]Paarth Shah was supported by an AWS Lighthouse Scholarship

an iterative approach of alternating the optimization of the centroidal dynamics and whole-body kinematics. Although the results from [10] were impressive, their framework needs to solve second-order cone programs which are more computationally demanding than solving simple quadratic programs (QP). Specifically, they used a customized variant of the ECOS solver [12], an interior-point solver which is therefore difficult to warm-start for model-predictive control applications.

In this paper, we propose a block coordinate descent (BCD) approach to solve the centroidal dynamics trajectory optimization problem [13]. The main idea of the approach is to leverage the inherent structure of the problem in order to simplify the non-convexity into two simpler, convex subproblems. By utilizing the sparse structure of the multi-contact locomotion problem, we show that we are able to efficiently generate and track different types of whole-body motions including challenging maneuvers that require tight tracking of angular momentum.

Unlike previous methods which rely on complicated decomposition procedures and customized solvers (both of which are difficult to implement), our approach can instead be easily implemented using any off-the-shelf QP solver.

The proposed method also has guaranteed convergence to a feasible solution under the assumption that the problem is well-posed unlike methods that rely on off-the-shelf nonlinear solvers such as IPOPT [14]. Although the solutions may not converge to a local minima, we are guaranteed to converge to feasible motions at every iteration [15]. We argue that feasibility, in terms of constraint satisfaction, is more important than optimality, as the local minima found by off-the-shelf solvers are often arbitrary. We show that in practice, our algorithm converges quickly and finds high quality trajectories that can be tracked in simulation.

Finally, we find that our method is often multiple times faster than the state of the art. Due to the structure of our optimal control problem as well as the convexity of each subproblem, we are able to leverage the maturity of QP solvers that offer features that can inherently exploit the sparsity patterns in our problem.

II. Whole Body Trajectory Optimization

The equations of motion for a floating-based rigid body dynamics robot can be written as

$$\boldsymbol{M}(\boldsymbol{q})\ddot{\boldsymbol{q}} + \boldsymbol{h}(\boldsymbol{q}, \dot{\boldsymbol{q}}) = \boldsymbol{S}^T\boldsymbol{\tau}_j + \sum_{i=0}^{n_c} \boldsymbol{J}_i^T \boldsymbol{\lambda}_i \tag{1}$$

where $\boldsymbol{q} = [\boldsymbol{x}^T \ \boldsymbol{q}_j^T]^T$ describes the robot configuration and comprises both the floating base position and orientation expressed with respect to a fixed inertial frame, $\boldsymbol{x} \in \mathbb{SE}(3)$, and joint positions, $\boldsymbol{q}_j$, of the robot. $\boldsymbol{M}(\boldsymbol{q}) \in \mathbb{R}^{(n_j+6)\times(n_j+6)}$ is the mass inertia matrix, $\boldsymbol{h}(\boldsymbol{q}, \dot{\boldsymbol{q}}) \in \mathbb{R}^{(n_j+6)}$ contains the Coriolis, centrifugal, gravity, and friction forces, $\boldsymbol{S} = [\boldsymbol{0}^{n_j\times 6} \ \boldsymbol{I}^{n_j\times n_j}]$ is the actuator selection matrix that defines the underactuation of the robot, $\boldsymbol{\tau}_j \in \mathbb{R}^{n_j}$ is the vector of joint torques, $\boldsymbol{J}_i$ are the end-effector jacobians and $\boldsymbol{\lambda}_i$ are the forces due to external contacts acting on the robot.

For underactuated robots (i.e. robots with more degrees of freedom than number of controllable joints), we can decompose our dynamics into the actuated (subscript a) and unactuated (subscript u) dynamics as follows

$$\boldsymbol{M_a}(\boldsymbol{q})\ddot{\boldsymbol{q}} + \boldsymbol{h}_a(\boldsymbol{q}, \dot{\boldsymbol{q}}) = \boldsymbol{\tau}_j + \sum_{i=0}^{n_c} \boldsymbol{J}_{i,a}^T \boldsymbol{\lambda}_i \tag{2a}$$

$$\boldsymbol{M_u}(\boldsymbol{q})\ddot{\boldsymbol{q}} + \boldsymbol{h}_u(\boldsymbol{q}, \dot{\boldsymbol{q}}) = \sum_{i=0}^{n_c} \boldsymbol{J}_{i,u}^T \boldsymbol{\lambda}_i \tag{2b}$$

Equation (2b) describes the change in momentum of the robot given external forces, λ. As previously described in [8], the actuated part of the dynamics provides the necessary torques to achieve combinations of the desired accelerations, $\ddot{\boldsymbol{q}}$, and contact forces, λ. Under the assumption of enough actuation torque, this allows us to ignore the actuated part of the dynamics, and focus solely on creating motions for the underactuated floating base using purely the external forces and torques. The underactuated dynamics are equivalent to the centroidal dynamics of the robot [16] and when expressed at the robot center of mass (CoM) can be written as

$$\dot{\boldsymbol{h}} = \begin{bmatrix} \dot{\boldsymbol{r}} \\ \dot{\boldsymbol{l}} \\ \dot{\boldsymbol{k}} \end{bmatrix} = \begin{bmatrix} \frac{1}{m}\boldsymbol{l} \\ m\boldsymbol{g} + \sum_e \boldsymbol{f}_i \\ \sum_e (\boldsymbol{p}_e + \boldsymbol{R}_{e,t}^{x,y}\boldsymbol{z}_e - \boldsymbol{r}) \times \boldsymbol{f}_e + \boldsymbol{\tau}_e \end{bmatrix}. \tag{3}$$

Here, $\boldsymbol{r}$ is the center of mass location, $\boldsymbol{l}$ is the linear momentum of the center of mass, $\boldsymbol{k}$ is the angular momentum of the center of mass, $\boldsymbol{f}_e$ are the external forces on the robot, $\boldsymbol{p}_e$ are the robot end-effector locations in the inertial frame, $\boldsymbol{z}_e$ the centers of pressure for each contact, $\boldsymbol{R}_{e,t}^{x,y} \in \mathbb{R}^{3\times 2}$ are the first two columns of the rotation matrix $\boldsymbol{R}_{e,t}$ which rotates maps quantities from end-effector frame to inertial frame, $\boldsymbol{\tau}_e$ the torques at each center of pressure (e.g. torques induced by flat feet of a robot leg), m, is the robot mass, and $\boldsymbol{g}$ is the gravity vector.

Using this form, [8] and [7] suggested a decomposition to create whole-body motions. [8] proposed an alternating method for the kinematics and dynamics of the robot by finding dynamically feasible trajectories using the centroidal dynamics then solving an inverse kinematics problem for the whole body of the robot. The output of this alternating process are whole-body motions that can then be tracked by a controller such as in [17].

A. Trajectory Optimization of Centroidal Dynamics

In this paper, we focus on the optimization of the centroidal dynamics, Eq. (3). Specifically, we are looking to find dynamically feasible trajectories, i.e. motions that optimize for the end-effector forces and torques subject to the non-convex constraints of the centroidal momentum. We assume that a contact surface is given and that the timing of each contact is fixed. Contact sequences can be found using a contact planner [18], [19], [20]. The optimization problem

we are trying to solve can be written as follows

$$\min_{\boldsymbol{h},\boldsymbol{p_e},\boldsymbol{f_e},\boldsymbol{\tau_e},\boldsymbol{z}} \quad \sum_{t=0}^{N} \Psi_t(\boldsymbol{h},\boldsymbol{p_e},\boldsymbol{z_e},\boldsymbol{f_e},\boldsymbol{\tau_e}) \;+\; \phi_t(\boldsymbol{h}_t - \boldsymbol{h}_t^{kin}) \tag{4a}$$

$$\text{s.t.} \quad \boldsymbol{h}_t = \begin{bmatrix} \boldsymbol{r}_t \\ \boldsymbol{l}_t \\ \boldsymbol{k}_t \end{bmatrix} = \begin{bmatrix} \boldsymbol{r}_{t-1} + \frac{1}{m}\boldsymbol{l}\Delta t \\ \boldsymbol{l}_{t-1} + m\boldsymbol{g}\Delta t + \sum_e \boldsymbol{f_{e,t}}\Delta t \\ \boldsymbol{k}_{t-1} + \sum_e \boldsymbol{\kappa}_{e,t}\Delta t \end{bmatrix} \tag{4b}$$

$$\boldsymbol{\kappa}_{e,t} = (\boldsymbol{p}_{e,t} - \boldsymbol{r_t}) \times \boldsymbol{f}_{e,t} + \boldsymbol{\gamma}_{e,t} \tag{4c}$$

$$\boldsymbol{\gamma}_{e,t} = (\boldsymbol{R}_{e,t}^{x,y}\boldsymbol{z}_{e,t}) \times \boldsymbol{f}_{e,t} + \boldsymbol{\tau}_{e,t} \tag{4d}$$

$$\boldsymbol{p_{e,t}} \in \mathcal{U}(\mathcal{S}) \tag{4e}$$

$$\boldsymbol{z}_{e,t}^{x,y} \in [{}^{min}\boldsymbol{z}^{x,y}, {}^{max}\boldsymbol{z}^{x,y}] \tag{4f}$$

$$|\boldsymbol{f}_{e,t}^{x}| \leq \; \boldsymbol{R}_{e,t}\boldsymbol{f}_{e,t}^{z},\; |\boldsymbol{f}_{e,t}^{y}| \leq \; \boldsymbol{R}_{e,t}\boldsymbol{f}_{e,t}^{z},\; \boldsymbol{R}_{e,t}\boldsymbol{f}_{e,t}^{z} \geq 0 \tag{4g}$$

$$\| \boldsymbol{p}_{e,t} - \boldsymbol{r}_t \| \leq \mathcal{L}^{max} \tag{4h}$$

where constraints (4g) are the pyramidal friction cone constraints, (4h) is an ℓ_1-norm approximation of the kinematic limit of the end-effectors, and (4a) minimizes a quadratic sum of the running cost on the discretized dynamics of the state Ψ and cost of tracking the output of the kinematic optimization, $\boldsymbol{\phi_t}$. We note that while some centroidal optimization approaches [10] uses second-order cones (ℓ_2-norms) for both the kinematic limit and friction cone constraints, we use linear approximations for both. Solving this problem efficiently is in general hard due to the cross product in (4c) and (4d) which introduce non-convex constraints.

III. Block Coordinate Descent

In this section, we give a brief overview of the block coordinate descent method used in the subsequent sections of the paper. We first detail the main idea and general approach, and later discuss the convergence properties of the chosen formulation and update methodology.

We are interested in optimization problems of the form

$$\min_{\boldsymbol{x}\in X} \quad F(\boldsymbol{x}_1, \cdots, \boldsymbol{x}_s) + \sum_{i=1}^{s} r_i(\boldsymbol{x}_i) \tag{5}$$

where the variable $\boldsymbol{x}$ is decomposed into $\boldsymbol{s}$ blocks, the set X is closed and a block multi-convex subset of $\mathbb{R}^n$. Note that the set X may be non-convex over $\boldsymbol{x} = (\boldsymbol{x}_1, \cdots, \boldsymbol{x}_s)$. r_i are extended value functions which mean $r_i(\boldsymbol{x}_i) = \infty$ if $\boldsymbol{x}_i \notin \text{dom}(r_i)$ and can be used as indicator functions for convex sets.

We call a set X block multi-convex if each block of variables is convex, that is, for each i and fixed $(s-1)$ blocks $\boldsymbol{x}_1, \cdots, \boldsymbol{x}_{i-1}, \cdots, \boldsymbol{x}_s$ the set

$$\begin{aligned} &X_i(\boldsymbol{x}_1 ... \boldsymbol{x}_{i-1}, \boldsymbol{x}_{i+1}, ... \boldsymbol{x}_s) \triangleq \\ &\{\boldsymbol{x}_i \in \mathbb{R}^n : (\boldsymbol{x}_1, ..., \boldsymbol{x}_{i-1}, \boldsymbol{x}_i, \boldsymbol{x}_{i+1}, ... \boldsymbol{x}_s) \in X\} \end{aligned} \tag{6}$$

is convex. We can then see, when all blocks except one are fixed, the function, F is convex.

Block coordinate descent (BCD) of the Gauss-Seidel type minimizes F cyclically over each of the individual blocks x_i while fixing the other blocks to their latest updated values [13], [21].

$$\boldsymbol{x}_i^{k+1} = \arg\min_{\boldsymbol{x}_i} \; f_i^{k+1}(\boldsymbol{x}_i^{k+1}) + r_i(\boldsymbol{x}_i) \tag{7}$$

The general block coordinate descent method for non-convex problems, however, has no guarantees of convergence (either to a local minimum or otherwise) and may cycle infinitely. In order to address this, we use a proximal update when updating and solving each block

$$\boldsymbol{x}_i^{k+1} = \arg\min_{\boldsymbol{x}_i} \; f_i^{k+1}(\boldsymbol{x}_i^{k+1}) + \frac{L_i^k}{2}||\boldsymbol{x}_i^{k+1} - \boldsymbol{x}_i^{k}||^2 + r_i(\boldsymbol{x}_i) \tag{8}$$

where L_i^k is a non-zero regularization parameter and $||\cdot||$ is the ℓ_2-norm. The proximal parameter, L_i^k, is in practice used to regularize the current solution to ensure we do not stray too far from the previous solution (i.e. introduces damping) and may change at every iterate. Using this method, we are guaranteed to converge to a feasible solution [15].

An important note is that that this does not necessarily mean we are guaranteed to converge to a local minimum of the original optimization problem F. Rather, we are only guaranteed a feasible trajectory (i.e. all constraints are satisfied). We will see in the following subsections how an appropriate choice of blocks and use of the proximal update parameter allows us to converge to reasonable solutions for the centroidal dynamics optimization.

A. Block Coordinate Descent for Centroidal Dynamics

In order to solve the centroidal optimization problem, Eq. 4, using the block coordinate descent method we first apply a change of variables similar to one introduced in [10]. Specifically, we apply a change of variables to the cross products between the force, $\boldsymbol{f}$ and contact location $(\boldsymbol{p}_e - \boldsymbol{r})$ as well as the $\boldsymbol{f}$ and rotated ZMP in Eqs. (4c) and (4d) to combine these into one variable, $\boldsymbol{\ell}$:

$$\begin{aligned} \boldsymbol{\kappa}_{e,t} &= (\boldsymbol{p}_{e,t} - \boldsymbol{r_t} + \boldsymbol{R}_{e,t}^{x,y}\boldsymbol{z}_{e,t}) \times \boldsymbol{f}_{e,t} + \boldsymbol{\tau}_{e,t} \\ &= \boldsymbol{\ell} \times \boldsymbol{f}_{e,t} + \boldsymbol{\tau}_{e,t} \\ &= \begin{bmatrix} 0 & -\boldsymbol{\ell}_{e,t}^{z} & \boldsymbol{\ell}_{e,t}^{y} \\ \boldsymbol{\ell}_{e,t}^{z} & 0 & -\boldsymbol{\ell}_{e,t}^{x} \\ -\boldsymbol{\ell}_{e,t}^{y} & \boldsymbol{\ell}_{e,t}^{x} & 0 \end{bmatrix} \begin{bmatrix} \boldsymbol{f}_{e,t}^{x} \\ \boldsymbol{f}_{e,t}^{y} \\ \boldsymbol{f}_{e,t}^{y} \end{bmatrix} + \boldsymbol{\tau}_{e,t} \end{aligned} \tag{9}$$

Using this change of variables, we see that the non-convexity is in fact biconvex. Specifically, for a fixed set of $\boldsymbol{\ell}$, the problem is convex with respect to our forces, $\mathbf{f}$, and vice-versa. This new change of variables then suggests the use of two-block minimizations.

Our algorithm is outlined as follows: We first fix $\boldsymbol{\ell_i}$, and solve for forces $\boldsymbol{f_i}$ in one quadratic program which we will call the Force-QP, $\boldsymbol{\zeta}$. We then use the forces to solve for $\boldsymbol{\ell}$ in a second QP which we call the Contact-QP, $\boldsymbol{\nu}$. We iterate on this process until a consensus is found or a maximum number of iterations is reached after which one final Force-QP is run to generate fully dynamically consistent profiles (i.e. forces for the appropriate CoM, momentum, and end-effector location profiles). Algorithm 1 provides an outline of the block coordinate descent algorithm.

B. Force Quadratic Program

The Force-QP solves the full centroidal dynamics problem, Eq. 4, for a fixed $\boldsymbol{\ell}$. During each iteration, k, we increase the parameter L_i by a factor α. Due to the use of quadratic costs, the parameter L_i in practice is used to regularize the solution from the previous solution. The Force-QP can be stated as follows

$$\min_{\boldsymbol{h}^k, \boldsymbol{f}_e^k, \boldsymbol{\tau}_e^k, \boldsymbol{z}^k} \quad \sum_{t=0}^{N} [\Psi_t(\boldsymbol{h}^k, \boldsymbol{z}_e^k, \boldsymbol{f}_e^k, \boldsymbol{\tau}_e^k) + \phi_t(\boldsymbol{h}_t^k - \boldsymbol{h}_t^{kin}) + L^{k,\zeta}(\boldsymbol{h}_t^{k,\zeta} - \boldsymbol{h}_t^{k-1,\nu})] \tag{10a}$$

$$\text{s.t.} \quad \boldsymbol{h}_t^k = \begin{bmatrix} \boldsymbol{r}_t^k \\ \boldsymbol{l}_t^k \\ \boldsymbol{k}_t^k \end{bmatrix} = \begin{bmatrix} \boldsymbol{r}_{t-1}^k + \frac{1}{m}\boldsymbol{l}_t^k \Delta t \\ \boldsymbol{l}_{t-1}^k + m\boldsymbol{g}\Delta t + \sum_e \boldsymbol{f}_{e,t}^k \Delta t \\ \boldsymbol{k}_{t-1}^k + \sum_e \boldsymbol{\kappa}_{e,t}^k \Delta t \end{bmatrix} \tag{10b}$$

$$\boldsymbol{\kappa}_{e,t}^k = (\boldsymbol{\ell}^{k-1,\nu}) \times \boldsymbol{f}_{e,t}^k + \boldsymbol{\tau}_{e,t} \tag{10c}$$

$$|\boldsymbol{f}_{e,t}^x| \leq \boldsymbol{R}_{e,t}\boldsymbol{f}_{e,t}^z, \; |\boldsymbol{f}_{e,t}^y| \leq \boldsymbol{R}_{e,t}\boldsymbol{f}_{e,t}^z, \; \boldsymbol{R}_{e,t}\boldsymbol{f}_{e,t}^z \geq 0 \tag{10d}$$

$$\| \boldsymbol{p}_{e,t}^{k-1,\nu} - \boldsymbol{r}_t^{k,\zeta} \| \leq \mathcal{L}^{max} \tag{10e}$$

We note that the optimization problem does not regularize the momentum dynamics from the previous Force-QP but rather from the previous Contact-QP (in the case of the first iteration when no Contact-QP has been run, we do not regularize the center of mass location at all).

C. Contact Quadratic Program

The Contact-QP is then used to solve for the length of the CoM wrench, $\boldsymbol{\ell}$, given the forces solved in the previous QP. Rather than optimizing over $\boldsymbol{\ell}$ directly, we need to remember the physics of the problem we are trying to solve; specifically that $\boldsymbol{\ell}_t = \boldsymbol{p}_{e,t} - \boldsymbol{r}_t + \boldsymbol{R}_{e,t}^{x,y}\boldsymbol{z}_{e,t}$. In order to give us finer control of individually tracking the end-effector location, $\boldsymbol{p}_{e,t}$, and center of mass, $\boldsymbol{r}_t$, we separate these individually in our QP. This gives us the following optimization problem:

$$\min_{\boldsymbol{r}^k, \boldsymbol{p}_e^k, \boldsymbol{l}^k, \boldsymbol{z}_e} \quad \sum_{t=0}^{N} [\Psi_t(\boldsymbol{r}^k, \boldsymbol{p}_e^k, \boldsymbol{l}^k, \boldsymbol{z}^k) + L^{k,\nu}(\boldsymbol{h}_t^{k,\nu} - \boldsymbol{h}_t^{k-1,\zeta}, \boldsymbol{p}_e^{k,\nu} - \boldsymbol{p}_e^{k-1,\nu})] \tag{11a}$$

$$\text{s.t.} \quad \begin{bmatrix} \boldsymbol{r}_t^k \\ \boldsymbol{k}_t^k \end{bmatrix} = \begin{bmatrix} \boldsymbol{r}_{t-1}^k + \frac{1}{m}\boldsymbol{l}_t^{k,\zeta}\Delta t \\ \boldsymbol{k}_{t-1}^k + \sum_e \boldsymbol{\kappa}_{e,t}^k \Delta t \end{bmatrix} \tag{11b}$$

$$\boldsymbol{\kappa}_{e,t}^k = \boldsymbol{f}_{e,t}^k \times (\boldsymbol{r}_t^{k,\nu} - \boldsymbol{p}_e^{k,\nu}) + \boldsymbol{\gamma}_{e,t}^k \tag{11c}$$

$$\boldsymbol{\gamma}_{e,t}^k = -\boldsymbol{f}_{e,t}^k \times (\boldsymbol{R}_{e,t}^{x,y}\boldsymbol{z}_{e,t}^k) \tag{11d}$$

$$\boldsymbol{z}_{e,t}^{x,y} \in [{}^{min}\boldsymbol{z}^{x,y}, {}^{max}\boldsymbol{z}^{x,y}] \tag{11e}$$

$$\boldsymbol{p}_{e,t}^{\nu} \in \mathcal{U}(\mathcal{S}) \tag{11f}$$

$$\| \boldsymbol{p}_{e,t}^{k,\nu} - \boldsymbol{r}_t^{k,\nu} \| \leq \mathcal{L}^{max} \tag{11g}$$

where we rearrange Eqs. (11c), (11d) using the cross product identity

$$\boldsymbol{a} \times \boldsymbol{b} = -\boldsymbol{b} \times \boldsymbol{a} \tag{12}$$

Once again, we use L_i to regularize the solution from the previous QP. However, because the momentum dynamics $\boldsymbol{h_t}$ appears in the Force-QP itself, we instead regularize the trajectories with the values from the previous Force-QP rather than the previous Contact-QP. As in the case of the Force-QP, we increase the value of L_i during each iteration. We would also like to note the lack of state transition constraints for the linear momentum, $\boldsymbol{l}_i$. If we were to add these constraints into our formulation, this would prevent the center of mass from being able to alter and track the angular momentum and would only rely on the contact location $\boldsymbol{p}_e$ and the ZMP, $\boldsymbol{z}$. By relaxing these constraints, we instead regularize the linear momentum from the previous Force-QP which allows our optimization the freedom to use the center of mass to reduce momentum. We once again note, that since we use one final Force-QP before finishing the alternating process, CoM trajectories that may be invalid due to the removal of the linear momentum constraints are pushed within the constraint set to satisfy the dynamic criteria.

D. Convergence Criteria

The convergence of the algorithm is dependent on the weights chosen, in particular, the weighting factor, L_i and the scaling factor between iterations, α. In practice, we find that a good stopping point is when the angular momentum profiles from one iteration to the next fall below some consensus threshold, ϵ_f or after K iterations. ϵ_f is defined as follows:

$$\epsilon_f = \frac{||\boldsymbol{\ell}^k - \boldsymbol{\ell}^{k-1}||^2}{N} \tag{13}$$

where N is the horizon of our optimal control problem.

Algorithm 1: Block Coordinate Descent for Biconvex Optimization

Initialize optimization variables: $\boldsymbol{f_0}$, $\boldsymbol{h_0}$, $\boldsymbol{\ell_0}$
set k = 0
while $k <$ *maximum iterations* **do**
  $\boldsymbol{f^{k+1,\zeta}}, \boldsymbol{h^{k+1,\zeta}} = \mathrm{QP}_{\mathrm{Force}}(\boldsymbol{h^{k,\nu}}, \boldsymbol{\ell^{k,\nu}})$
  $\boldsymbol{L^{k+1,\zeta}} = \alpha \boldsymbol{L^{k,\zeta}}$
  $\boldsymbol{\ell^{k+1,\nu}} = \mathrm{QP}_{\mathrm{Contact}}(\boldsymbol{h^{k+1,\zeta}}, \boldsymbol{f^{k+1,\zeta}})$
  $\boldsymbol{L^{k+1,\nu}} = \alpha \boldsymbol{L^{k,\nu}}$
  if $||\ell^k - \ell^{k-1}||^2/N \leq \epsilon_f$ **then**
    terminate
$\boldsymbol{f^{k+1,\zeta}}, \boldsymbol{h^{k+1,\zeta}} = \mathrm{QP}_{\mathrm{Force}}(\boldsymbol{h^{k,\nu}}, \boldsymbol{\ell^{k,\nu}})$

IV. Experimental Setup

A. Multi-contact control pipeline and platform

In order to verify and test the profiles generated by the dynamics optimization, we utilize the open-source kinodynamic trajectory optimization package [22]. Contact sequences can either be computed using the MIQCQP [10] or set manually and are then given to our block coordinate descent framework. After our method finds a dynamically feasible motion, these are then sent to the kinematics optimizer. The resulting output of the framework is a whole-body motion which is then tracked by the whole-body controller (WBC) outlined in [23]. The controller uses feedback on the centroidal momentum combined with a desired task-space impedance plus a joint space PD controller to solve a QP for end-effector forces. These forces are then mapped to

actuator torques using the jacobian transpose and executed and tracked as torque commands.

The motions were generated for a 12 degree of freedom quadruped, Solo12, which is simulated using the PyBullet simulation software [24]. Due to the use of a quadruped with point contacts, we eliminate the end-effector torque constraints from Eq. 10c as well as the ZMP constraints in Eq. 11e and Eq. 11d when generating motions.

B. Solver details

The proposed method was implemented in Python using the open source QP solver OSQP [25]. All experiments were run with the solver settings shown in Table II located in the Appendix. All computations were performed using a single thread with an Intel Core i7-9850H CPU @ 4.6 GHz and 16 GB 2666 MHz RAM.

V. Results

We tested our algorithm in several different scenarios from flat ground motion generation to multi-contact navigation on uneven terrain as well as more dynamic motions such as jumping and bounding. We would like to note that most trajectories were optimized over relatively long horizons (around 5 s) with several contact switches. Despite this, we were able to successfully track these open-loop using only the WBC without re-optimization of the trajectory (i.e. without model predictive control). This suggests that the computed trajectories are of good quality. For each motion, we either give a convex region for the contact optimization or fix the contact location and only allow the center of mass to reduce the angular momentum.

A. Generating various locomotion behaviors

We first generated and tracked motions for different types of gaits such as walking, trotting, and bounding. For simple motions, such as walking and trotting, the algorithm converges within two iterations (i.e. after k = 2). For bounding, which requires finer control of angular momentum, a third iteration was required before the algorithm converged.

Bounding in particular tests our algorithm's ability to generate and track trajectories that incur a high amount of angular momentum. Without explicitly re-optimizing angular momentum via model predictive control or receding horizon control, tracking of these types of trajectories is often difficult. In Fig. 3b we show that despite this, we were able to generate and track such a motion for an 8 second horizon using only the WBC.

We further tested the ability of our algorithm to generate arbitrary motions for navigating uneven terrain. We tested our algorithm on multiple staircases with the height of each staircase ranging in values from 12.5% to 35% of the center of mass height of the robot. For navigating both up and down stairs, our algorithm converged within two iterations which took 0.827 s for a horizon of $N = 300$ which coincided with a 3 second horizon.

We also computed motions for non-coplanar terrain such as angled stepping stones as in Fig. 1. We noticed that as the contact angles became more aggressive, our algorithm took longer to converge, often requiring three major iterations (i.e. iterations of Algorithm 1) with total solve times on average 2.398 s for $N = 300$, which is also due to the individual QPs requiring more iterations to converge. Finally, we also tested the ability of our algorithm to generate highly dynamic motions with flight phases. We were able to generate and track trajectories for jumping in place as well as jumping forward and rotationally (around the z-axis). We note the importance of regulating angular momentum in directional jumps. In particular, although a trajectory may be feasible, regulation of angular momentum plays a large role such that the robot does not flip in the air. We found that even during fairly aggressive jumps, we were able to generate motions that generated minimal angular momentum and could thus be successfully tracked.

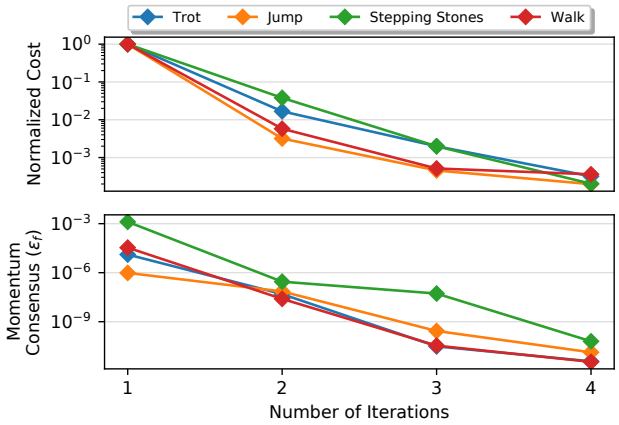


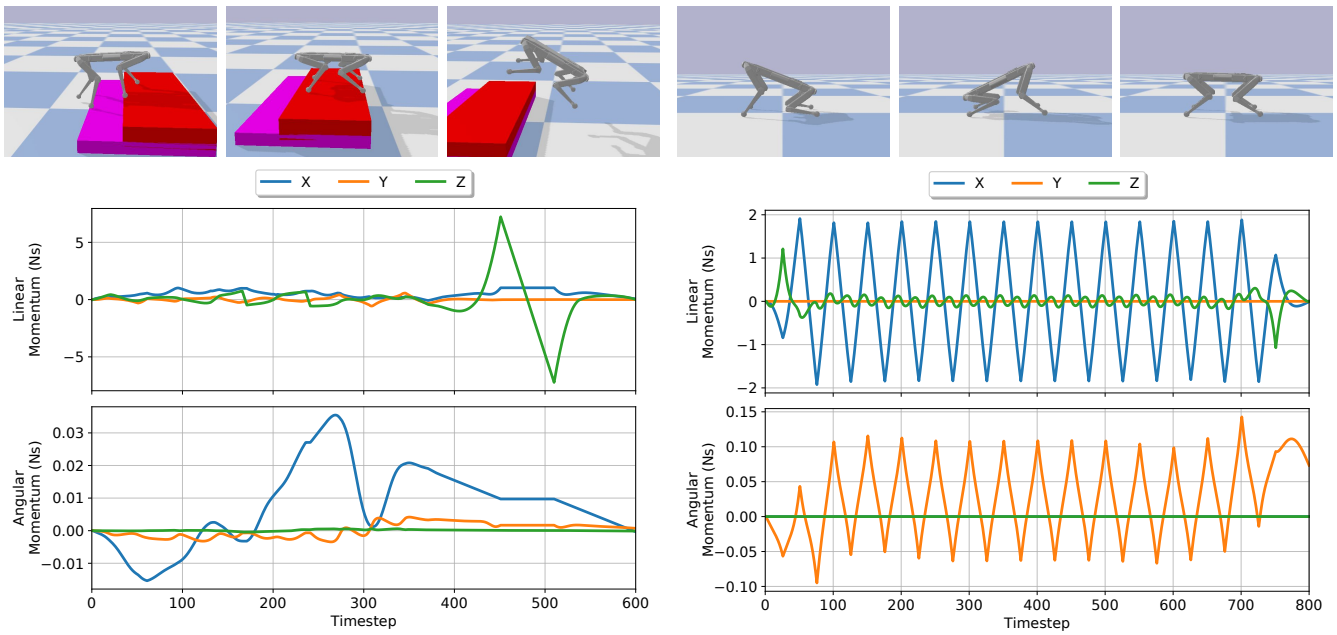
Fig. 2. In the top graph, we show the normalized cost per iteration for a variety of motions. We can see that our algorithm tends to converge after the 3rd major iteration, although every iteration after the first is technically feasible. On the bottom, we graph our convergence criteria ϵ_f. By the end of the second iteration, we find that generally our momentum profiles $\boldsymbol{\ell}$ find consensus to a high enough tolerance that these motions can be tracked by our WBC.

Fig. 3a shows the results of a non-gaited multi-contact motion that requires navigating uneven terrain with a jump off the top step. Despite the multiple contact switches and dynamic motion, our WBC was able to track such a motion successfully. All the resulting motions simulated in pyBullet are shown in the accompanying video.

B. Evaluation of convergence

We plot the cost per iteration for 50 motions from the categories above (various legged gaits, uneven terrain, and jumping) with a range of L_i from $[100 : 1,000,000]$ and $\alpha = 100$ in Fig. 2. As we can see, both our cost and momentum profiles tend to converge and find a lower bound after three to four iterations.

Figure 2 also shows how our momentum consensus parameter, $\boldsymbol{\epsilon}_f$ changes per iteration. For the problems of interest, we experimentally found that a good value for our convergence parameter ϵ_f was 10^{-7}. Using the above values for L_i and α, for every motion we generated, we were able to converge within a maximum of three iterations.



(a) Momentum profiles and tracking of non-gaited motion

(b) Momentum profiles and tracking of a bounding gait

Fig. 3. In this figure we plot the resulting profiles for two different multi-contact scenarios with multiple contact switches as well simulation screenshots showing the successful tracking of these motions. Fig. 3a shows the profiles for a motion that included navigating over uneven terrain with a jump onto the ground which showcases the ability of our algorithm to compute trajectories with arbitrary, non-gaited contact switches together with highly dynamic motions. Fig. 3b, shows the profiles for a bounding motion computed using our method. We set the contact locations to switch between the front and hind legs every $0.25\,\mathrm{s}$. As we can see in both scenarios, we are able to generate high enough quality motions such that despite the numerous contact switches, we can successfully track each motion without the explicit re-optimization of the trajectories.

C. Speed and scaling

We plot the total solution time of the algorithm in Fig. 4. The solution times plotted are the total solve times of each QP and do not include the time to modify the appropriate QP from their previous solutions as these are trivial. We also do not include the initial setup time of the solver for each problem as this can be set beforehand with the predefined, sparse structure before running the actual optimization and, thus, only has to be run once for each QP. In general we found that for simple motions such as walking and trotting, our algorithm tended to converge within 2 overall iterations which on average took $0.7972\,\mathrm{s}$ for $N = 300$. For navigation of non-coplanar terrain we saw that an extra iteration was often required; we believe this was due to the the tightness of the constraint set of our trajectories (e.g. due to the rotated friction cones). For $N = 300$, navigation of non-coplanar terrain averaged $2.647\,\mathrm{s}$.

The algorithm spends the majority of the time of each iteration on the Force-QP. For motions with a horizon of $N = 300$, the algorithm spends on average $97.8\,\%$ of the total solution time on the Force-QP and $2.2\,\%$ on the Contact-QP despite their similar size. This is due to the fewer number of constraints in the Contact-QP.

Finally, we explore the scalability of our method for different lengths of time horizons N and show the result in Fig. 4. We observed an approximately linear-time complexity in solve time with respect to horizon length. This is due to the banded, sparse nature of our problem which can be efficiently exploited by sparse QP solvers.

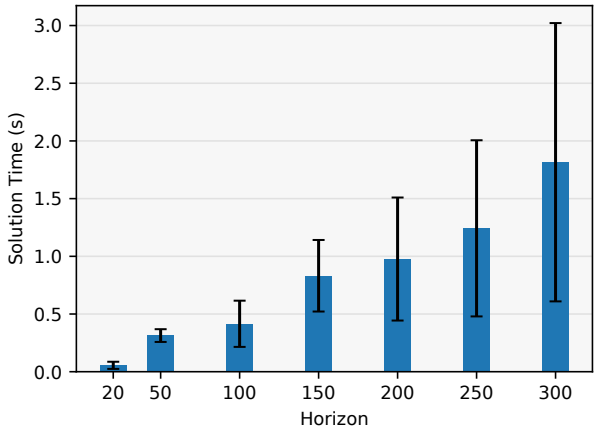


Fig. 4. The total solution time for our algorithm for a variety of 50 motions. We note a roughly linear increase in time with horizon for the problem sizes we are interested in due to the sparsity of our problem. Our solution time contains only the solve time of each QP and does not include the setup time of each QP.

D. Comparison to other solution methods

In this section, we compare the solutions found between our block coordinate descent method proposed above, the soft constraint sequential convex relaxation (SCR) method outlined in [10], and the primal-dual interior point method used in IPOPT which is often used in locomotion research [26], [27]. For the following comparisons, we configured

Solver Method	Average Normalized Cost	Solve Time ($N = 300$)	Change In Solve Time	Solver Success Rate
Block Coordinate Descent	**1.00**	$\mathbf{1.77 \pm 1.23}$s	–	**100**%
Sequential Convex Relaxations	1.08	7.34 ± 3.34s	4.15x slower	**100**%
Interior Point Method (IPOPT)	1.34	8.12 ± 2.32s	4.59x slower	92%

TABLE I

COMPARISON OF METHODS FOR SOLVING THE CENTROIDAL DYNAMICS OPTIMIZATION PROBLEM.

IPOPT with the sparse symmetric linear solver MA57. Although [10] provided multiple heuristics, we chose to use the soft constraints as in our experience these performed better than the trust region method. We provide this comparison as a means to quantify the solutions found by our method to against approaches where the solutions are associated with guaranteed local minima and not as a means to compare the quality of the solutions.

We first compare the final cost obtained for a variety of different multi-contact locomotion problems including flat ground motions, uneven terrain, and jumping motions. We note that the final cost calculated using the block coordinate descent method uses the original cost function, without the proximal term. Specifically, we use the solution generated by Algorithm 1, and then plug this back into the original cost function, Eq. 4a, to evaluate the cost. We found that our method generally finds lower costs than both alternative approaches despite not converging to a local minimum.

Next, we compare the solution time of each method. Once again, we evaluate our speed for a variety of different motions. Our method readily outperformed both and is on average more than four times faster than than both the SCR method and IPOPT for trajectories with horizons of $N = 300$.

The results of both comparisons can be found in Table I. We note that the discrepancy in solve times between our reported values and those reported in [10] are likely due to the types of motions we generate as well as the default tolerances and settings of the open source implementation. We would also like to point out that for the problem sizes we are interested in, the authors of [10] noted a roughly linear time complexity in the problem horizon solve time due to the sparse nature of our problem (similar to our BCD implementation). Due to the use of a sparse solver in our IPOPT implementation we expect to see a similar trend.

Of particular note is the ability of our solver to find motions that require finer tracking of angular momentum such as bounding. Unlike the SCR method, our method was able to create trajectories for such motions that could be tracked by our WBC reasonably well in simulation. We plot a comparison of the results of tracking an open-loop bounding trajectory in Fig. 5. From a numerical optimization standpoint, we found that, compared to the SCR method, BCD solutions were often satisfied to much higher numerical tolerances, especially for dynamic motions such as jumping and bounding. Finally, we note that IPOPT failed to find solutions for $8\,\%$ of the given multi-contact motions whereas both the BCD and SCR methods were able to find solutions for all the given scenarios.

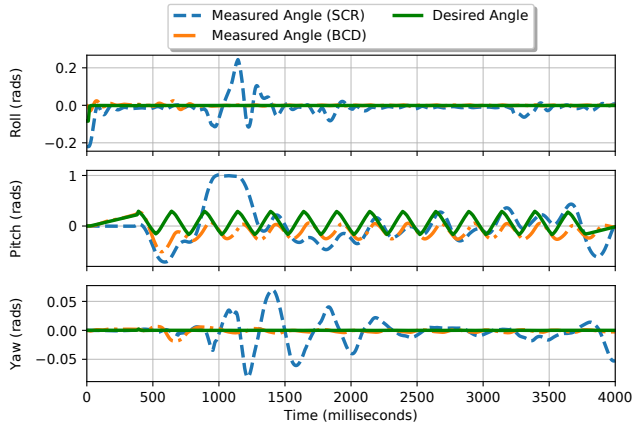


Fig. 5. The tracking ability of a planned bounding gait between the SCR method and our BCD method. In order to generate this motion, we gave each method a reference angular momentum corresponding to a maximum pitch of $\pm 15°$. The BCD was able to create a motion that could be tracked by our WBC (yellow), however, the SCR method was unable to do so (blue). Specifically, we note the large error starting at $1000\,\mathrm{ms}$. The RMSE (Root Mean Square Error) of the SCR pitch tracking is $5.84 \cdot 10^{-3}$ whereas the RMSE for the BCD approach is $3.33 \cdot 10^{-3}$ ($43\,\%$ lower). We note that the desired angle was converted into a desired angular momentum using the matrix logarithm.

VI. DISCUSSION AND CONCLUSION

The experiments above indicate that despite the lack of guarantees of convergence to a critical point, our algorithm is still able to generate high quality multi-contact profiles that can be tracked by our robot. Although guarantees of global convergence are nice to have, in practice, the robotics community uses tools such as trajectory optimization as motion generators and the cost functions used are often arbitrary.

Due to the structured, sparse nature of our problem, we also see a significant speedup that can be exploited by off-the-shelf quadratic solvers. OSQP, for example, employs factorization caching which is utilized when re-solving the linear system of equations as well as polishing, a feature to predict the number of active constraints. We believe that as quadratic programming solvers continue to mature, we may be able to further exploit these types of features to further speed up our solve times.

We presented a novel method for solving the non-convex centroidal dynamics optimization problem. Rather than relying on off-the-shelf nonlinear solvers which have no convergence guarantees or complicated relaxation methods, we showed that by applying BCD and using a standard quadratic programming solver, we are able to efficiently find reasonable solutions for the non-convex problem. Compared to the state of the art, we are also able to solve the problem multiple times faster and can often generate and track trajectories that require finer tracking of angular momentum.

We believe our algorithm is well-suited for model predictive control or variable horizon control. In particular, we can exploit our algorithm's capability of finding feasible trajectories at every iteration to reduce the number of cycles required even further. We are also able to warm-start solutions from previous solutions which may further decrease our solve time. Due to the use of interior point methods, this ability cannot be exploited by the relaxation methods proposed in [10] as well as frameworks built upon IPOPT. Additionally, these properties also enable us to efficiently combine our method with pre-computed libraries or data-driven techniques [28], [29], [30]. Finally, we believe BCD is a versatile approach for trajectory optimization in robotics due to its simplicity of implementation. In the future, we plan to validate our trajectories using hardware experiments and extend our method to real-time control.

APPENDIX

OSQP Solver Settings	
Absolute Tolerance, ϵ_{abs}	$1e-7$
Relative Tolerance, ϵ_{rel}	$1e-7$
Primal Infeasibility Tolerance, $\epsilon_{prim\,inf}$	$1e-6$
Dual Infeasibility Tolerance, $\epsilon_{dual\,inf}$	$1e-6$
Polish	True
Scaled Termination	True
Adaptive Rho	True
Check Termination	50

TABLE II
OSQP SOLVER SETTINGS FOR FORCE AND CONTACT QP

ACKNOWLEDGMENTS

The authors would like to thank Julian Viereck for his assistance with the kino-dynamic planner.

REFERENCES

[1] R. Full and D. Koditschek, "Templates and anchors: neuromechanical hypotheses of legged locomotion on land," *Journal of Experimental Biology*, vol. 202, no. 23, pp. 3325–3332, 1999.

[2] S. Kajita, F. Kanehiro, K. Kaneko, K. Fujiwara, K. Harada, K. Yokoi, and H. Hirukawa, "Biped walking pattern generation by using preview control of zero-moment point," in *Proc. IEEE Int. Conf. Rob. Autom. (ICRA)*, vol. 2, 2003, pp. 1620–1626 vol.2.

[3] A. Herdt, H. Diedam, P.-B. Wieber, D. Dimitrov, K. Mombaur, and M. Diehl, "Online walking motion generation with automatic footstep placement," *Advanced Robotics*, vol. 24, no. 5-6, pp. 719–737, 2010.

[4] J. Englsberger, C. Ott, and A. Albu-Schäffer, "Three-dimensional bipedal walking control based on divergent component of motion," *IEEE Transactions on Robotics*, vol. 31, no. 2, pp. 355–368, 2015.

[5] M. A. Hopkins, D. W. Hong, and A. Leonessa, "Humanoid locomotion on uneven terrain using the time-varying divergent component of motion," in *Proc. IEEE-RAS Int. Conf. Hum. Rob. (Humanoids)*, 2014, pp. 266–272.

[6] D. Orin, A. Goswami, and S.-H. Lee, "Centroidal dynamics of a humanoid robot," *Autonomous Robots*, vol. 35, 10 2013.

[7] H. Dai, A. Valenzuela, and R. Tedrake, "Whole-body motion planning with centroidal dynamics and full kinematics," in *Proc. IEEE-RAS Int. Conf. Hum. Rob. (Humanoids)*, 2014, pp. 295–302.

[8] A. Herzog, S. Schaal, and L. Righetti, "Structured contact force optimization for kino-dynamic motion generation," in *Proc. IEEE/RSJ Int. Conf. Intell. Rob. Sys. (IROS)*, 10 2016, pp. 2703–2710.

[9] J. Carpentier and N. Mansard, "Multicontact locomotion of legged robots," *IEEE Transactions on Robotics*, vol. 34, no. 6, pp. 1441–1460, 2018.

[10] B. Ponton, M. Khadiv, A. Meduri, and L. Righetti, "Efficient Multi-Contact Pattern Generation with Sequential Convex Approximations of the Centroidal Dynamics," *IEEE Transactions on Robotics*, pp. 1–19, 2021.

[11] H. Dai and R. Tedrake, "Planning robust walking motion on uneven terrain via convex optimization," in *Proc. IEEE-RAS Int. Conf. Hum. Rob. (Humanoids)*, 2016, pp. 579–586.

[12] A. Domahidi, E. Chu, and S. Boyd, "Ecos: An socp solver for embedded systems," in *European Control Conference (ECC)*, 2013, pp. 3071–3076.

[13] X. Shen, S. Diamond, M. Udell, Y. Gu, and S. Boyd, "Disciplined multi-convex programming," in *2017 29th Chinese Control And Decision Conference (CCDC)*, 2017, pp. 895–900.

[14] A. Wächter and L. Biegler, "On the implementation of an interior-point filter line-search algorithm for large-scale nonlinear programming," *Mathematical programming*, vol. 106, pp. 25–57, 03 2006.

[15] Y. Xu and W. Yin, "A block coordinate descent method for regularized multiconvex optimization with applications to nonnegative tensor factorization and completion," *SIAM Journal on Imaging Sciences*, vol. 6, no. 3, pp. 1758–1789, 2013.

[16] P.-B. Wieber, "Holonomy and nonholonomy in the dynamics of articulated motion," in *Fast motions in biomechanics and robotics*. Springer, 2006, pp. 411–425.

[17] A. Herzog, N. Rotella, S. Mason, F. Grimminger, S. Schaal, and L. Righetti, "Momentum control with hierarchical inverse dynamics on a torque-controlled humanoid," *Autonomous Robots*, vol. 40, 10 2014.

[18] R. Deits and R. Tedrake, "Footstep planning on uneven terrain with mixed-integer convex optimization," in *Proc. IEEE-RAS Int. Conf. Hum. Rob. (Humanoids)*, 2014, pp. 279–286.

[19] S. Tonneau, D. Song, P. Fernbach, N. Mansard, M. Taïx, and A. Del Prete, "SL1M: Sparse L1-norm Minimization for contact planning on uneven terrain," in *Proc. IEEE Int. Conf. Rob. Autom. (ICRA)*, 05 2020, pp. 6604–6610.

[20] Y.-C. Lin and D. Berenson, "Humanoid navigation planning in large unstructured environments using traversability-based segmentation," in *2018 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE, 2018, pp. 7375–7382.

[21] G. H. Golub and C. F. Van Loan, *Matrix Computations (3rd Ed.)*. USA: Johns Hopkins University Press, 1996.

[22] B. Ponton, "Kino-dynamic trajectory optimization for multi-ped robots," https://github.com/machines-in-motion/kino_dynamic_opt, accessed: 2020-11-30.

[23] F. Grimminger, A. Meduri, M. Khadiv, J. Viereck, M. Wüthrich, M. Naveau, V. Berenz, S. Heim, F. Widmaier, T. Flayols, J. Fiene, A. Badri-Spröwitz, and L. Righetti, "An open torque-controlled modular robot architecture for legged locomotion research," *IEEE Robotics and Automation Letters*, vol. 5, no. 2, pp. 3650–3657, 2020.

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

[25] B. Stellato, G. Banjac, P. Goulart, A. Bemporad, and S. Boyd, "OSQP: an operator splitting solver for quadratic programs," *Mathematical Programming Computation*, vol. 12, no. 4, pp. 637–672, 2020.

[26] A. W. Winkler, C. D. Bellicoso, M. Hutter, and J. Buchli, "Gait and trajectory optimization for legged systems through phase-based end-effector parameterization," *IEEE Robotics and Automation Letters*, vol. 3, no. 3, pp. 1560–1567, 2018.

[27] G. Bledt and S. Kim, "Implementing regularized predictive control for simultaneous real-time footstep and ground reaction force optimization," in *Proc. IEEE/RSJ Int. Conf. Intell. Rob. Sys. (IROS)*, 2019, pp. 6316–6323.

[28] J. Viereck and L. Righetti, "Learning a centroidal motion planner for legged locomotion," 2020.

[29] M. Stolle and C. G. Atkeson, "Policies based on trajectory libraries," in *Proc. IEEE Int. Conf. Rob. Autom. (ICRA)*, 2006, pp. 3344–3349.

[30] W. Merkt, V. Ivan, and S. Vijayakumar, "Leveraging precomputation with problem encoding for warm-starting trajectory optimization in complex environments," in *Proc. IEEE/RSJ Int. Conf. Intell. Rob. Sys. (IROS)*, Oct 2018, pp. 5877–5884.