BiConMP: A Nonlinear Model Predictive Control Framework for Whole Body Motion Planning

Avadesh Meduri*[1], Paarth Shah*[3], Julian Viereck[1], Majid Khadiv[2], Ioannis Havoutis[3] and Ludovic Righetti[1]

***Abstract*— Online planning of whole-body motions for legged robots is challenging due to the inherent nonlinearity in the robot dynamics. In this work, we propose a nonlinear MPC framework, the BiConMP which can generate whole body trajectories online by efficiently exploiting the structure of the robot dynamics. BiConMP is used to generate various cyclic gaits on a real quadruped robot and its performance is evaluated on different terrain, countering unforeseen pushes and transitioning online between different gaits. Further, the ability of BiConMP to generate non-trivial acyclic whole-body dynamic motions on the robot is presented. The same approach is also used to generate various dynamic motions in MPC on a humanoid robot (Talos) and another quadruped robot (AnYmal) in simulation. Finally, an extensive empirical analysis on the effects of planning horizon and frequency on the nonlinear MPC framework is reported and discussed.**

I. INTRODUCTION

Legged robots can autonomously navigate and operate in environments built for humans. The efficacy of such robots depends largely on how efficiently they can move around, adapt to changes in their surroundings and recover from unforeseen disturbances. These decisions are usually made by trajectory optimization algorithms which compute optimal robot movements and contact forces and contact planners which decide which end-effector should make contact with the environment. Consequently, the trajectory optimization algorithm needs to be general enough that it can generate any behavior that is needed for the robot to achieve the desired task. At the same time, these algorithms should run as fast as possible such that they adapt online to any changes in the environment. In this work, we propose a general approach to generate whole-body trajectories that is sufficiently fast to be used in a model predictive control (MPC) fashion.

Initially, algorithms based on simplified models such as the linear inverted pendulum model (LIPM) [1], [2] were developed to generate trajectories online for humanoid robots. These algorithms make use of a predefined footstep sequence provided by the user to generate a feasible center of mass (CoM) trajectory. Since the LIPM leads to an optimization problem with quadratic costs and linear constraints, the problem can be solved quickly using a quadratic program (QP) [3]. While a further extension of these algorithms enabled adaptation of the step location and timing [4], [5], [6], they are only capable of generating walking motions for flat grounds with co-planar contacts.

On the other hand, frameworks that can plan contacts and optimal motions for complex scenarios have also been developed. In [7], [8], [9], the full-body motion and contact selection problems are formulated as single nonlinear optimization problems. In [10], a more efficient phase-based formulation of contact planning is proposed. Furthermore, [11] makes use of differential dynamic programming (DDP) to solve the motion optimization problem through contact. While these approaches can in principle find complex contact sequences, they tend to be computationally too expensive to be used in real-time. Although [12] showed nonlinear model predictive control on a quadruped using a Gauss-Newton multiple shooting variation of DDP and a relaxed spring damper contact model, results were limited to motions with low angular momentum such as trotting and jumping in place.

Classically, the trajectory optimization problem has been split into two different sub-problems: contact planning and motion optimization. This decomposition reduces the complexity of the overall problem which allows them to be tractable. The main idea is to first generate a contact sequence given the terrain around the robot. The contact sequence is then provided to the motion planner to generate a feasible trajectory for the robot. The contact planning sub-problem can be solved using a variety of approaches such as mixed integer optimization [13], [14], L1-loss based optimization [15], graph search [16] and sampling-based approach [17]. For the motion optimization sub-problem, the nonlinear dynamics can be split into two components, the actuated and unactuated dynamics (centroidal dynamics) [18]. One of the interesting approaches to generate whole body motions quickly is to use the centroidal dynamics and full kinematics of the robot in one optimization problem [19]. Further, a feasible whole-body motion can be generated more efficiently by iteratively optimizing for the centroidal dynamics and whole-body kinematics problems [20]. Despite splitting the trajectory optimization problem into two parts, each individual sub-problem remains nonlinear and challenging to solve in real time.

In order to further reduce computation times, several relaxations have been proposed to the centroidal dynamics formulation. In [21], sequential convex relaxations are used

*Both the authors contributed equally.

This work was supported by New York University, the European Union's Horizon 2020 research and innovation program (grant agreement 780684) and the National Science Foundation (grants 1825993 and 1925079). Paarth Shah was supported by an AWS Lighthouse Scholarship.

[1]Tandon School of Engineering, New York University (NYU), USA. am9789@nyu.edu, ludovic.righetti@nyu.edu

[2]Max-Planck Institute for Intelligent Systems, Tuebingen, Germany. majid.khadiv@tuebingen.mpg.de

[3]Oxford Robotics Institute, University of Oxford, England. paarth@oxfordrobotics.institute, ioannis@oxfordrobotics.institute

and each relaxation is solved using a second-order cone program [22]. Even though this approach can be used to quickly optimize a variety of motions, the reduction in compute times are not yet sufficient for closed-loop optimization. Further, in our experience, the relaxations are often not tight enough for very dynamic motions. Another approach to convexify the centroidal dynamics problem is to only minimize the worst case L_1 bound on the angular momentum [23]. While this formulation allows us to solve the motion planning problem with a QP, it was only shown to be capable of generating motions with low angular momentum such as walking. For quadrupeds, with negligible leg inertia, the centroidal dynamics is also often approximated by linearizing the base rotation [24], [25]. In these approaches, the swing foot trajectories are predefined which restricts the possibilities for whole-body motions and may lead to physical inconsistency. In [26], DDP is used to solve an optimization problem with the centroidal dynamics and first-order kinematics in real time. This approach achieved re-planning frequencies suitable for real-time use. However, the approach relied on a low-level whole-body controller and it is unclear whether this controller solely tracked the motions generated by DDP or whether the low-level controller behaved as a dynamic filter to ensure physical consistency [27]. From our experience, the solve times for such methods increase for more dynamic motions such as bounding or rapid (forward) jumping due to the drastic changes in the momenta profiles.

In this work, we propose a nonlinear trajectory optimization framework that can be used in a real-time closed-loop model predictive control to generate whole-body motions using the kino-dynamic decomposition proposed in [20]. The dynamics optimization problem is solved efficiently by exploiting the biconvex structure of the centroidal dynamics. We previously explored this structure in [28] which leveraged the biconvex nature of the problem to formulate two separate, convex, sub-problems. Given the convexity of each sub-problem, an alternating procedure based on block coordinate descent was used which allowed the use of state-of-the-art QP solvers and resulted in a speedup in solve times. However, very little is understood about the convergence rates of block coordinate descent which makes it unreliable for MPC where new solutions are needed in a fixed time[29]. In this work, we explore a different approach that also exploits the biconvex structure of the centroidal dynamics but we formulate the optimization problem using the Alternating Direction Method of Multipliers (ADMM) [30], leading to a more efficient and reliable algorithm. We also split the biconvex dynamics differently which reduces the number of optimization variables.

Compared to the block coordinate descent algorithm, the ADMM algorithm provides favorable convergence properties such as the ability to reach acceptable solutions in fewer iterations and guaranteed sublinear convergence [30]. Crucially, due to the unconstrained nature of each sub-problem, each iteration is computationally cheap with respect to wall time which allows us to exploit the aforementioned convergence properties and make it attractive for use in an MPC fashion. In the proposed ADMM formulation, each convex sub-problem is solved using a custom implementation of the Fast Iterative Shrinkage Thresholding Algorithm (FISTA) [31]. This approach guarantees quadratic convergence of the convex sub-problems while enforcing a variety of constraints including second-order friction cone constraints. Our custom implementation exploits 1) the accelerated gradient nature of FISTA by warm starting the line search step to reduce solve times and 2) the sparse nature of our optimal control problem (OCP) as each iteration of the line-search only involves sparse matrix-vector multiplication which is less expensive compared to typical Quadratic Program (QP) or Quadratically constrained QP solvers which often involve expensive matrix decompositions. Due to this formulation, we solve the exact centroidal optimization problem without relaxing the friction cone constraints [32] or the dynamic constraints [21]. Finally, the first-order optimization procedure of FISTA increases robustness to convergence in the absence of true gradients [33], [34]. This situation often occurs in real robots due to a lack of accurate sensor measurements. Furthermore, problems such as ill-conditioning of the Hessians that second-order methods like DDP encounter are also absent with FISTA.

We also propose a second-order kinematics optimization formulation to generate smooth joint trajectories that track centroidal momentum profiles as required in the kino-dynamic setup. We choose to solve this kinematics problem with a DDP solver [35] to exploit the sparsity in the problem. The second-order nonlinear optimization removes the need to specify heuristic-based end-effector trajectories (e.g. via a spline-based swing foot trajectory) as is often done in MPC implementations [24], [36]. Although these methods of swing-foot generation work well for simple motions, they are often restrictive in nature and do not allow the algorithm the freedom to find trajectories that may utilize the full capabilities of the end-effectors. The nonlinear kinematics solver generates non-trivial swing foot trajectories to track the desired centroidal momentum provided by the dynamics optimization. Further, the automatic generation of smooth joint acceleration profiles allows direct computation and tracking of torques on the robot with a simple inverse dynamics controller. This removes the need for a complicated QP-based whole-body controller that often behaves as an additional dynamic filter.

We demonstrate our approach in closed-loop MPC on the real Solo12 quadruped robot [37] at 20 Hz to generate several gaits such as trotting, jumping, and bounding. We also display the robustness of the framework against external disturbances and terrain noise. A high-five motion is also shown to demonstrate the generality of the approach to non-trivial, acyclic motions. Since the framework does not relax or impose assumptions to make the original problem convex, we are able to generate a wide array of motions in real time. Furthermore, the same approach is used to generate various dynamic motions in MPC on a humanoid robot (Talos [38]) and another quadruped robot (AnYmal [39]) in simulation. This underlines the generality, robustness, and low compu-

tation times of our proposed framework, BiConMP, despite the changes in the robot mass distributions, number of joints (size of optimization problem), and nature of the robot. This is often not as easy with other approaches which use simplified dynamics whose assumptions may not hold valid for different robots. Finally, we empirically analyze the effects of the horizon length and re-planning frequency on the robustness and performance of the nonlinear MPC on the real robot (Solo12). To the best of our knowledge, this is the first reported empirical analysis of closed-loop nonlinear MPC performance for legged robots.

II. Background

This section introduces the background necessary to describe our approach. First, we discuss the centroidal momentum dynamics of a floating base robot. Next, we explain the kino-dynamic trajectory optimization scheme used to generate feasible whole-body multi-contact motions for legged robots. Finally, we briefly introduce an optimization technique used in the solver.

A. Centroidal dynamics

The rigid body dynamics of a floating base robot can be described as

$$\mathbf{M}(\mathbf{q})\dot{\mathbf{v}} + \mathbf{N}(\mathbf{q},\mathbf{v}) = \mathbf{S}^T\boldsymbol{\tau} + \sum_{j=1}^{N}\mathbf{J}_j^T\boldsymbol{\lambda}_j \tag{1}$$

where $\mathbf{q} \in \mathbb{R}^n \times SE(3)$ is the generalized configuration of the robot (joint positions and base pose), and $\mathbf{v} \in \mathbb{R}^{n+6}$ is the generalized velocity vector. $\mathbf{M}(\mathbf{q}) \in \mathbb{R}^{(n+6)\times(n+6)}$ is the mass matrix for the given robot configuration, $\mathbf{N}(\mathbf{q},\mathbf{v}) \in \mathbb{R}^{n+6}$ is the vector containing all generalized forces (Coriolis, centrifugal, gravity, etc), $\boldsymbol{\tau} \in \mathbb{R}^n$ is the vector of joint torques, $\mathbf{S}$ is a selection matrix reflecting the underactuation of the robot, $\mathbf{J_j} \in \mathbb{R}^{6\times(n+6)}$ are the end effector Jacobians and $\boldsymbol{\lambda}_j = [\mathbf{f}_j, \boldsymbol{\kappa}_j] \in \mathbb{R}^6$ is the vector of forces and torques applied at each end effector.

The dynamics can further be split into its actuated and unactuated parts [18], [20]

$$\mathbf{M}_u(\mathbf{q})\dot{\mathbf{v}} + \mathbf{N}_u(\mathbf{q},\mathbf{v}) = \sum_{j=1}^{N}\mathbf{J}_{u,j}^T\boldsymbol{\lambda}_j \tag{2a}$$

$$\mathbf{M}_a(\mathbf{q})\dot{\mathbf{v}} + \mathbf{N}_a(\mathbf{q},\mathbf{v}) = \boldsymbol{\tau} + \sum_{j=1}^{N}\mathbf{J}_{a,j}^T\boldsymbol{\lambda}_j \tag{2b}$$

where the subscript a, u correspond to actuated and unactuated dynamics respectively. The unactuated dynamics is equivalent to the Newton-Euler equations of the center of mass (CoM)

$$\begin{bmatrix}\dot{\mathbf{l}}\\ \dot{\mathbf{k}}\end{bmatrix} = \begin{bmatrix} m\mathbf{g} + \sum_{j=1}^{N} n_j\mathbf{f}_j \\ \sum_{j=1}^{N} n_j((\mathbf{r}_j - \mathbf{c}) \times \mathbf{f}_j + \boldsymbol{\kappa}_j)\end{bmatrix} \tag{3}$$

where $\mathbf{l}, \mathbf{k}$ are the linear and angular momentum [40], m is the robot mass, $\mathbf{g}$ is the gravity vector, $\mathbf{c}$ represents the center of mass CoM location, n_j is a binary integer that describes whether the end effector j is in contact, $\mathbf{f}_j, \boldsymbol{\kappa}_j, \mathbf{r}_j$ are the end effector force, torque, and location respectively. The linear momentum is related to the CoM velocity $\dot{\mathbf{c}}$ as $\mathbf{l} = m\dot{\mathbf{c}}$. The linear momentum and angular momentum can also be described in terms of the generalized joint configuration using the centroidal momentum matrix(D(q)) of the robot as $\begin{bmatrix}\mathbf{l}\\ \mathbf{k}\end{bmatrix} = \mathbf{D}(\mathbf{q})\mathbf{v}$, [40].

B. Kino-dynamic motion generation

Splitting the dynamics enables multi-contact motion generation by only considering the unactuated dynamics or centroidal dynamics of the robot. Subsequently, a feasible whole-body trajectory can then be determined based on the centroidal plan and desired whole-body tasks, provided there is sufficient torque authority [20][21]. This is an attractive approach since it breaks the original nonlinear optimization problem into two simpler sub-problems.

A desired motion plan using the centroidal dynamics can be generated by solving the following discrete optimal control problem (OCP)

$$\min_{\mathbf{c},\dot{\mathbf{c}},\mathbf{k},\mathbf{f},\boldsymbol{\kappa}} \quad \sum_{t=0}^{T-1}\phi_t(\mathbf{c}_t,\dot{\mathbf{c}}_t,\mathbf{k}_t,\mathbf{f}_t,\boldsymbol{\kappa}_t) + \phi_T(\mathbf{c}_T,\dot{\mathbf{c}}_T,\mathbf{k}_T,\mathbf{f}_T,\boldsymbol{\kappa}_T)$$

$$\text{s.t.} \quad \mathbf{c}_{t+1} = \mathbf{c}_t + \dot{\mathbf{c}}_t\Delta t \tag{4}$$

$$\dot{\mathbf{c}}_{t+1} = \dot{\mathbf{c}}_t + \sum_{j=1}^{N} n_t^j\frac{\mathbf{f}_t^j}{m}\Delta t + \mathbf{g}\Delta t \tag{5}$$

$$\mathbf{k}_{t+1} = \mathbf{k}_t + \sum_{j=1}^{N} n_t^j((\mathbf{r}_t^j - \mathbf{c}_t) \times \mathbf{f}_t^j + \boldsymbol{\kappa}_t^j)\Delta t \tag{6}$$

$$\forall_{t,j},\ \sqrt{(\mathbf{f}_{t,x}^j)^2 + (\mathbf{f}_{t,y}^j)^2} \le \mu\mathbf{f}_{t,z}^j \quad , \mathbf{f}_{t,z}^j \ge 0 \tag{7}$$

$$\forall_{t,j},\ \mathbf{r}_t^j \epsilon \Psi, \quad \forall \mathbf{c}_t \in \Omega, \quad \mathbf{c}_0, \dot{\mathbf{c}}_0 = \mathbf{c}_{init}, \dot{\mathbf{c}}_{init} \tag{8}$$

where $\phi_t(\mathbf{c}_t,\dot{\mathbf{c}}_t,\mathbf{k}_t,\mathbf{f}_t,\boldsymbol{\kappa}_t)$ is the running cost, $\phi_T(\mathbf{c}_T,\dot{\mathbf{c}}_T,\mathbf{k}_T,\mathbf{f}_T,\boldsymbol{\kappa}_T)$ is the terminal cost, Δt is the time discretization, μ is the friction coefficient, Ψ is the set of all allowed stepping locations, Ω are kinematic constraints written as bounds on the CoM position, $\mathbf{c}_{init}, \dot{\mathbf{c}}_{init}$ are the initial conditions for the CoM.

The optimal joint trajectory is generated by solving a whole-body kinematic optimizer which tracks the optimal centroidal momentum obtained from the previous step using the centroidal momentum matrix, along with additional full-body tasks, such as swing foot motion [21]. The generated momentum trajectory from the whole-body kinematic optimizer is then used as a soft constraint in the centroidal OCP to obtain refined centroidal and contact forces trajectories. This process is iterated until the two sub-problems converge [20] leading to a solution to the original problem.

One can then directly use plain inverse dynamics to recover actuated joint torques from desired state trajectories and contact forces using Eq. 2b

$$\boldsymbol{\tau}_{RNEA} = \mathbf{M_a}(\mathbf{q})\dot{\mathbf{v}} + \mathbf{N}_a(\mathbf{q},\mathbf{v}) - \sum_{j=1}^{N}\mathbf{J}_{a,j}^T\boldsymbol{\lambda}_j \tag{9}$$

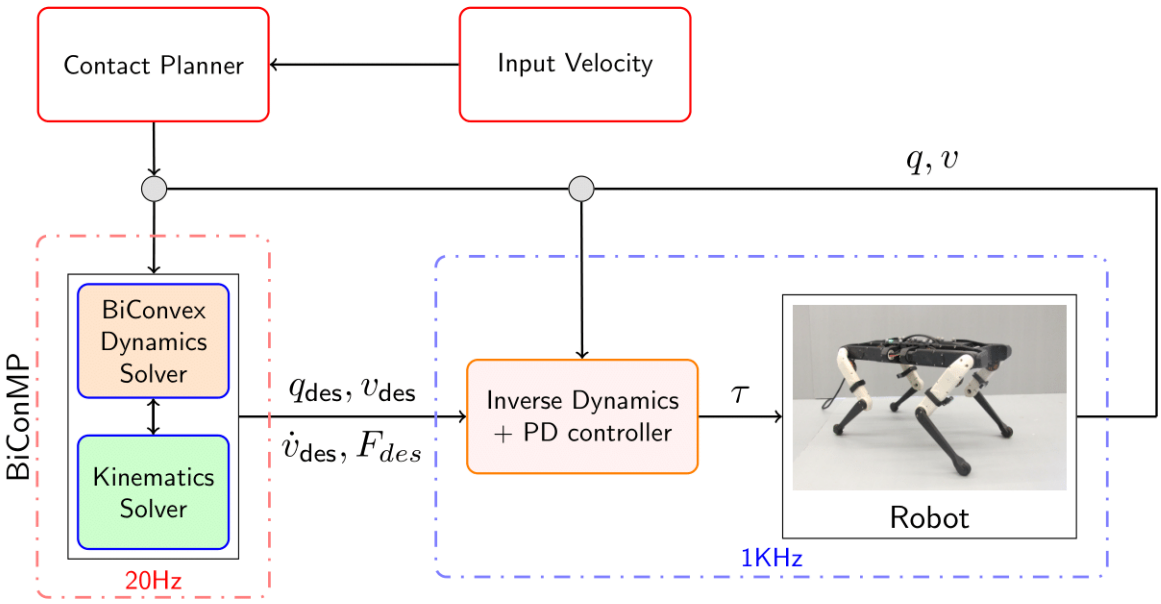

Fig. 1: A birds eye view of the entire nonlinear MPC framework. First, centroidal trajectories are generated using the ADMM framework explained in Section III-B. These trajectories are used within a DDP-based kinematic optimizer that generates the desired joint trajectories (Section III-D). The optimal force and joint trajectories from this kino-dynamic iteration are recomputed at 20 Hz and are used in unconstrained inverse dynamics (9) to compute the desired joint torques at 1 kHz. Finally, these actuator torques are summed up with a fixed low gain impedance joint controller that results in the torques sent to the robot actuators (20).

Note that in this work we do not use a constrained QP-based inverse dynamics as is usually done, but we simply use the computed joint positions, velocities, accelerations, and force trajectories in the Recursive Newton Euler Algorithm (RNEA) [41] to compute the torques.

C. Fast Iterative Shrinkage Thresholding Algorithm (FISTA)

Proximal gradient methods [42] is a popular family of algorithms used to solve problems of the form

$$\min_{\mathbf{x}} \quad T(\mathbf{x}) + I(\mathbf{x}) \tag{10}$$

where $\mathbf{x}$ is the optimization variable, $T(\mathbf{x})$ is the cost function to be optimized, and $I(\mathbf{x})$ is usually an indicator function that enforces feasibility constraints or forces $\mathbf{x}$ to remain inside a feasible set. The cost function $T(\mathbf{x})$ can be non-smooth, nonlinear, or convex and $I(\mathbf{x})$ is restricted to be convex. Each algorithm in the proximal gradient family varies slightly in the step length computation and update procedure for $\mathbf{x}$, however each iteration in the proximal methods is fundamentally of the form

$$\mathbf{x}_{k+1} = P_c(\mathbf{x}_k + t_k \nabla f(\mathbf{x}_k)) \tag{11}$$

where t_k is the step length, $\mathbf{x}_{k+1}$ is the value of the optimization variable at the next iteration $(k+1)$, and P_c is the proximal operator that ensures that after the descent step is taken, the new $\mathbf{x}_{k+1}$ lies within the domain of $I(\mathbf{x})$ [42]. Depending on the function represented by $I(\mathbf{x})$, the proximal operator may or may not have a closed-form solution. The function $I(\mathbf{x})$ can only be used if it is possible to compute a closed-form solution for the proximal operator [31]. Consequently, arbitrary inequality constraints cannot be enforced with these methods. However, in the presence of closed-form solutions (as is in our case) these proximal operators are very cheap to evaluate.

In the specific case when the cost function $T(\mathbf{x})$ is convex, a proximal gradient method, the Fast Iterative Shrinkage Thresholding Algorithm (FISTA) is an attractive choice. FISTA is an accelerated first-order gradient method that displays quadratic convergence. Algorithm 1 shows the steps in FISTA. The key point in the algorithm, as compared to other proximal methods, is the introduction of the auxiliary optimization variable $\mathbf{y}_k$ and the update procedure of t_k which is the primary reason for the quadratic convergence. The step length L_k is chosen based on a sufficient decrease condition (similar to Wolfe's condition [3]). For more details regarding the algorithm, we refer the reader to [42], [31].

In the following, we will exploit FISTA's quadratic convergence properties to quickly solve the convex sub-problems of the centroidal OCP (see Section III-C). The indicator function in our formulation enforces kinematic (box constraints) and friction cone constraints (second-order cone projections) for which the proximal operator exists. In practice, FISTA is computationally very cheap because it does not need the inversion of the Hessian to achieve quadratic convergence, and the proximal step that enforces feasibility of $\mathbf{x}$ (inequality constraints) is inexpensive.

Algorithm 1: FISTA algorithm

Initialize optimization variables: $\mathbf{y}_0 = \mathbf{x}_0, t_0 = 1$
set $k = 0$
while $k <$ *maximum iterations* **do**
  Pick $L_k > 0$
  $\mathbf{x}_{k+1} = prox_{\frac{1}{L_k}I}(\mathbf{y}_k + \frac{1}{L_k}\nabla f(\mathbf{y}_k))$
  $t_{k+1} = \frac{1+\sqrt{1+4t_k^2}}{2}$
  $\mathbf{y}_{k+1} = \mathbf{x}_{k+1} + \frac{t_k - 1}{t_{k+1}}(\mathbf{x}_{k+1} - \mathbf{x}_k)$

III. Approach

We now introduce the main components of our solver, the BiConMP. First, we present the biconvex dynamics solver in detail and explain how it exploits the structure of the nonlinearity in the centroidal OCP to solve the problem efficiently. Second, we discuss the DDP based second order kinematics formulation used in the framework to solve the nonlinear problem. Finally, we give a birds eye view of how the BiConMP is used in a non-linear MPC setting to generate full body motions in real-time.

A. Biconvexity in centroidal dynamics

The unactuated dynamics constraints (4), (5), (6) are nonlinear due to the cross product term in the angular momentum constraint (6). This non-convexity makes the problem inherently difficult to solve. These constraints, however, have an interesting feature: they are biconvex [30]. That is, the constraints are affine in terms of $\mathbf{c}, \dot{\mathbf{c}}, \mathbf{k}$ when $\mathbf{f}_t^j, \forall_{t,j}$ is kept constant and vice-versa. Consequently, the terms of the discrete constraints (4), (5), (6) can be rearranged as an affine equation in terms of $\mathbf{X}$, $A(\mathbf{F})\mathbf{X} = b(\mathbf{F})$ and $\mathbf{F}$, $A(\mathbf{X})\mathbf{F} = b(\mathbf{X})$, where $\mathbf{X} = \{\mathbf{c}_t, \dot{\mathbf{c}}_t, \mathbf{k}_t \dots\}$ and $\mathbf{F} = \{\mathbf{f}_{t,x}^j, \mathbf{f}_{t,y}^j, \mathbf{f}_{t,z}^j \dots\}$, for $t = 0, \dots, T-1$, $j = 1, \dots, N$. Here $A(\mathbf{F})$ is a matrix whose elements depend on $\mathbf{F}$ and the centroidal dynamic constraints. Similarly, $A(\mathbf{X})$ is a matrix depending on $\mathbf{X}$ and dynamics constraints. $b(\mathbf{F})$ and $b(\mathbf{X})$ are vectors whose elements depend on $\mathbf{F}$ and $\mathbf{X}$ respectively.

B. Biconvex optimization with ADMM

Using the observation from the previous section, the centroidal dynamics OCP in Section II-B can be alternatively formulated as shown below, to highlight and exploit the biconvexity in the problem:

$$\min_{\mathbf{X},\mathbf{F}} \quad \Phi(\mathbf{X}) + I(\mathbf{X}) + \Phi(\mathbf{F}) + I(\mathbf{F}) \tag{12}$$

$$\text{s.t.} \qquad G(\mathbf{X}, \mathbf{F}) = 0 \tag{13}$$

where $\Phi(\mathbf{X}), \Phi(\mathbf{F})$ are the running and terminal cost functions in terms of $\mathbf{X}$ and $\mathbf{F}$ respectively, $G(\mathbf{X}, \mathbf{F}) = 0$ are the nonlinear constraints (4) such that it is bi-affine in terms of X and F, (5), (6) and the initial state constraints $(\mathbf{c}_0, \dot{\mathbf{c}}_0 = \mathbf{c}_{init}, \dot{\mathbf{c}}_{init})$, $I(\mathbf{X})$ is an indicator function that enforces kinematic constraints ($\forall \mathbf{c}_t \in \Omega$, section II-B) while $I(\mathbf{F})$ is an indicator function that enforces unilaterality and friction cone constraints. This formulation makes it possible to exploit the biconvexity in the dynamics and solve the nonlinear problem very efficiently using the Alternating Direction Method of Multipliers (ADMM) [30]. We want to use the ADMM algorithm because it has an interesting property of reaching reasonably good solutions in a few iterations [30]. The downside however is that ADMM takes far more iterations as compared to a second-order method (DDP) to obtain a high-resolution result. In our application (closed loop MPC), solutions that have dynamic constraint violation tolerances lower than the sensor noise and satisfy feasibility constraints are sufficient to be successfully deployed on the robot (as shown in our experiments). Consequently, when the need arises, ADMM allows us to terminate the solver before complete convergence (early termination) to ensure that the new trajectory is available in the desired time (real-time). At the same time, be sure that a reasonably good solution that is realizable on the robot is available. Furthermore, ADMM has a sub-linear convergence rates, which is not guaranteed with block-coordinate descent.

The proposed BiConMP solves the dynamics optimization by iteratively solving the two convex sub-problems (shown in Algorithm 2) as a part of a larger ADMM optimization scheme [30]. The ADMM algorithm solves both the convex sub-problems (force ($\mathbf{F}$) problem and state ($\mathbf{X}$) problem) iteratively until the dynamics violation falls below the desired tolerance (exit criteria). The dynamics violation is computed as

$$||A(\mathbf{F}_{k+1})\mathbf{X}_{k+1} - b(\mathbf{F}_{k+1})||^2 \leq \epsilon_{dyn} \tag{14}$$

where ϵ_{dyn} is the termination tolerance.

Algorithm 2: Biconvex Centroidal Optimization

Initialize optimization variables: F_0, X_0, P_0, ρ
set $k = 0$
while $k <$ *maximum iterations* **do**
  $\min_F \Phi(F) + \frac{\rho}{2}||A(X_k)F - b(X_k) + P_k||^2 + I(F)$
  $\min_X \Phi(X) + \frac{\rho}{2}||A(F_{k+1})X - b(F_{k+1}) + P_k||^2 + I(X)$
  $P_{k+1} = P_k + A(F_{k+1})X_{k+1} - b(F_{k+1})$
  if $||A(F_{k+1})X_{k+1} - b(F_{k+1})||^2 \leq \epsilon_{dyn}$ **then**
    terminate

The cost function in the state sub-problem is always of the form $\Phi(\mathbf{X}) = (\mathbf{X} - \mathbf{X}_{nom})^T \mathbf{W}_x (\mathbf{X} - \mathbf{X}_{nom})$ where $\mathbf{X}_{nom}$ is a nominal trajectory, $\mathbf{W}_x$ is a diagonal weight matrix. The nominal trajectory gives the solver a heuristic idea of the desired centroidal trajectory (need not be dynamically consistent). In practice, the nominal trajectory usually consists of the desired base height and velocity (forward and sideways velocity). In addition, for some motions (cyclic gaits), a nominal angular momentum trajectory is provided to track the desired base orientation since direct orientation tracking is not possible with the centroidal OCP formulation (it is possible with the kinematics solver, section III-D). The nominal angular momentum trajectory is computed as follows:

$$\mathbf{k}_{nom} = \mathbf{w}\, log_3(\mathbf{q}_0 \ominus \mathbf{q}_{des}) \tag{15}$$

where $\mathbf{q}_0$ and $\mathbf{q}_{des}$ are the current and desired base orientation quaternions. $\ominus$ is the difference operator for quaternions and log_3 is the logarithmic map from $SE(3)$ to $\mathfrak{se}(3)$. $\mathbf{w}$ is a 3 dimensional weight vector. $\mathbf{k}_{nom}$ is set as the desired nominal angular momentum value for each time step in the planning horizon. In practice, the desired base orientation quaternion is always set to $[0, 0, 0, 1]$ which corresponds to zero roll, pitch, and yaw.

***Remark** 1:* Note that the exact Netwon-Euler dynamics are considered in the centroidal dynamics optimization. The inertia of the base is only ignored (assumes unit inertia) while computing the nominal angular momentum to be tracked in the cost (Eq. (15)). We use this heuristic computation because the resulting trajectories from BiConMP were able to track the desired orientation (even during external pushes) on the robot. A more accurate nominal trajectory could be used if the need arises. Note that the exact base orientation is optimized concurrently in the kinematic solver.

In the force sub-problem, the cost is $\Phi(\mathbf{F}) = \mathbf{F}^T\mathbf{W}_f\mathbf{F}$ which penalizes unnecessary contact forces. To enforce complementarity constraints [9] based on the contact plan, coefficients of the variables corresponding to the time step for the given end effector where contact does not exist (elements of the matrix $A(\mathbf{X}_k)$), are set to zero in the force sub-problem. This automatically sets the planned forces to zero at that time step after optimization because of the cost function.

C. Convex sub-problems

The state sub-problem can be solved using any Quadratic Program (QP). The force subproblem will need a Quadratically Constrained QP solver due to the second-order friction cone constraints[3]. These constraints could be relaxed and made linear, but this leads to more conservative motions (discussed later). In the BiConMP, we use FISTA (section II-C) because of the following favorable reasons: 1) FISTA maintains quadratic convergence even while enforcing constraints, 2) FISTA has low computation cost since its a first-order method. The biconvex problem was formulated with indicator functions enforcing inequality constraints (kinematic and friction cone constraints) because FISTA can efficiently impose them using proximal operators, which are computationally inexpensive.

1) State sub-problem (optimizing for $\mathbf{X}$): the kinematic constraints are enforced by the indicator function $I(\mathbf{X})$. This constrains the CoM to stay within a cube whose size depends on the location of the contact points at the particular time step. The proximal operator then becomes a box projection [31] while computing a descent step in FISTA for the k^{th} iteration,

$$\mathbf{X}_{k+1} = max(min(\mathbf{X}^*_k, \mathbf{u}), \xi) \tag{16}$$

where

$$\mathbf{X}^*_k = \mathbf{Y}_k + \frac{1}{L_k}(\Phi^{'}(\mathbf{Y}_k) + \rho A(\mathbf{F}_k)^T(A(\mathbf{F}_k)\mathbf{Y}_k - b(\mathbf{F}_k) + \mathbf{P}_k))$$

is the updated $\mathbf{X}$ parameter after the descent step is taken with L_k as the line search step, $\mathbf{F}_k$ is an auxilary variable to $\mathbf{X}_k$ used in FISTA (subsection II-C), $\mathbf{u}$ and ξ are the upper and lower bounds required to be satisfied for kinematic feasibility. For the components of $\mathbf{X}$ corresponding to velocity and angular momentum, the upper and lower bounds are set to $+\infty$ and $-\infty$ to enforce bound constraints only on the CoM location.

2) Force sub-problem (optimizing for $\mathbf{F}$): the indicator function $I(\mathbf{F})$ enforces second order friction cone constraints [43]. The proximal operator enforcing this constraint for each group f_x, f_y, f_z corresponding to one contact point and time step in the F vector is

$$\begin{cases} (0,0,0) & \mu\sqrt{(f_k^x)^2 + (f_k^y)^2} \leq -f_k^z or f_k^z < 0 \\ (\beta f_x, \beta f_y, \gamma f_z) & \mu\sqrt{(f_k^x)^2 + (f_k^y)^2} > f_k^z \\ (f_x, f_y, f_z) & \sqrt{(f_k^x)^2 + (f_k^y)^2} \leq \mu f_z \end{cases}$$

where μ is the friction coefficient,

$$\beta = \frac{\mu^2\sqrt{(f_k^x)^2 + (f_k^y)^2} + \mu f_z}{(\mu^2+1)\sqrt{(f_k^x)^2 + (f_k^y)^2}} \tag{17}$$

and

$$\gamma = \frac{\mu\sqrt{(f_k^x)^2 + (f_k^y)^2} + f_z}{(\mu^2+1)}. \tag{18}$$

Subsequently, after a descent step is taken in FISTA to update the force vector

$$\mathbf{F}^*_k = \mathbf{Y}_k + \frac{1}{L_k}(\Phi^{'}(\mathbf{Y}_k) + \rho A(\mathbf{X}_k)^T(A(\mathbf{X}_k)\mathbf{Y}_k - b(\mathbf{X}_k) + \mathbf{P}_k))$$

every f_x^*, f_y^*, f_z^* in $\mathbf{F}^*_k$ is then projected based on the friction cone proximal operator to obtain the force vector $\mathbf{F}_{k+1}$ for the next iteration. Here $\mathbf{Y}_k$ is the auxiliary variable to $\mathbf{F}_k$ (section II-C). The projection of each group of forces independently works mathematically with the FISTA algorithm because the control decision variables are independent [31] of each other in the centroidal problem. There is no explicit constraint enforcing unilaterality in f_z because the friction cone projection implicitly enforces $f_z \geq 0$. The interesting point to note here is that with FISTA, the second-order cone projection can be enforced directly while still maintaining quadratic convergence rates. In contrast, other QCQP solvers do not have quadratic convergence properties. They are also more computationally intensive because they need second-order information. To improve solve time the friction cones are often approximated as linearized (i.e. converted to pyramidal polyhedral constraints) to solve them with QPs. This usually results in conservative trajectory solutions which are not desirable when dynamic motions are to be performed.

3) FISTA implementation: To reduce the solve times in each iteration we specialize our implementation of the FISTA solver to exploit certain additional details specific to the dynamics optimization problem. Firstly, the analytical gradients of the cost function are used to compute the descent direction instead of using auto-diff or numerical differentiation methods. Secondly, the sparsity of the matrices $A(\mathbf{X}), A(\mathbf{F})$

is exploited during the matrix-matrix and matrix-vector computation in each iteration. Thirdly, the matrix multiplications, such as $A(\mathbf{X}_k)^T A(\mathbf{X}_k)$, $A(\mathbf{X_F})^T A(\mathbf{X_F})$, etc., which are only computed once in each convex sub-problem, are cached and reused. Finally, the accelerated gradient nature of the solver ensures that the first line search step is successful after a certain number of iterations are reached. In practice, we noticed that almost the same step lengths were used in each iteration. Subsequently, we warm start the solver with these line search steps which significantly improves the solve times as the solver no longer searches for the optimal values during run time. The warm starting of the line search works well in FISTA because of an interesting property in its convergence proof which states that any line search parameter bigger than the Lipschitz constant of the cost function will satisfy Wolfe's condition (chapter 10 [31]). In this case, we conjecture that the empirically determined value satisfies this property. Warm starting the line search would not be possible conveniently with other QP solvers.

***Remark** 2:* The quadratic convergence property of FISTA, the computationally inexpensive proximal operators used to enforce the inequality constraints, and the above-mentioned details in the implementation significantly improve the solve times which play a crucial role in being able to re-plan online on the real robot. It is important to note that, thanks to FISTA, we have quadratic convergence for each sub-problem of the ADMM framework. We however only expect super-linear convergence for the whole centroidal trajectory optimization problem.

D. Kinematics solver

The full-body kinematics trajectory generation problem described in section II-B is also nonlinear in nature. In the BiConMP, the problem is solved quickly using Differential Dynamic Programming (DDP). DDP exploits the block diagonal structure of the matrices while optimizing the problem and also shows quadratic convergence [44], [45]. We use Crocoddyl [35], an open-source DDP implementation. We formulate the problem as

$$\min_{\mathbf{q},\mathbf{v},\dot{\mathbf{v}}} \quad \sum_{t=0}^{T} \Phi^t_{mom}(\mathbf{l}^*_t, \mathbf{k}^*_t) + \Phi^t_{CoM}(\mathbf{c}^*_t) + \Phi^t_{eff}(\mathbf{q}_t, \mathbf{v}_t) + ||\dot{\mathbf{v}}||$$
$$\text{s.t.} \quad \mathbf{q}_{t+1} = \mathbf{q}_t \oplus \mathbf{v}_t\, \Delta t, \quad \mathbf{v}_{t+1} = \mathbf{v}_t + \dot{\mathbf{v}}_t\, \Delta t \qquad (19)$$

where, $\Phi^t_{mom}(\mathbf{l}^*_t, \mathbf{k}^*_t) = ||D(\mathbf{q}_t)\mathbf{v}_t - \begin{bmatrix}\mathbf{l}^*_t \\ \mathbf{k}^*_t\end{bmatrix}||$ is a momentum cost that tracks the optimal linear and angular momentum computed by the centroidal OCP (Algorithm 2), $\Phi^t_{CoM}(\mathbf{c}^*_t)$ is the center of mass tracking cost with the optimal CoM trajectory ($\mathbf{c}^*_t$) obtained from the centroidal OCP, $\Phi^t_{eff}(\mathbf{q}_t, \mathbf{v}_t)$ is the end effector locations and velocity cost, and $||\dot{\mathbf{v}}||$ is a penalty on the control. In practice, the cost on the control encourages smooth motions of the end effectors, especially during contact transitions. For example, during the landing phase of a jumping motion, the kinematics solver retracts the legs of the robot in the air so that a large torque is not needed at the time of contact to bring the legs to rest (satisfy complementarity constraints). In essence, this reduces the impact of the legs during landing and makes the motion smooth on the real robot. This is one of the advantages of using a nonlinear kinematics formulation that plans a full body motion by taking the future into account.

E. Model predictive control pipeline

An overview of the entire framework is shown in Fig. 1. Given the current states of the robot $q_{init}, v_{init}, \dot{v}_{init}$, desired gait, planning horizon, and velocity, a contact plan is either generated and adapted using the Raibert controller [46] or pre-defined without contact adaptation for acyclic or general motions. The BiConMP framework takes the input states and computes the optimal end effector forces, joint positions, joint velocities, and joint acceleration trajectories for the entire horizon. Given the desired joint trajectories and contact forces, we use (9) along with a low joint impedance around the desired states to compute the desired torques at 1 kHz (Equation (20)). The desired torques are then sent to the robot which is tracked on board at 10 kHz. The BiConMP control loop is re-computed at 20 Hz (50 ms) to update for optimal motion and control trajectories in real-time.

In our BiConMP framework, we re-plan the whole-body trajectories every 50 ms, for a horizon larger than 50 ms, based on feedback from the current state of the robot. The feedforward torques are computed every 1 ms with inverse dynamics (Equation (9)) based on the open loop trajectories in between two re-planning instances. Finally, a low joint impedance around the desired states is added to the computed torque to result in the final joint torques

$$\tau_i = \tau_{RNEA,i} + K_p(\mathbf{q}_{d,i} - \mathbf{q}_{r,i}) + K_y(\mathbf{v}_{d,i} - \mathbf{v}_{r,i}) \quad (20)$$

where τ_i is the torque sent to joint i, $\tau_{RNEA,i}$ is computed using interpolated values of $\mathbf{f}_i$, $\mathbf{q}_i, \mathbf{v}_i$, and $\dot{\mathbf{v}}_i$ between each MPC cycle, K_p and K_y are the joint position and velocity impedance gains respectively. Subscripts d and r stand for desired and actual respectively. In the rest of the paper, we refer to the RNEA based controller (20) as the inverse dynamics (ID) controller.

When the provided contact plan is longer than the desired MPC horizon (e.g. for acyclic motions below), the plan is segmented into a smaller section matching the desired horizon length and then provided to the BiConMP. As time elapses, the segment is shifted (moving horizon) to select the part of the contact plan starting from the elapsed time t and ending at $t + T$ where T is the desired horizon. For gaited motions, the contact plan is automatically updated based on the time that has elapsed which determines which phase the legs should be in and for how long depending on the gait parameters and desired horizon length (horizon is kept constant after the start of the motion). The desired velocity is also updated at every cycle based on the user input.

***Remark** 3:* We would like to emphasize the importance of each component in our BiConMP pipeline. First, we exploit the biconvex structure in the centroidal dynamics and efficiently generate centroidal trajectories while respecting

force constraints (Section III-B). Second, for solving the convex sub-problems in the centroidal problem, we use FISTA which ensures quadratic convergence even with second-order friction cone constraints. Third, we solve a simple second-order whole-body kinematic optimizer given the momentum profiles provided by the centroidal dynamics optimization. This allows for drastically reduced computation time compared to the approaches using full-body dynamics DDP for the kinematics problem [47]. Fourth, given the planned forces from centroidal MPC and desired joint trajectories from the IK, we compute the joint torques using Eq. (9) without a need to solve a constrained whole-body inverse dynamics. Note that contrary to [26], we intentionally did not use constrained inverse dynamics on top of our MPC block to demonstrate the quality of the plans generated by our approach without the need to further filter them for physical consistency [27]. Specifically, torque references computed from our MPC algorithm (which enforces friction cone constraints) can be applied directly to the robot.

IV. Experiments

In this section, we present results obtained on a real Solo12 quadruped [37] along with simulation results on a humanoid and another quadruped robot. We first study the optimizer, including effects on the solve times as the size of the optimization problem changes and the behavior of termination criteria used in the biconvex dynamics solver. Next, we present the different motions (cyclic and acyclic) generated on Solo12 along with the performance of the BiConMP in various scenarios to test the robustness of our approach. Finally, we present simulation results on the AnYmal quadruped and the Talos humanoid to show that the approach can be directly applied to legged robots with other morphologies and mass distributions. The attached video illustrates all these experiments.

A. Implementation details

The entire BiConMP is implemented in C++. The biconvex dynamics optimizer is implemented from scratch including a custom implementation of FISTA. Croccoddyl [35] is used to solve the kinematics problem. The code for the BiconMP is available in this GitHub link

All experiments were run on a Dell precision 5820 tower machine with a 3.7 GHz Intel Xeon processor and rt-preempt kernel. Robot Operating System 2 (ROS 2) was used to handle the multi-threading requirement of the approach to communicate between the 1 kHz inverse dynamics control loop and the 20 Hz MPC loop. The BiConMP is run on a node using a service client setup while the main inverse dynamics control loop (20) is run on a separate node at 1 kHz. The BiConMP service node is called at 20 Hz to update the plan and provide it to the inverse dynamics controller. The desired torque commands are then computed and provided to Solo12 via an Ethernet channel.

B. Solver analysis

1) Solve Times: To analyze the solve times of the BiConMP as a function of the number of collocation points/problem size, three motions (trot, jump, and bound) are used. These cyclic gaits are used as it is straightforward to change the horizon of the problem for this analysis. For each of these motions, the weights of the optimization problem, step time, discretization time, tolerances, etc., are kept the same and only the horizon of the problem is increased. In all cases, the solver is terminated only after satisfying the termination criteria. The resulting solve times are shown in Fig. 3. The top 3 plots contain the solve times from the dynamics biconvex solver, the DDP-based kinematics solver and the total solve time (including miscellaneous operations like cost creations), respectively. The biconvex dynamics solver shows a linear increase in the solve times as the problem size increases. The kinematic solver also shows an almost linear growth with an increase in the number of collocation points. The solver does violate this trend at times depending on the termination criteria used in Crocoddyl [35]. However, the kinematics solver maintains a strong linear behavior in the problem size that is mainly important in this work to achieve MPC (between 6-12 collocation points). On the other hand, the total solve times of the BiConMP framework maintain a linear growth in the solve rates. The solver always remains real-time, that is, it converges faster than the horizon of the plan. For example, the solver takes less than 0.85 seconds to generate a jump motion with a horizon of 7 seconds (140 collocation points $\times$ 0.05). The third plot also shows that the biconvex solver takes the most time in the framework. Consequently, a further decrease in solve times can be achieved by warm starting the solver with pre-computed solutions or by highly optimizing the code [48].

2) Termination Criteria: The dynamics violation is used as the termination criteria for the dynamics biconvex solver (section III-B). The solver is terminated when the centroidal dynamics constraints fall below the threshold of 0.001 or until we hit a maximum number of ADMM iterations. Each iteration here refers to one ADMM iteration. In Fig. 4, the dynamic violation vs the number of iterations is plotted for the three motions discussed previously. Each convex sub-problem in the biconvex ADMM problem is solved until the norm of the gradient falls below $1e-5$. High tolerance is necessary to ensure that the main biconvex problem converges to high-quality solutions. The dynamics violation rapidly decreases to a small value across all motions in alignment with the sublinear convergence property of the ADMM algorithm [30]. Note that a reasonable solution is found in a few iterations as reflected by the rapid drop in the dynamic violation. This is in alignment with the property of ADMM to get to roughly good solutions in a few iterations (section III-B). This allows us to set a maximum number of iterations of the ADMM algorithm to guarantee the real-time performance of our framework. We found that in practice there are times when we may hit the maximum number of iterations before our solver finds a solution that satisfies our criteria for acceptable dynamic violation. Even though our solutions don't fall within our predefined threshold for dynamic violation, we found empirically that

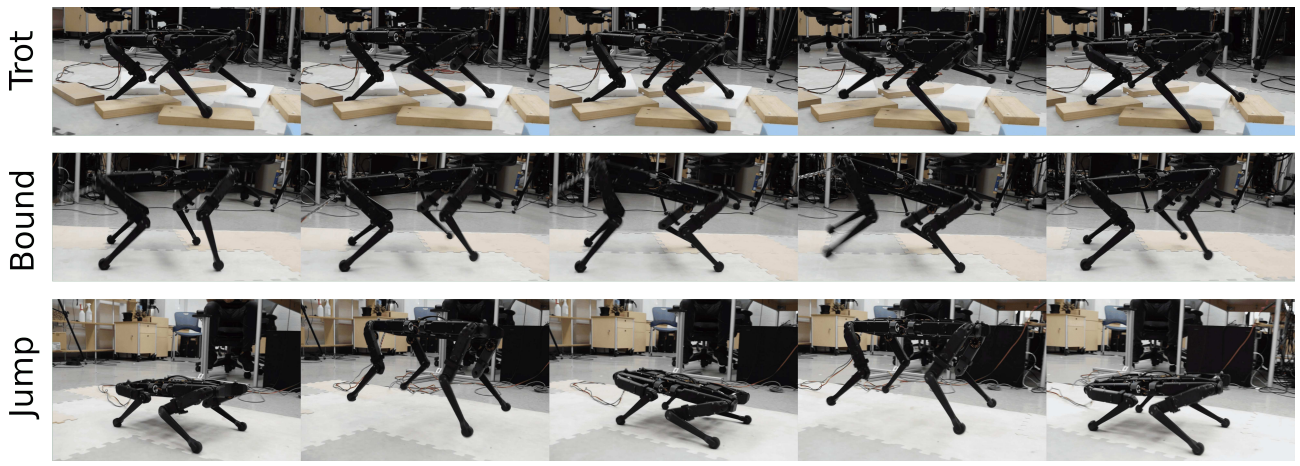

Fig. 2: Different motions demonstrated on the real Solo12 robot.

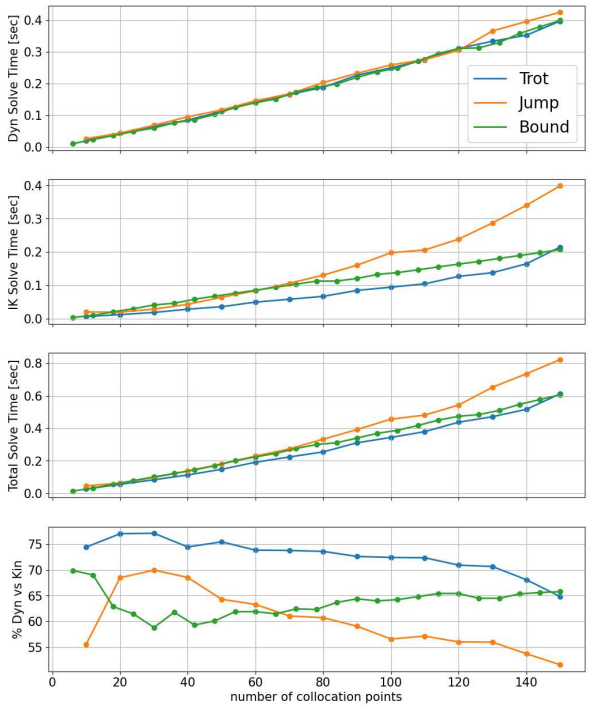

Fig. 3: Solve times vs the number of collocation points.

these trajectories are of sufficient quality to be executed on the robot. In practice, setting the maximum number of ADMM iterations to 50 allowed us to run all of the required motions on the robot. The kinematics solver is terminated based on the default settings provided by Crocoddyl.

C. Cyclic gaits

We generated different gaits such as trots, jumps, and bounds for the Solo12 quadruped. The resulting motions are shown in Fig. 2. Table I outlines the parameters used to design the gaits. Figure 5 shows the actual and desired base angular velocity about the Y axis (pitching axis) and the desired forces from the planner. The BiconMP is able to generate a bounding motion with considerable change in angular momentum and pitch magnitude.

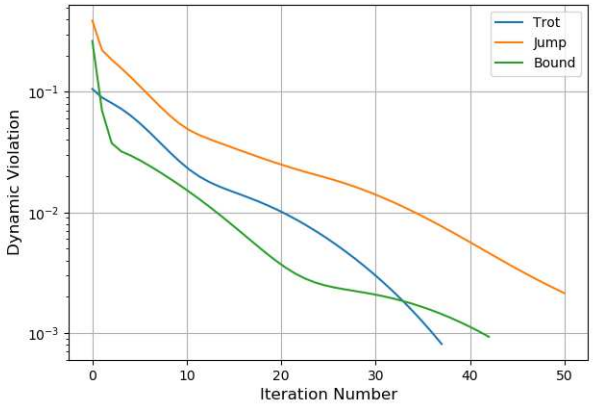

Fig. 4: Dynamic violation as compared to iterations in the dynamics optimizer.

We observed empirically that a smaller gait horizon than the ones chosen for each motion discussed above often led to the solver diverging after the completion of a few gait cycles. We hypothesize that this instability is due to the lack of a terminal cost/constraint that ensures the viability of the gait [49]. In the presence of a suitable terminal cost, the horizon could be reduced [50]. A planning horizon of 2 gait cycles was necessary to ensure the stability of the solver for bounding motions. The need for a longer planning horizon became especially necessary at higher speeds as these motions require tighter regulation of the angular momentum. Specifically, in order to create ground reaction forces to control motions with higher angular momentum, we believe a higher amount of control authority is required over time to bring the motion to a viable state that can be tracked successfully.

The BiConMP is able to track desired linear and angular velocities accurately on the robot irrespective of the gait. In Fig. 6, the velocity tracking performance for the trot and

Motion	Stance Duration (s)	Gait Duration (s)	Collocation Discretization Δt (s)	Number Of Collocation Points
Trot	0.15	0.3	0.03	10
Jump	0.2	0.5	0.05	10
Bound	0.15	0.3	0.05	12

TABLE I: Gait parameters for the various gaits tested on hardware.

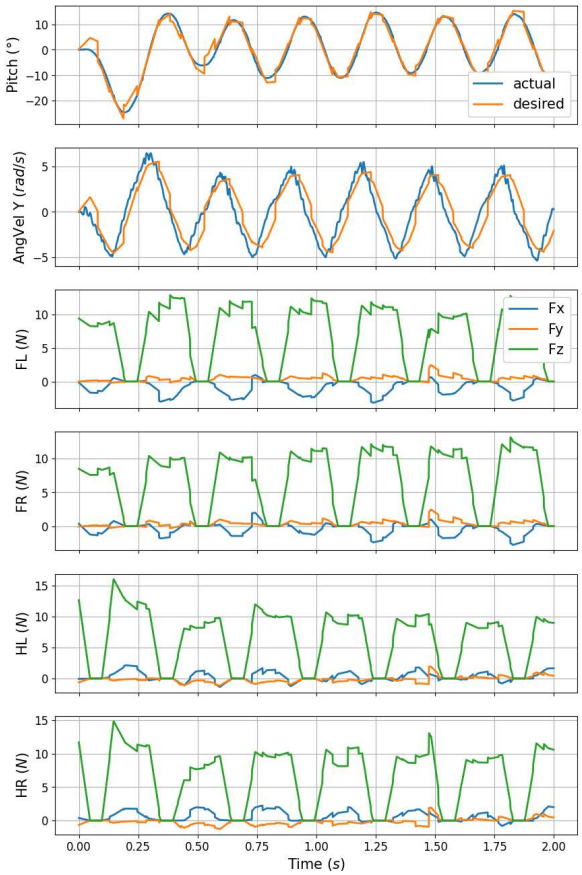

Fig. 5: Angular velocity and forces during bounding gait.

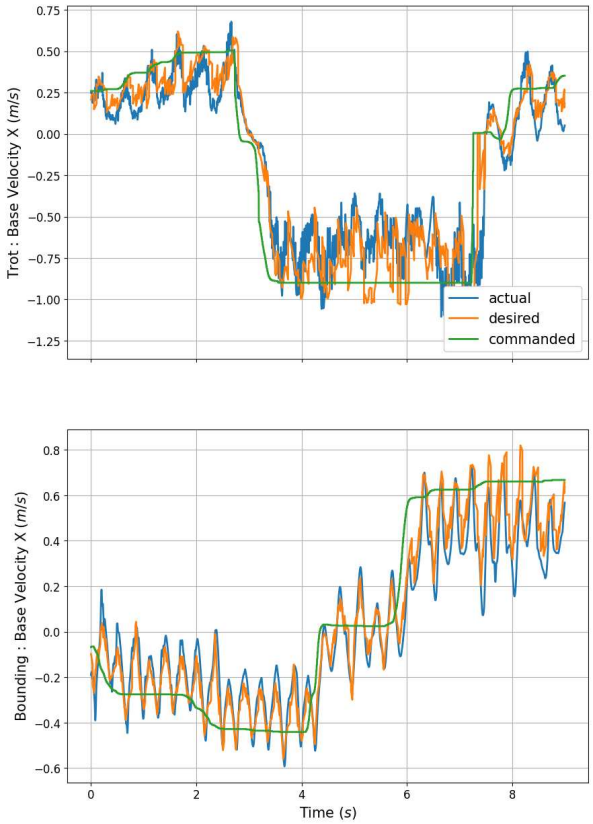

Fig. 6: Velocity tracking for trot and bound gait on the real robot. The x-axis is the elapsed time in seconds.

bounding gaits are shown. The framework is able to trade off the desired velocity input commands while staying within the limit cycle in real time. The motions are also robust to unforeseen disturbances (push recovery) and unaccounted uneven terrain as shown in the accompanying video.

1) Solve times in real experiments: To evaluate the solve times, each gait (trot, jump, and bound) is run on the robot in 3 different scenarios for 15 seconds (300 replan calls of BiConMP), i) flat ground with no disturbance, ii) flat ground with external disturbances, iii) uneven terrain (step height of 5-8 cm ($20-32\%$ of the nominal base height)) without external disturbances. Table. II shows the solve time statistics for each of these scenarios. As can be seen, the max solve time (worst case solve time) does not exceed the replanning time of 50 ms (20 Hz) and remains real-time regardless of the circumstances. Also, it is interesting to note that the mean solve times for each motion in the presence of terrain and pushes are quite similar to the flat ground scenario, which means that the solver remains unaffected by uncertain situations. On the other hand, the solve times with the same gait parameters/weights were lower on average by about 5-6 milliseconds without sensor noise (such as in simulation). Hence, to achieve higher re-planning frequencies, obtaining clean sensor data seems important.

The solve times of the framework for different motions range between 20-35 milliseconds on average, which is about 20-35 control cycles. Since the solve times are not negligible there exists a plan lag or delay from when sensor input is received and when the new plan is available. This lag has been shown to cause instabilities on real robots during run times and several approaches have been proposed to deal with this issue [51]. In our experiments, we skip the section of the plan that falls between the planning time and track the rest of the plan with the inverse dynamics controller. That is if a new plan is requested at $T = 0$ and the plan is available after t milliseconds, the plan from t milliseconds to the end is used assuming that the robot is close to the plan at t milliseconds. Even though this might not necessarily be true all the time (for example - push recovery, terrain noise) this strategy did not affect the stability of the gaits on the robot

Gait	Scenario	Replan Frequency (Hz)	Mean Solve Time (ms)	Standard Deviation (ms)	Max Solve Time (ms)
Trot	Flat Ground	20	23.47	2.45	32.8
Trot	Push	20	21.49	3.1	33.03
Trot	Terrain	20	26.13	4.45	36.5
Jump	Flat Ground	20	23.32	6.7	44.2
Jump	Push	20	22.45	3.4	37.8
Jump	Terrain	20	29.55	7.37	40.4
Bound	Flat Ground	20	27.15	5.2	42.6
Bound	Push	20	27.16	5.3	43.1
Bound	Terrain	20	27.27	5.3	40.0

TABLE II: Solve times of the BiConMP on Solo12 for various gaits and scenarios.

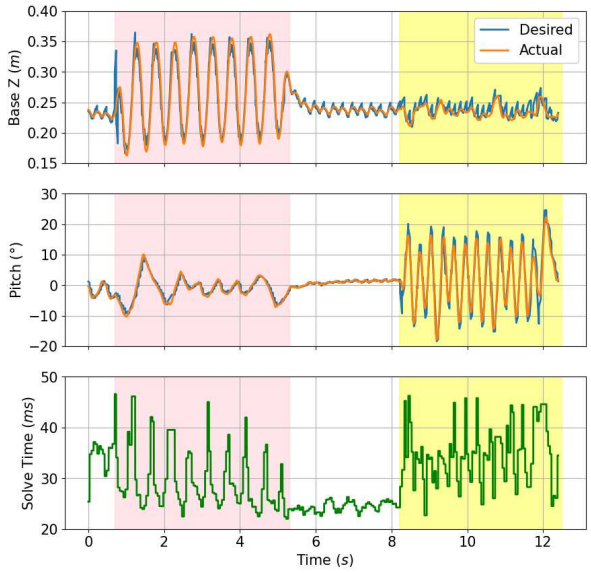

Fig. 7: Solve times during live gait transitions. The white color region corresponds to the trot gait, the pink section represents the jumping gait and the yellow section depicts the bounding phase.

when compared to a simulation where time can be frozen until a new plan is available.

2) Gait transitions: Taking advantage of the re-planning capability of the framework, unplanned stable transitions are possible between the 3 different gaits. A plot of the base height and pitch of the base along with the solve times are shown in Fig. 7. The robot is initially in a brief trot phase depicted by the almost constant pitch and base height. After which, the robot transitions to the jumping gait at around 1 second, as can be seen from the large amplitude oscillations in the base height. After about 5 seconds the robot transitions back to the trot gait for about 2 seconds. Finally, the robot moves to the bounding gait between 8 to 12 seconds as can be observed with the large changes in the pitch of the robot. All these transitions happen when the user desires these changes without any pre-planning between gaits changes. When the gait change is triggered, the gait parameters (Table I) and contact plan (discussed in the previous subsection) are changed to the new gait. The BiConMP then generates a stable motion to transition to the desired gait. During all these transitions the framework keeps its solve times below 50 milliseconds. Also, the solve times for each gait remain similar when there are no transitions (Table. II).

D. Acylic motions

To demonstrate the capability of the BiconMP to generate dynamic trajectories beyond mere gaited locomotion, we perform a high-five motion on Solo12. The goal for the robot is to give a high-five to a person in front of it by first raising both its front legs at a height above its base. Then, the robot must reach one of its arms out forward to high-five at a fixed position. Figure 8 shows the high-five motion generated on the robot. Since Solo12 must balance on its hind legs for the duration of the high-five, the motion needs to build significant momentum and is non-trivial to achieve. The motion planner initially gets the robot to crouch before the front two legs lift off to generate enough angular momentum. After the lift off, the front two legs try to reach the goal position provided by the user in the plan. The swing foot trajectory of the front left leg is shown in Fig. 10. The red dots in the top three sub-plots denote the desired goal position provided by the user in the x, y, and z axis. The shaded section of the plots represents the duration of the motion where the front leg is supposed to be in an air phase trying to reach the desired goal, which can be verified by noticing that the planned z force becomes zero in this zone.

Note that during this motion, the kinematics solver finds a non-trivial rotating motion for the front legs after performing the high-five to track the centroidal momentum trajectories provided by the biconvex centroidal dynamics solver. Specifically, the front legs of the robot swing their legs backward and pivot around the hip joint to obey the momentum profile. Such a swing foot trajectory is very difficult to design a priori for a given motion which highlights the advantages of using a whole-body motion optimizer.

E. Simulations with other robots

We illustrate the generality of our approach by testing it on two other robots with different morphology, complexity, and mass distributions. In all these experiments, no changes are made to the BiConMP framework (including the use of the ID controller). We run these experiments in simulation. The solve times obtained for these different motions as they are run in MPC are shown in table III. The resulting movements are also shown in the attached video.

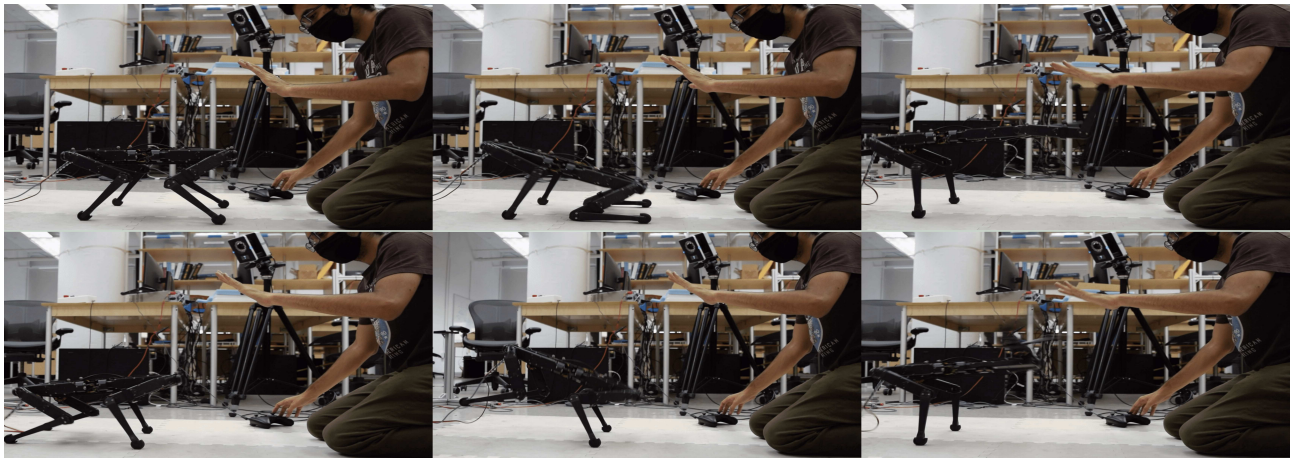

Fig. 8: High five motion (top left to bottom left in a clockwise direction).

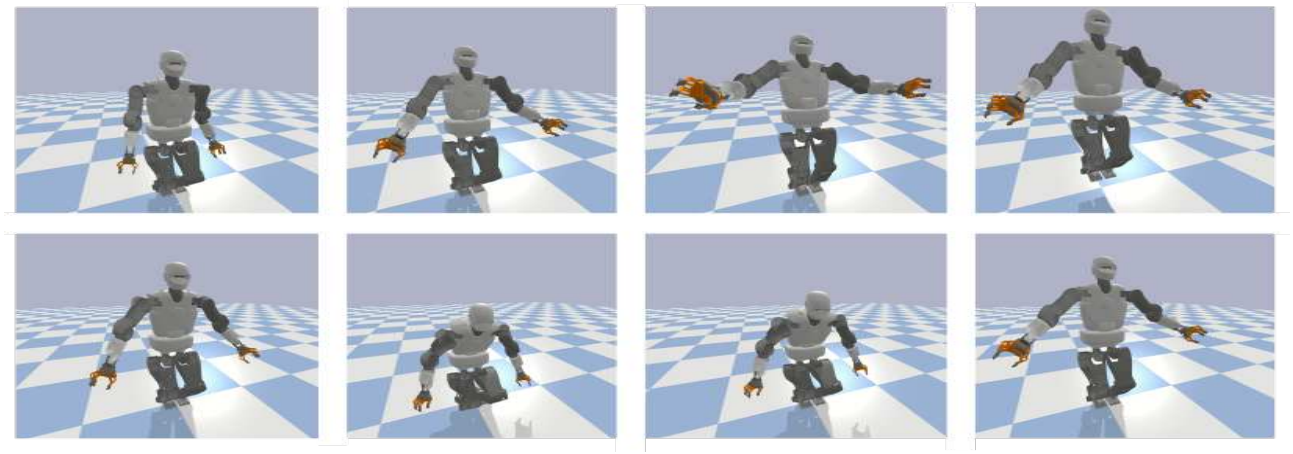

Fig. 9: Jump motion with Talos (Top left to bottom left in a clockwise direction).

1) Talos: - The humanoid robot Talos [38] weighs around 94 kg, stands at 170 cm, and has 35 joints, i.e. it is larger, heavier, and more complex than Solo12. Firstly, we validate the approach by generating stable walking in MPC. Only a reference CoM height, a Raibert controller based contact plan at $0.3s$ of step time, and a nominal mid-swing phase via points are provided to our optimizer. Note that in this case the MPC loop is run at 10Hz as the mean solve time slightly exceeds 50ms. This is because we optimize full-body motions with 35 joints now (instead of 12 for Solo12). Secondly, we generate a jump, which is a highly dynamic motion, in MPC. The framework is only provided a contact plan which specifies a flight and stance time of 0.4 seconds. The planner then automatically generates the required force profile to be applied on the ground and the whole-body motions that track the momentum profiles stably. This again demonstrates the strength of using a nonlinear kinematics solver to automate swing foot trajectory generation. The resulting motion in simulation is shown in figure 9.

2) AnYmal: - The quadruped robot AnYmal [39] weighs around 30 Kg which is significantly heavier than Solo12. We generate a low stepping frequency trot of 0.4 seconds and cyclic jumping with an air time of 0.3 seconds. The solve times of AnYmal trot are slightly higher as compared to Solo12 since we use more collocation points in the trajectory. We also demonstrate stable unplanned gait transitions just like with Solo12.

***Remark** 4:* Note that we still track these motions on these robots without the need for a complicated QP-based ID controller even though AnYmal and Talos' legs are heavy (i.e. contribute significant momentum) as compared to Solo12. Our framework remains unaffected even with low stepping frequencies (Anymal Trot) and arm swinging motions (Talos Jump).

V. Analysis of the Model Predictive Controller

In this section, we analyze the contribution of the inverse dynamics (RNEA) controller on the stability of the robot and the advantages of using a nonlinear MPC setting on the robot as compared to pure trajectory optimization (when a fixed plan is tracked on the robot). Further, we thoroughly analyze the impact of the horizon and replanning frequency on the performance and robustness of the MPC framework to gain insights into tuning parameters in the case of MPC for

Robot	Motion	Replan Frequency (Hz)	Mean Solve Time (sec)	Number Of Collocation Points	Time Discretization Δt (sec)
Talos	Jump	10	0.031	10	0.04
Talos	Walk	10	0.052	10	0.1
AnYmal	Trot	20	0.042	18	0.05
AnYmal	Jump	20	0.034	10	0.05

TABLE III: Solve times for various motions generated on different robots in simulation.

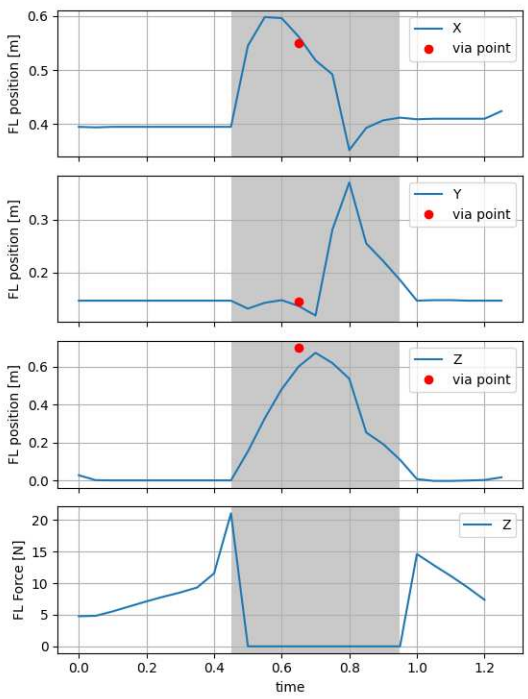

Fig. 10: Swing foot trajectory of the front left leg from the IK.

legged robotics. We also discuss qualitative advantages that were observed on the robot, which are difficult to quantify.

A. Inverse dynamics controller

We compare the contribution of the PD gains with the MPC planned trajectory in the inverse dynamics controller. In Fig. 11, the total torque sent to the robot along with the torque resulting from the PD controller is plotted for three joints belonging to the front left leg during the periodic jumping gait. The shaded sections highlight the contact phases of the leg during the motion. The torques due to the PD controller are very close to zero for most of the contact phase while the total torque is not zero (especially for the knee joint which does most of the work in the jump motion). This shows that the joint torques computed by the BiConMP account for most of the control during contact phases, underlining the quality of the optimized control trajectories. However, the PD controller seems to play a significant role during an air to contact phase transition where there is a sudden change in the planned contact forces. During run time, the optimized forces from the BiConMP are interpolated to match the low-level control frequency (i.e. we have a 20Hz MPC loop and a 1KHz torque control loop). Consequently, the forces are interpolated from zero to the next desired value which also prevents sudden jumps in the commanded torques. Consequently, the motion run on the robot is slightly different from the plan during these transitions. We hypothesize that it is the reason why the contribution of the PD controller seems to be larger during these transitions. We made qualitatively similar observations for other movements and simulation experiments on the other robots (Talos and Anymal).

B. Sim-to-real transfer

Initially, the motions discussed in section IV-C and IV-D were first validated in simulation (Raisim [52]). During this stage, the cost functions (weight tuning of the optimization problem) and gait parameters were altered until a desirable motion was observed in the simulation, after which the motion was tested on the real system. All the transfers from the simulation to the real robot was instantaneous. That is, any motion that was stable in simulation for a given set of weight parameters and gains worked directly on the robot without further tuning. This suggests that the MPC framework is sufficiently robust to model mismatch to ensure direct sim-to-real transfer. Further, it is interesting to note that the same set of K_p gains of 3.0 and K_y gains of 0.05 were used across all the motions presented in the result section for both simulation and real robot experiments with Solo12. In our experience, this is rarely the case when using pure trajectory optimization [21] without online re-planning. Often different gains are needed across different motions in simulation and these gains need to be further adapted on the robot. This is a significant benefit provided by the closed-loop MPC setup as gain tuning on the real robot is often cumbersome.

C. Impact of re-planning frequency on performance

In this second set of experiments, we seek to understand the effect of the re-planning frequency on the performance of the MPC. This is an important factor, albeit seldom analyzed, as this sets minimum requirements for optimizer performances. During this experiment, we choose the trot gait and make Solo12 track a fixed desired velocity trajectory from the same starting point. The desired velocity trajectory is set to 0.5 m/s for a fixed duration of 6 seconds after which the desired velocity is changed briefly to 0 m/s and then finally to -0.5 m/s for 6 seconds. During each run, the robot is kept at the same starting position, the weights, horizon length of the plan, and the gains in the ID controller are all kept constant. Only the re-planning frequency is changed during each run. The mean optimal cost returned by the BiConMP is used as a metric [53], [50] to evaluate the performance of the MPC. We ran this experiment 4 times in which 2

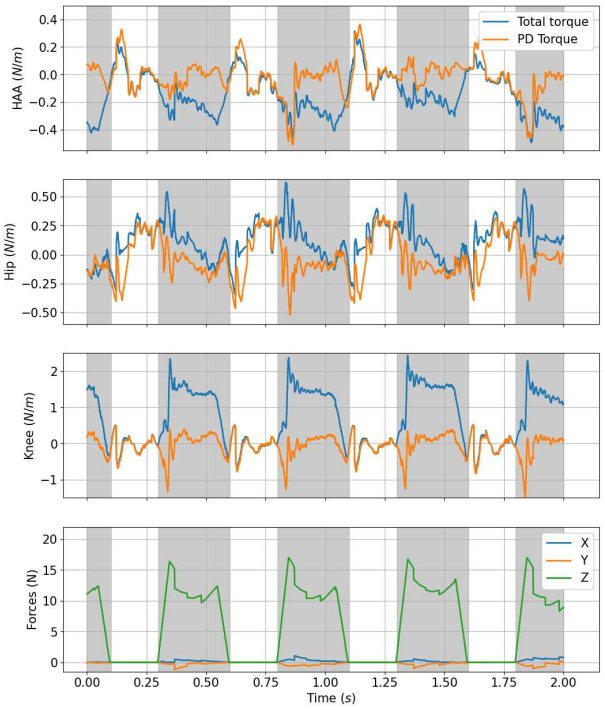

Fig. 11: Contribution of PD gains to the total torques. The three joints are the hip abduction adduction (denoted HAA), the hip flexion-extension (Hip), and the knee joint (Knee).

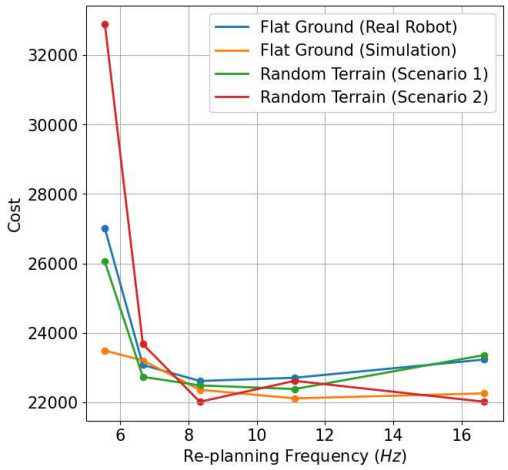

Fig. 12: Comparison of MPC performance vs. replanning frequency. The change in the mean optimal cost returned by the MPC is plotted against replanning frequency.

times Solo12 walks on flat ground while the other two times it walks on random terrain (as shown in Fig. 2). The same experiment is also run on flat ground in simulation for further comparison.

The results obtained from the experiments are shown in Fig. 12. The plot shows that after a certain threshold re-planning frequency is reached the MPC performance does not change. In this case, the threshold re-planning frequency is approximately 7 Hz. Further, this threshold does not change even in the presence of terrain uncertainties. Consequently, this suggests that after reaching the desired threshold

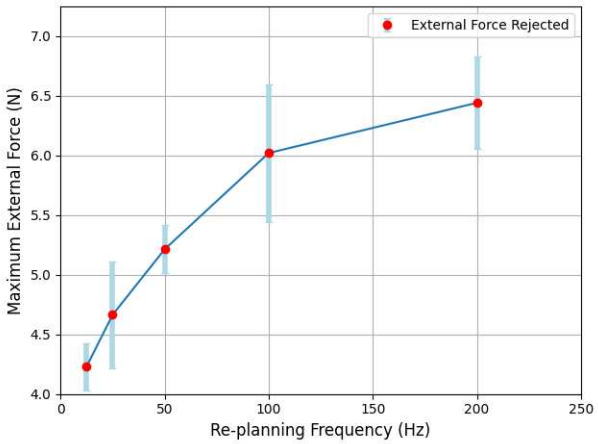

Fig. 13: Comparison of maximum external disturbance vs. re-planning frequency for 3 different motions: trotting, bounding, and jumping. Experiment performed in simulation.

there does not seem to be any benefit, in re-planning much faster with respect to performance. Further, the performance of the MPC in simulation as compared to the real robot is also very similar after this threshold re-planning frequency is reached. This result could also help explain why we were able to directly transfer motions from the simulation to the real robot without any additional changes (i.e. our re-planning frequency was sufficiently high).

D. Impact of planning horizon on performance

Another important parameter in MPC is the length of the optimization horizon. To reduce the computational cost, it needs to be as small as possible but a too short horizon (depending on the choice of terminal cost) will lead to unstable behaviors. In this third set of experiments, we analyze how the length of the planning horizon influences the performance of the MPC. The experimental setup was kept identical to the previous subsection (V-C). The only difference here is that the planning horizon is changed while the re-planning frequency is kept constant. For these experiments, we were unable to determine a good evaluation metric. The cost function can not be used as a metric since the total value attainable by the cost changes with problem size (number of collocation points) which in turn depends on the horizon. Consequently, we only discuss qualitative results observed during the experiments.

For a low re-planning frequency where a stable gait is not observed (frequency lower than 7 Hz, section V-C), we observed that performance/stability could be improved by increasing the length of the planning horizon. After a threshold length is reached, additional increases did not seem to bring any visible benefits. In the presence of terrain uncertainty, a similar result was observed. Further, the threshold planning horizon for a given re-planning frequency remained the same with or without terrain uncertainty. Consequently, we found that at low re-planning frequency, the stability of the motion could be increased by increasing the horizon. This result is in alignment with well-established theoretical results [50].

E. Impact of re-planning frequency on robustness

In the final set of experiments, we analyze how overall motion robustness increases as the re-planning frequency increases. Here we choose to specifically evaluate robustness to external disturbances. Since measuring during locomotion, in a reproducible manner, the maximum magnitude of disturbance rejection is difficult on the real system, we perform this experiment in simulation. Moreover, we test the robustness at very high frequencies which are not realizable in the real world (solve times are higher than the re-planning time). However, our previous experiments suggest that results from the simulation should carry to the real robot. In each run, an external force is applied on the base in a random direction discretized at intervals between 0 and 360 degrees. We push the robot for a duration of 0.2 to 0.5 seconds. We run each experiment 10 times for each duration and run the experiments on three different gaits (trotting, bounding, and jumping.

The results are shown in Fig. 13. This experiment shows that there is a gain in robustness, in terms of disturbance rejection when moving from 20 Hz to 100 Hz, after which the relative gain starts to decay. Based on the previous experiment (Sim-to-Real and MPC performance), a similar result could be expected on the real system provided the solver was 4-5 times faster as the solution quality of the MPC is almost the same in simulation and real robot beyond a re-planning frequency of at least 10 Hz.

VI. Discussion

This section discusses the proposed approach and the experimental results with respect to the current state of the art.

1) Algorithms for closed-loop whole-Body MPC: Very few algorithms have demonstrated closed-loop whole-body MPC on real legged robots. General purpose interior-point or sequential quadratic programming (SQP) methods are not capable of providing solve times low enough for real-time control. Most of the existing approaches use custom DDP-like methods to solve the entire whole-body optimization problem at once [12], [54] or decompose it [26]. These approaches have demonstrated closed-loop MPC ranging from $20 - 80$ Hz for particular motions like trot and jumps. However, these algorithms have seldom been used to show diverse and dynamic motions on robots with different anatomy in closed loop. To our knowledge, our method is the first algorithm using ADMM and first-order proximal methods to demonstrate closed-loop MPC at competitive rates.

The advantage of using DDP-like methods is that they can in principle solve any OCP and efficiently exploit the time-induced sparsity in the problem. DDP further provides optimal feedback gains that can be used for local high-frequency control. However, enforcing constraints becomes challenging (typically hard constraints are not enforced in reported MPC results). The most common practice of using log-barriers to enforce constraints can be numerically problematic [32] due to ill-conditioning of the Hessian. To address this issue, constraints can be relaxed in order to find reasonable solutions [32]. Recently, DDP-based algorithms that enforce constraints directly with low solve times have been proposed [55]. However, they are yet to be demonstrated on high-dimensional problems in MPC.

On the other hand, the BiConMP does not solve general optimal control problems and largely relies on the structure of the floating-based dynamics (kino-dynamic decomposition and biconvex structure of the centroidal dynamics). This has the advantage that constraint enforcement, especially on contact forces, becomes rather straightforward due to the use of the proximal operators (including second-order cones) without any loss in the solve times or convergence rates. Furthermore, problems related to Hessian ill-conditioning inherent to second-order methods are removed with the use of FISTA (a first-order method). While we exploited the structure of the centroidal dynamic, we still use DDP to solve the kinematic optimization problem. It might be interesting to explore whether this problem can be further decomposed to exploit proximal methods and potentially improve efficiency while also including hard constraints. Force constraint enforcement plays an important role (as already highlighted in [56], [57]) and our results suggest that this helps find control trajectories that can be directly executed on the robot with a simple inverse dynamics controller, without the need for an additional dynamic filter (as a QP-based inverse dynamics) unlike DDP-based approaches [26].

In general, existing DDP-based approaches need a good warm start trajectory [54], [48], [12] to actually achieve high replanning rates. Indeed, quadratic convergence rates are only guaranteed close to a local minimum and DDP can be quite slow away from it. Obtaining these trajectories is, however, a challenge that still limits the applicability of DDP-based whole-body solvers. In contrast, we did not encounter the need for good initialization in any of our experiments. The ability of ADMM to converge to good solutions quickly allows early termination of the solver. This property is favorable in closed-loop MPC settings with real-time requirements and is not present with other second-order methods. This property could also be exploited for other problems with closed-loop MPC settings.

2) Advantages of proximal methods for MPC: The use of first-order optimization methods for MPC is not common in robotics, nor is the use of general proximal methods. We believe that beyond the legged robots, our work highlights a few interesting properties more broadly applicable. First, algorithms such as FISTA are very easy to implement (they only need gradients) and are numerically robust. Since sensor noise can limit the availability of true gradients, first-order methods are more likely to converge to good solutions compared to second-order methods [33], [58]. Indeed, we observed that the algorithm was surprisingly stable in a closed loop MPC setting. Further, the use of proximal operators renders the constraint satisfaction problem rather easy. This suggests that first-order methods might play an increasing role for MPC solvers in robotics. Furthermore, developing

custom solvers based on proximal operators and related augmented Lagrangian formulations [55], [59] for other closed loop application is also a promising research direction, especially when early termination is more important than very high precision.

3) Enforcing torque constraints: The main assumption in the kino-dynamic decomposition of the nonlinear robot dynamics is that there exists sufficient torque authority [20], without which computing feasible torques become impossible (or needs several kino-dynamic iterations) for the given plan. During our experiments on the robot, this assumption has never been violated for all the motions even though only one dynamic to kinematic iteration is performed. Further, the computed torques have been much lower than the maximum torque limits of Solo12 and subsequently, more aggressive behaviors can be performed if needed. In the case that this limit is being reached, more than one kino-dynamic iteration can be performed to ensure better consensus. It is also possible to add torque constraints in the kinematic optimizer at the cost of slightly higher solve times as discussed in previous work [21].

4) Insights from Nonlinear MPC Implementation: Running the nonlinear whole body MPC has shown several advantages on the robot along with a few key insights: re-planning online in general improves the robustness of the robot to disturbances and terrain. In addition, the whole body optimization allows the robot to automatically change swing foot trajectories without highly specified references. An interesting result from our analysis is that increasing the re-planning frequency or horizon above a certain threshold does not seem to give any major advantages in terms of performance for the tasks we analyzed. However, there is a significant improvement in robustness as the frequency is increased from 20 Hz to 100 Hz after which the rate of gain starts to decrease. Consequently, this analysis suggests that re-planning frequencies higher than 10 Hz are not needed to achieve direct sim to real transfer for the Solo12. However, to gain more robustness to external disturbances, higher frequencies are needed and avenues such as warm starting the solver, optimizing the implementation, or further exploiting the problem structure can be explored.

5) Comparison to Deep Reinforcement Learning (DRL): Recently, DRL has become an increasingly popular choice to generate robust trajectories for legged robots [60], [61]. One main reason stems from the fact that MPC approaches need fast optimizers while DRL approaches learn a policy offline which is rather cheap to evaluate online. However, our proposed method has a re-planning frequency comparable to these methods, with the scope of further speed up in future work. In addition, generating new trajectories with an optimizer is significantly less cumbersome and does not require one to re-train a policy for different types of motions. In our case, the same cost function with different weights can be used to generate different motions. Finally, the sim-2-real transfer is simple and instantaneous in our approach as the BiConMP is able to compensate for modeling errors automatically thanks to the closed-loop optimization. On the other hand, sim-2-real transfer with DRL methods is usually not simple, because they depend heavily on the trained robot model. Subsequently, they require very accurate robot actuator models [52] in simulation, or domain randomization is necessary [61] for successful transfer.

Conclusion

We proposed a nonlinear MPC framework, the BiConMP, capable of generating dynamic behaviors in real-time for various legged robots. We exploit the biconvex nature of the centroidal dynamics to propose an efficient solver based on ADMM and proximal gradient methods. We further propose to formulate the kinematic problem as an optimal control problem which is then solved using off-the-shelf DDP solvers [35]. Through various real robot and simulation experiments we demonstrated the ability of the approach to generate and control very dynamic movements. We conducted an extensive analysis of the various parameters of the MPC framework such as frequency, cost, and horizon to understand their impact on the performance and robustness on the real robot, hence suggesting general guidelines for MPC requirements. In future work, we intend to investigate the effect of warm-start on the solver efficiency. We also intend to further explore the capabilities of first order proximal methods for more general MPC applications to robotics.

References

[1] 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 *2003 IEEE International Conference on Robotics and Automation (Cat. No. 03CH37422)*, vol. 2. IEEE, 2003, pp. 1620–1626.

[2] P.-B. Wieber, "Trajectory free linear model predictive control for stable walking in the presence of strong perturbations," in *2006 6th IEEE-RAS International Conference on Humanoid Robots*. IEEE, 2006, pp. 137–142.

[3] J. Nocedal and S. Wright, *Numerical optimization*. Springer Science & Business Media, 2006.

[4] 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.

[5] M. Khadiv, A. Herzog, S. A. A. Moosavian, and L. Righetti, "Step timing adjustment: A step toward generating robust gaits," in *2016 IEEE-RAS 16th International Conference on Humanoid Robots (Humanoids)*. IEEE, 2016, pp. 35–42.

[6] M. A. Hopkins, D. W. Hong, and A. Leonessa, "Humanoid locomotion on uneven terrain using the time-varying divergent component of motion," in *2014 IEEE-RAS International Conference on Humanoid Robots*, 2014, pp. 266–272.

[7] I. Mordatch, E. Todorov, and Z. Popović, "Discovery of complex behaviors through contact-invariant optimization," *ACM Transactions on Graphics (TOG)*, vol. 31, no. 4, pp. 1–8, 2012.

[8] S. Lengagne, J. Vaillant, E. Yoshida, and A. Kheddar, "Generation of whole-body optimal dynamic multi-contact motions," *The International Journal of Robotics Research*, vol. 32, no. 9-10, pp. 1104–1119, 2013.

[9] 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.

[10] 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.

[11] Y. Tassa, T. Erez, and E. Todorov, "Synthesis and stabilization of complex behaviors through online trajectory optimization. in 2012 ieee," in *RSJ International Conference on Intelligent Robots and Systems*, pp. 4906–4913.

[12] M. Neunert, M. Stäuble, M. Giftthaler, C. D. Bellicoso, J. Carius, C. Gehring, M. Hutter, and J. Buchli, "Whole-body nonlinear model predictive control through contacts for quadrupeds," *IEEE Robotics and Automation Letters*, vol. 3, no. 3, pp. 1458–1465, 2018.

[13] R. Deits and R. Tedrake, "Footstep planning on uneven terrain with mixed-integer convex optimization," in *2014 IEEE-RAS international conference on humanoid robots*. IEEE, 2014, pp. 279–286.

[14] B. Ponton, A. Herzog, S. Schaal, and L. Righetti, "A convex model of humanoid momentum dynamics for multi-contact motion generation," in *2016 IEEE-RAS 16th International Conference on Humanoid Robots (Humanoids)*. IEEE, 2016, pp. 842–849.

[15] 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 *2020 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 2020, pp. 6604–6610.

[16] Y.-C. Lin, B. Ponton, L. Righetti, and D. Berenson, "Efficient humanoid contact planning using learned centroidal dynamics prediction," in *2019 International Conference on Robotics and Automation (ICRA)*. IEEE, 2019, pp. 5280–5286.

[17] S. Tonneau, A. Del Prete, J. Pettré, C. Park, D. Manocha, and N. Mansard, "An efficient acyclic contact planner for multiped robots," *IEEE Transactions on Robotics*, vol. 34, no. 3, pp. 586–601, 2018.

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

[19] H. Dai, A. Valenzuela, and R. Tedrake, "Whole-body motion planning with centroidal dynamics and full kinematics," in *2014 IEEE-RAS International Conference on Humanoid Robots*. IEEE, 2014, pp. 295–302.

[20] A. Herzog, S. Schaal, and L. Righetti, "Structured contact force optimization for kino-dynamic motion generation," in *2016 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE, 2016, pp. 2703–2710.

[21] 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*, 2021.

[22] F. Alizadeh and D. Goldfarb, "Second-order cone programming," *Mathematical programming*, vol. 95, no. 1, pp. 3–51, 2003.

[23] H. Dai and R. Tedrake, "Planning robust walking motion on uneven terrain via convex optimization," in *2016 IEEE-RAS 16th International Conference on Humanoid Robots (Humanoids)*. IEEE, 2016, pp. 579–586.

[24] J. Di Carlo, P. M. Wensing, B. Katz, G. Bledt, and S. Kim, "Dynamic locomotion in the mit cheetah 3 through convex model-predictive control. in 2018 ieee," in *RSJ International Conference on Intelligent Robots and Systems (IROS)*, pp. 1–9.

[25] Y. Ding, A. G. Pandala, C. Li, Y.-H. Shin, and H.-W. Park, "Representation-free model predictive control for dynamic motions in quadrupeds," *IEEE Transactions on Robotics*, vol. 37, pp. 1154–1171, 2021.

[26] J.-P. Sleiman, F. Farshidian, M. V. Minniti, and M. Hutter, "A unified mpc framework for whole-body dynamic locomotion and manipulation," *IEEE Robotics and Automation Letters*, vol. 6, no. 3, pp. 4688–4695, 2021.

[27] K. Yamane and Y. Nakamura, "Dynamics filter-concept and implementation of online motion generator for human figures," *IEEE transactions on robotics and automation*, vol. 19, no. 3, pp. 421–432, 2003.

[28] P. Shah, A. Meduri, W. Merkt, M. Khadiv, I. Havoutis, and L. Righetti, "Rapid convex optimization of centroidal dynamics using block coordinate descent," in *2021 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE, 2021, pp. 1658–1665.

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

[30] S. Boyd, N. Parikh, E. Chu, B. Peleato, and J. Eckstein, "Distributed optimization and statistical learning via the alternating direction method of multipliers," *Foundations and Trends® in Machine learning*, vol. 3, no. 1, pp. 1–122, 2011.

[31] A. Beck, *First-order methods in optimization*. SIAM, 2017.

[32] R. Grandia, F. Farshidian, R. Ranftl, and M. Hutter, "Feedback mpc for torque-controlled legged robots," in *2019 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE, 2019, pp. 4730–4737.

[33] H. Mohammadi, M. Razaviyayn, and M. R. Jovanović, "Robustness of accelerated first-order algorithms for strongly convex optimization problems," *IEEE Transactions on Automatic Control*, vol. 66, no. 6, pp. 2480–2495, 2020.

[34] J. Liang, J. Fadili, and G. Peyré, "Activity identification and local linear convergence of forward–backward-type methods," *SIAM Journal on Optimization*, vol. 27, no. 1, pp. 408–437, 2017.

[35] 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 *2020 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 2020, pp. 2536–2542.

[36] M. Khadiv, A. Herzog, S. A. A. Moosavian, and L. Righetti, "Walking control based on step timing adaptation," *IEEE Transactions on Robotics*, 2020.

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

[38] O. Stasse, T. Flayols, R. Budhiraja, K. Giraud-Esclasse, J. Carpentier, J. Mirabel, A. Del Prete, P. Souères, N. Mansard, F. Lamiraux, *et al.*, "Talos: A new humanoid research platform targeted for industrial applications," in *2017 IEEE-RAS 17th International Conference on Humanoid Robotics (Humanoids)*. IEEE, 2017, pp. 689–695.

[39] M. Hutter, C. Gehring, D. Jud, A. Lauber, C. D. Bellicoso, V. Tsounis, J. Hwangbo, K. Bodie, P. Fankhauser, M. Bloesch, *et al.*, "Anymal-a highly mobile and dynamic quadrupedal robot," in *2016 IEEE/RSJ international conference on intelligent robots and systems (IROS)*. IEEE, 2016, pp. 38–44.

[40] D. E. Orin, A. Goswami, and S.-H. Lee, "Centroidal dynamics of a humanoid robot," *Autonomous robots*, vol. 35, no. 2, pp. 161–176, 2013.

[41] R. Featherstone, *Rigid body dynamics algorithms*. Springer, 2014.

[42] A. Beck and M. Teboulle, "A fast iterative shrinkage-thresholding algorithm for linear inverse problems," *SIAM journal on imaging sciences*, vol. 2, no. 1, pp. 183–202, 2009.

[43] H. H. Bauschke, "Projection algorithms and monotone operators," Ph.D. dissertation, Theses (Dept. of Mathematics and Statistics)/Simon Fraser University, 1996.

[44] J. DE O. PANTOJA, "Differential dynamic programming and newton's method," *International Journal of Control*, vol. 47, no. 5, pp. 1539–1553, 1988.

[45] D. Murray and S. Yakowitz, "Differential dynamic programming and newton's method for discrete optimal control problems," *Journal of Optimization Theory and Applications*, vol. 43, no. 3, pp. 395–414, 1984.

[46] D. Kim, J. Di Carlo, B. Katz, G. Bledt, and S. Kim, "Highly dynamic quadruped locomotion via whole-body impulse control and model predictive control," *arXiv preprint arXiv:1909.06586*, 2019.

[47] R. Budhiraja, J. Carpentier, and N. Mansard, "Dynamics consensus between centroidal and whole-body models for locomotion of legged robots," in *2019 International Conference on Robotics and Automation (ICRA)*. IEEE, 2019, pp. 6727–6733.

[48] T. S. Lembono, C. Mastalli, P. Fernbach, N. Mansard, and S. Calinon, "Learning how to walk: Warm-starting optimal control solver with memory of motion," in *2020 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 2020, pp. 1357–1363.

[49] P.-B. Wieber, R. Tedrake, and S. Kuindersma, "Modeling and control of legged robots," in *Springer handbook of robotics*. Springer, 2016, pp. 1203–1234.

[50] D. Q. Mayne, "Model predictive control: Recent developments and future promise," *Automatica*, vol. 50, no. 12, pp. 2967–2986, 2014.

[51] J. Koenemann, A. Del Prete, Y. Tassa, E. Todorov, O. Stasse, M. Bennewitz, and N. Mansard, "Whole-body model-predictive control applied to the hrp-2 humanoid," in *2015 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE, 2015, pp. 3346–3351.

[52] 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.

[53] L. Grüne, "Nmpc without terminal constraints," *IFAC Proceedings Volumes*, vol. 45, no. 17, pp. 1–13, 2012.

[54] E. Dantec, R. Budhiraja, A. Roig, T. Lembono, G. Saurel, O. Stasse, P. Fernbach, S. Tonneau, S. Vijayakumar, S. Calinon, *et al.*, "Whole

body model predictive control with a memory of motion: Experiments on a torque-controlled talos," in *2021 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 2021, pp. 8202–8208.

[55] W. Jallet, N. Mansard, and J. Carpentier, "Implicit differential dynamic programming," in *2022 International Conference on Robotics and Automation (ICRA)*. IEEE, 2022, pp. 1455–1461.

[56] L. Righetti, J. Buchli, M. Mistry, M. Kalakrishnan, and S. Schaal, "Optimal distribution of contact forces with inverse-dynamics control," *The International Journal of Robotics Research*, vol. 32, no. 3, p. 280–298, Mar 2013.

[57] J. Viereck, A. Meduri, and L. Righetti, "Valuenetqp: Learned one-step optimal control for legged locomotion," in *Learning for Dynamics and Control Conference*. PMLR, 2022, pp. 931–942.

[58] J. Liang, T. Luo, and C.-B. Schönlieb, "Improving "fast iterative shrinkage-thresholding algorithm": Faster, smarter, and greedier," *SIAM Journal on Scientific Computing*, vol. 44, no. 3, pp. A1069–A1091, 2022.

[59] B. E. Jackson, T. Punnoose, D. Neamati, K. Tracy, R. Jitosho, and Z. Manchester, "Altro-c: A fast solver for conic model-predictive control," in *2021 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 2021, pp. 7357–7364.

[60] J. Lee, J. Hwangbo, L. Wellhausen, V. Koltun, and M. Hutter, "Learning quadrupedal locomotion over challenging terrain," *Science robotics*, vol. 5, no. 47, 2020.

[61] J. Siekmann, K. Green, J. Warila, A. Fern, and J. Hurst, "Blind bipedal stair traversal via sim-to-real reinforcement learning," in *Robotics: Science and Systems*, 7 2021.