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[1,2]

***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. 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 forces and contact planners which decide which end-effector should make contact along with where and when these forces are to be applied to generate a desired behaviour. Consequently, it is crucial for these algorithms to run as fast as possible such that they adapt online to dynamic changes in the environment.

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 renders the optimization problem with a quadratic cost and linear constraints, the motion planning problem can be solved quickly using a quadratic program (QP) [3]. While 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 slightly more efficient phase-based formulation of contact planning is proposed. Furthermore, [11] makes use of differential dynamic programming (DDP) [12] to solve the motion planning problem through contact. While these approaches can in principle find complex contact sequences, they are computationally too expensive to be used in real-time. Although the authors in [13] showed nonlinear model predictive control with heavy numerical and software engineering on a quadruped using Gauss-Newton multiple shooting and a relaxed spring damper contact model, the results were displayed only for 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 planing and motion planning. 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 [14], [15], L1-loss based optimization [16], and graph search methods [17], [18]. For the motion planning sub-problem, the nonlinear dynamics of the legged robot is split into two components, the actuated and unactuated dynamics (centroidal dynamics) [19]. 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 [20]. In [21], it was shown that a feasible whole body motion plan can be generated more efficiently by iteratively optimizing for the centroidal dynamics and inverse kinematics (IK) problem. Despite splitting the trajectory optimization problem into two parts, each individual sub-problem still remains nonlinear and cannot be solved in real-time.

In order to further reduce the computation times, several relaxations have been proposed to the centroidal dynamics formulation. In [22], sequential convex relaxations are used to tackle the nonlinearity in the dynamics and each relaxation is solved using a second order cone programs [23]. Even though this setup can be used to solve for a variety of motions, the reduction in compute times are not yet sufficient to be used to re-plan online and the relaxations were often found to not be tight enough for dynamic motions. Another approach used to make the centroidal dynamics convex is by only minimizing the worst case L_1 bound on the angular momentum [24]. While this formulation allows 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,

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

the centroidal dynamics can be made convex and solved online by linearizing the base rotation [25], [26]. Common to these convex approaches is that the swing foot trajectories are predefined which restrict the whole-body motion planning problem and may be physically inconsistent. The authors of [27] use DDP to solve an optimization problem with the centroidal dynamics and first order kinematics in real-time. While this approach achieved re-planning frequencies suitable for real-time use when solving the original nonlinear problem, their method utilizes a whole body controller and it is unclear whether or not this whole-body controller explicitly tracks the profiles generated by their method (i.e. whether the trajectories are dynamically feasible at all times). Also, from our experience, the solve times would increase drastically for more dynamic motions such as bounding gaits (which have high angular momentum) or rapid (forward) jumping due to the drastic changes in the momenta. However, it is not clear if this framework would still be able to generate plans for such motions in real-time as this is not discussed.

In this work, we propose a nonlinear trajectory optimization framework that can generate whole body motion plans in real-time using the kino-dynamic structure proposed in [21]. The dynamics optimization problem is solved efficiently by exploiting the biconvex structure in the centroidal dynamics. We already used this structure in our previous work [28] which decomposed the biconvex nature of our problem into two separate, convex, sub-problems. Given the convexity of each sub-problem, an alternating procedure called block coordinate descent was used which allowed the use of state of the art QP solvers and resulted in a speedup in solve times. In this work, we explore a different approach that once again exploits the biconvex nature of the centroidal dynamics using the Alternating Direction Method of Multipliers (ADMM) [29].

Compared to the block coordinate descent algorithm, the ADMM algorithm provides favourable convergence properties such as the ability to reach acceptable solutions in fewer iterations and guaranteed sublinear convergence. 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) [30]. 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 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 Quadratic Program (QP) solvers which often involve expensive matrix inversions.

The nonlinear inverse kinematics (IK) sub-problem is solved using DDP. This 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 [25], [31]. Although these methods of swing-foot generation work well for simple motions, they are often be restrictive in nature and do not allow the algorithm the freedom to find trajectories which may utilize the full capabilities of the end-effectors. The nonlinear IK often generates non-trivial swing foot trajectories to track the desired centroidal momentum provided by the dynamics optimization.

The entire framework is run in an MPC fashion on a real quadruped robot Solo12 [32] to generate several gaits such as trotting, jumping and bounding. In addition, we 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 with the same framework while still re-planning in real-time. Furthermore, the framework can be used for different types of robots such as bipeds and humanoids, as the kino-dynamic formulation has been shown to generate reasonable motions previously [33], [22]. This is often not possible with other frameworks due to their use of simplified dynamics whose assumptions may not hold valid [25]. Finally, we extensively analyze the empirical effects of horizon length and re-planning frequency on the robustness and performance of the nonlinear MPC on the real robot. To the best of our knowledge, this is the first reported empirical analysis of closed loop non linear MPC for legged robots.

A. Contributions

The main contributions of this work are as follows:

- We propose a novel solver based on ADMM that takes advantage of the biconvex structure of the centroidal dynamics and generates centroidal trajectories efficiently. We also propose to use an efficient method based on proximal operators to solve the convex sub-problems (with second-order cone constraints) which ultimately enables us to perform centroidal MPC.
- To obtain whole-body trajectories, we propose to use a DDP-based kinematic solver (rather than whole-body MPC) to track the centroidal trajectories while penalizing other whole-body constraints. Given the desired forces from the centroidal MPC and joint trajectories from DDP-based kinematic solver, we compute the joint torques through an unconstrained inverse dynamics. This whole framework enable us to perform 20 Hz whole-body MPC on a quadruped robot where we send directly the computed joint torques to the robot without having a constrained QP-based inverse dynamics on top of the MPC.
- We implement different cyclic and acyclic gaits on the real Solo12 quadruped along with the transition between these gaits. In particular, we perform a bounding motion on Solo12 with a considerable amount of angular momentum which is not possible to do with state-of-the-art linear MPC approaches.
- We perform extensive analysis on the effects of horizon length and MPC frequency on the performance and robustness of the whole control framework in the presence and absence of different uncertainties and disturbances.

II. BACKGROUND

In this section, we review the necessary background to describe our proposed motion planner, BiConMP. 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 introduce two optimization techniques used in the solver briefly.

A. Centroidal Dynamics

The rigid body dynamics of any robot can be described as follows:

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

where q represents the generalized joint configuration of the robot, and v is the generalized velocity vector. $M(q)$ is the mass matrix for the given robot configuration, $N(q, v)$ is Coriolis matrix which consist of the nonlinear effects of the dynamics, τ is the vector of joint torques, S is the selection matrix that define the underactuation of the robot, J_j are the end effector Jacobians and $\lambda = [f, \tau]$ is the vector of forces and torques applied at each end effector.

In the case of a floating base robot, the dynamics can be split into two parts - the actuated and unactuated part [19], [21] as shown below

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

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

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)

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

where l, k are the linear and angular momentum [34], m is the robot mass, g is the gravity vector, c represents the center of mass location, n_j is a binary integer that describes whether the end effector j is in contact, f_j, τ_j, r_j are the end effector force, torque and location respectively. The linear and angular momentum can also be described in terms of the generalized joint configuration using the centroidal momentum matrix $A(q)$ of the robot [34] as shown below

$$\begin{bmatrix} \mathbf{l} \\ \mathbf{k} \end{bmatrix} = \mathbf{A}(\mathbf{q})\mathbf{v} \tag{4}$$

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 [21][22]. This is an attractive approach since it breaks the original highly 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)

$$\begin{aligned}
\min_{c,\dot{c},k,f,\tau} \quad & \sum_{t=0}^{T-1} \phi_r^t(c_t, \dot{c}_t, k_t, f_t, \tau_t) \\
& + \phi^T(c_T, \dot{c}_T, k_T, f_T, \tau_T) \\
\text{s.t.} \quad & c_{t+1} = c_t + \dot{c}_t \Delta t && (5) \\
& \dot{c}_{t+1} = \dot{c}_t + \sum_{j=1}^{N} n_t^j \frac{\mathbf{f}_t^j}{m} \Delta t + g\Delta t && (6) \\
& k_{t+1} = k_t + \sum_{j=1}^{N} n_t^j((r_t^j - c_t) \times \mathbf{f}_t^j + \tau_t^j)\Delta t && (7) \\
& \forall_{t,j},\ \sqrt{(\mathbf{f}_{t,x}^j)^2 + (\mathbf{f}_{t,y}^j)^2} \leq \mu \mathbf{f}_{t,z}^j \\
& \forall_{t,j},\ \mathbf{f}_{t,z}^j \geq 0 && (8) \\
& \forall_{t,j},\ r_t^j \epsilon \Psi \\
& \forall c_t \in \Omega && (9) \\
& c_0, \dot{c}_0 = c_{init}, \dot{c}_{init}
\end{aligned}$$

where $\phi_r^t(c_t, \dot{c}_t, k_t, f_t, \tau_t)$ is the running cost, $\phi^T(c_T, \dot{c_T}, k_T, f_T, \tau_T)$ 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 the kinematic constraints, $c_{init}, \dot{c}_{init}$ are the initial conditions of the robot center of mass (CoM).

The optimal joint trajectory is generated by solving an inverse kinematics (IK) problem which tracks the optimal centroidal momentum obtained from the previous step using the centroidal momentum matrix, along with additional full body tasks [22]. The generated momentum trajectory from the inverse kinematics problem is then used as a soft constraint in the centroidal OCP to obtain a refined centroidal trajectory and forces. This process is iterated until the two sub-problems converge [21].

Finally, inverse dynamics is used to map state trajectories and contact forces to actuated joint torques as shown below:

$$\tau_{RNEA} = M_a(q)\dot{v} + N_a(q, v) - \sum_{j=1}^{N} J_{a,j}^T \lambda_j \tag{10}$$

Note that we do not use a constrained QP-based inverse dynamics, but we simply use the joint positions, velocities, and force trajectories within the Recursive Newton Euler Algorithm (RNEA) [35] to compute the torques.

C. Alternating Direction Method of Multipliers

The Alternating Direction Method of Multipliers (ADMM) algorithm [29] is designed to solve problems of the form

$$\begin{aligned} \min_{x,z} \quad & N(x) + G(z) \\ \text{s.t.} \quad & Ax + Bz = d \end{aligned}$$

where $N(x)$, $G(z)$ are cost functions, A and B are matrices while c is a vector of constraints. The ADMM method, as the name suggests, alternatively minimizes for x and z until convergence. The algorithm has an interesting property that it can reach reasonably good solutions in a few iteration and then take several more iterations to provide high quality solutions. Consequently, this can be exploited in a MPC setup where a new trajectory is needed in within a fixed amount of time (real-time).

The ADMM algorithm can also be used to efficiently solve problems of the form

$$\begin{aligned} \min_{x,z} \quad & N(x,z) \\ \text{s.t.} \quad & G(x,z) = 0 \end{aligned}$$

where the cost function $N(x,z)$ and constraints $G(x,z)$ are biconvex in terms of x,z. The ADMM algorithm minimizes the problem efficiently by iteratively optimizing for x keeping z constant and vice versa. More details can be found in [29]. In this paper, we use the ADMM algorithm to efficiently solve the centroidal OCP in Section III-B.

D. Fast Iterative Shrinkage Thresolding Algorithm

Proximal gradient methods are a popular family of algorithms used to solve problems of the form

$$\min_{x} \quad F(x) + I(x)$$

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

$$x_{k+1} = P_c(x_k + t_k \nabla f(x_k)) \tag{11}$$

where t_k is the step length, 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 x_{k+1} lies within the domain of $I(x)$ [36]. The proximal operator has a closed form solution and is usually very cheap to evaluate. On the other hand, the function $I(x)$ can only be used if it is possible to compute the proximal operator for it [30]. Consequently, arbitrary inequality constraints cannot be enforced with these methods.

In the specific case when the cost function $F(x)$ 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 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 [36], [30].

Algorithm 1: FISTA algorithm

Initialize optimization variables: $y_0 = x_0, t_0 = 1$
set $k = 0$
while $k <$ *maximum iterations* **do**
  Pick $L_k > 0$
  $x_{k+1} = prox_{\frac{1}{L_k} I}(y_k + \frac{1}{L_k} \nabla f(y_k))$
  $t_{k+1} = \frac{1+\sqrt{1+4t_k^2}}{2}$
  $y_{k+1} = x_{k+1} + \frac{t_k - 1}{t_{k+1}}(x_{k+1} - x_k)$

In this work we use FISTA to exploit its 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 x (inequality constraints) is inexpensive.

III. Approach

In this section, we introduce the various 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 IK 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 for a robot.

A. Biconvexity in Centroidal Dynamics

The unactuated dynamics constraints (5), (6), (7) are non-linear due to the cross product term in the angular momentum constraint (7). This non-convexity makes the problem inherently difficult to solve. These constraints, however, have an interesting feature: they are biconvex (section II-C). That is, the constraints are affine in terms of $c, \dot{c}, k$ when F is kept constant and vice-versa. Consequently, the terms of the discrete constraints (5), (6), (7) can be rearranged as an affine equation in terms of X, $A(F)X = b(F)$ and F, $A(X)F = b(X)$, where $X = \{c_t, \dot{c}_t, k_t ....\}$ and $F = \{F^t_{j,x}, F^t_{j,y}, F^t_{j,z} ...\}$, for $t = 0, ..., n$. Here $A(F)$ is a matrix whose elements depend on F and the centroidal dynamic constraints. Similarly, $A(X)$ is a matrix depending on X and dynamics constraints. $b(F)$

and $b(X)$ are vectors whose elements depend on F and X respectively.

B. Biconvex optimization with ADMM

The centroidal dynamics OCP in section II-B can be formulated alternatively as shown below:

$$\begin{aligned} \min_{X,F} \quad & \Phi(X) + I(X) + \Phi(F) + I(F) \\ \text{s.t.} \quad & G(X, Z) = 0 \end{aligned}$$

where $\Phi(X), \Phi(F)$ are the running and terminal cost functions in terms of X and F respectively, $G(X, F) = 0$ are the nonlinear constraints (5), (6), (7) and the initial state constraints ($c_0, \dot{c}_0 = c_{init}, \dot{c}_{init}$), $I(X)$ is an indicator function that enforces kinematic constraints ($\forall c_t \in \Omega$, section II-B) while $I(F)$ 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. The proposed BiConMP does exactly this in the dynamics optimization by iteratively solving the two convex sub-problems (shown in Algorithm 2) as a part of a larger ADMM optimization scheme (section II-C). The ADMM algorithm solves both the convex sub-problems (force (F) problem and state (X) problem) iteratively until the dynamics violation falls below a desired tolerance (exit criteria). The dynamics violation is computed as shown below

$$||A(F_{k+1})X_{k+1} - b(F_{k+1})||^2 \leq \epsilon_{dyn}$$

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. Φ(F) + ρ||A(X_k)F − b(X_k) + P_k|| + I(F)
     F
    min. Φ(X) + ρ||A(F_{k+1})X − b(F_{k+1}) + P_k|| + I(X)
     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 ≤ ε_dyn then
        terminate
```

The cost function in the state sub-problem is always of the form

$$\Phi(X) = (X - X_{nom})^T W_x (X - X_{nom})$$

where X_{nom} is a nominal trajectory, 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 a 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 a desired base orientation since direct orientation tracking is not possible with the centroidal OCP formulation. The nominal angular momentum trajectory is computed as follows:

$$k_{nom} = w log_3(q_0 \ominus q_{des}) \tag{12}$$

where q_0 and q_{des} are the current and desired base orientation in quaternions. $\ominus$ is the difference operator for quaternions and the log_3 is the logarithmic operator for the SE3 to se3 manifold. w is the 3 dimensional weight vector. This 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 is always set to $[0, 0, 0, 1]$ which corresponds to zero roll, pitch and yaw. Even though the computed nominal angular momentum ignores the inertia of the base (assumes unit inertia), the resulting trajectories from BiConMP are able to track a desired orientation (even during external pushes) on the robot.

In the case of the force sub-problem, the cost is

$$\Phi(F) = F^T W_f F$$

which penalizes unnecessary control. 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 matrix $A(X_k)$), 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 two convex sub-problems can be solved using any Quadratic Program (QP) solver [3]. In the BiConMP, the FISTA (section II-D) is used. 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 X): the kinematic constraints are enforced by the indicator function $I(X)$. 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 [30] while computing a descent step in FISTA for the k^{th} iteration, as shown below

$$X_{k+1} = max(min(X_k^*, u), l)$$

where

$$X_k^* = Y_k + \frac{1}{L_k}(\Phi^{'}(Y_k) + \rho A(F_k)^T (A(F_k)Y_k - b(F_k) + P_k)$$

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

2) Force sub-problem (optimizing for F): the indicator function $I(F)$ enforces second order friction cone constraints [37]. 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

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

where μ is the friction coefficient,

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

and

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

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

$$F_k^* = Y_k + \frac{1}{L_k}(\Phi^{'}(F) + A(X_k)^T\rho(A(X_k)Y_k - b(X_k) + P_k)),$$

every f_x^*, f_y^*, f_z^* in F_k^* is then projected based on the friction cone proximal operator to obtain the force vector F_{k+1} for the next iteration. Here Y_k is the auxiliary variable to F_k (section II-D). The projection of each group of forces independently works mathematically with the FISTA algorithm because the control decision variables are independent [30] of each other in the centroidal problem. There is no explicit constraint enforcing unilaterality in f_z because the friction cone projection implicitly enforce $f_z \geq 0$. 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. With other QP solvers the friction cones are usually approximated as polyhedrons to make them linear. This usually results in conservative trajectory solutions which is 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 is used to compute the descent direction instead of using auto-diff or numerical differentiation methods. Secondly, the sparsity of the matrices $A(X), A(F)$ are exploited during the matrix-matrix and matrix-vector computation in each iteration. Thirdly, the matrix multiplications, such as $A(X_k)^T A(X_k)$, $A(X_F)^T A(X_F)$, 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 certain number of iterations are reached. In practice, we noticed that almost the same step lengths were used in each iteration for the QPs. 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 [30]). In this case, we believe that the empirically determined value satisfies this property. Warm starting the line search would not be possible conveniently with other QP solvers.

***Remark** 1:* 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.

D. Inverse Kinematics

The full-body inverse kinematics problem described in section II-B is also nonlinear in nature. In the BiConMP, the problem is solved quickly using Differential Dynamic Programming (DDP) [12]. DDP exploits the block diagonal structure of the matrices while optimizing the problem and also shows quadratic convergence [38], [39]. We use Crocoddyl [40], an open-source DDP implementation, to solve the inverse kinematics problem in our framework.

We formulate the inverse kinematics formulation as shown below

$$\begin{aligned} \min_{q,v,\dot{v}} \quad & \sum_{t=0}^{T} \Phi_{mom}^t(l_t^*, k_t^*) + \Phi_{CoM}^t(c_t^*) + \Phi_{eff}^t(q_t, v_t) + ||\dot{v}|| \\ \text{s.t.} \quad & q_{t+1} = q_t + v_t\,\Delta t \\ & v_{t+1} = v_t + \dot{v}_t\,\Delta t \end{aligned}$$

where, $\Phi_{mom}^t(l_t^*, k_t^*) = ||A(q_t)v_t - \begin{bmatrix} l_t^* \\ k_t^* \end{bmatrix}||$ is a momentum cost that tracks the optimal linear (l^*) and angular momentum (k^*) computed by the centroidal OCP (Algorithm 2), $\Phi_{CoM}^t(c_t^*)$ is the center of mass tracking cost with the optimal CoM trajectory (c_t^*) obtained from the centroidal OCP, $\Phi_{eff}^t(q_t, v_t)$ is the end effector locations and velocity cost, and $||\dot{v}||$ 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 IK 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 IK that plans a full body motion by taking the future into account.

E. The 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 [41] 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 (10) along with a low joint impedance around the

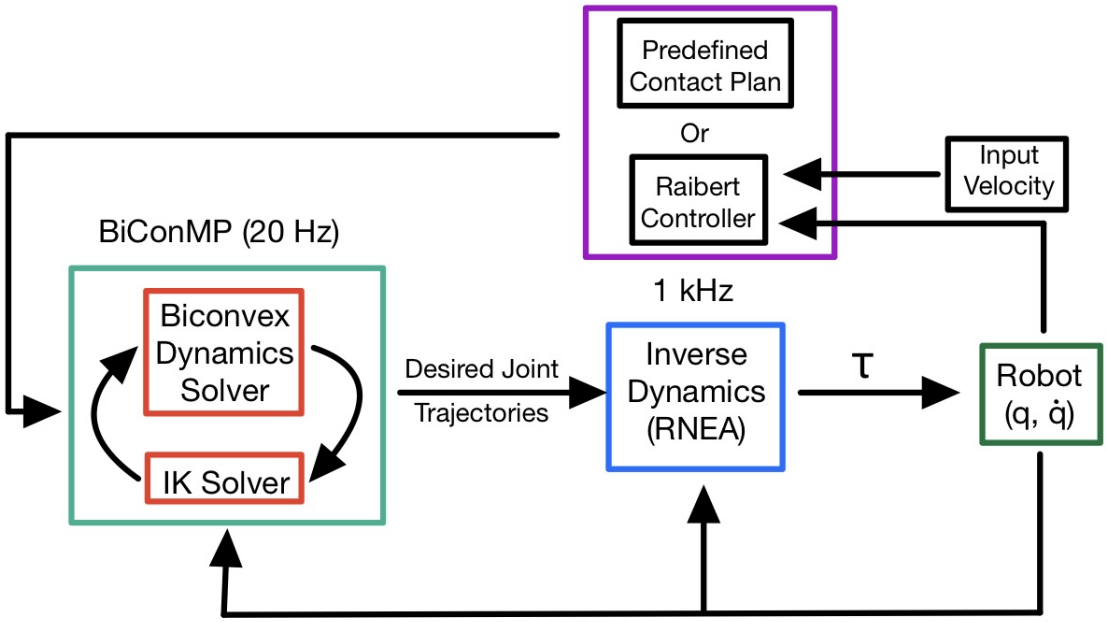

Fig. 1: A birds eye view of the entire nonlinear MPC framework. First, centroidal trajectories are generated using ADMM framework explained in Section III-B. These trajectories are used within a DDP-based kinematic optimizer that generates the desired joint trajectories (Section III-D). The optimal force and joint trajectories from this kino-dynamic iteration are recomputed every 50 ms and are used in an unconstrained inverse dynamics (10) to compute the desired joint torques at 1 KHz. Finally these actuator torques are summed up with a fixed low impedance joint controller that result in the torques sent to the robot actuators (13).

desired states to compute the desired torques at 1 KHz 13. The desired torques is then sent to the robot which are tracked on board at 10 KHz. The entire full body planning loop is re-planned at 20 Hz (50 ms) to update optimal trajectories.

In our BiConMP framework, we re-plan the whole-body trajectories every 50 ms, for a horizon larger then 50 ms, based on feedback from the current state of the robot. The feedforward torques are computed every 1 ms using (10) based on the open loop trajectories in between the two re-planning instances. Finally, a low joint impedance around the desired states are added to the computed torque to result in the final joint torques:

$$\tau_i = \tau_{RNEA,i} + K_p(q_{d,i} - q_{a,i}) + K_d(v_{d,i} - v_{a,i}) \qquad (13)$$

where τ_i is the torque sent to the robot, and $\tau_{RNEA,i}$ is computed using (10) based on the interpolated values of f_i, q_i, v_i, and a_i that are made available through our full-body BiConMP framework. Also, subscripts d and a stand for desired and actual respectively. In the rest of the paper, we refer to the RNEA based controller (13) as the inverse dynamics (ID) controller.

Sometimes the provided contact plan for acyclic motions is longer than the desired horizon the BiConMP uses. In this case, the plan is segmented into a smaller section matching the desired horizon length and then provided to the BiConMP. As time elapses and new plan is required, 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 cyclic 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 the replanning time based on the user input. In total, the contact plan and the corresponding matrices generated in the BiConMP (section III-B) are updated as the time elapses to satisfy the moving horizon.

***Remark** 2:* We would like to emphasize the importance of each individual 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 provides us with quadratic convergence even with second order friction cone constraints. Third, we solve a simple second order inverse kinematics problem given the momentum profiles provided by the centroidal dynamics optimization. This allows us to drastically reduce the computation time compared to the approaches that use full-body DDP for the kinematics problem [42]. Fourth, given the planned forces from centroidal MPC and desired joint trajectories from the IK, we compute the joint torques using (10) without a need to solve a constrained whole-body inverse dynamics. Also we would like to emphasis that contrary to [27], we intentionally did not use a constrained, complicated, inverse dynamics on

top of our MPC block to demonstrate the quality of our plans generated by our approach. Specifically, we note that the outputs (torque references) from our proposed framework can be applied directly to the robot.

IV. Experiments

In this section we present the results obtained with BiConMP framework on Solo12 [32]. We initially discuss the effect on the solve times as the size of optimization problem changes and the behaviour 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 thoroughly analyze the impact of horizon and replanning frequency on the performance and robustness of the MPC framework in order to gain insights on tuning parameters in the case of MPC for legged robotics.

The entire framework was 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 control loop and the MPC loop which was run at 20 Hz. The BiConMP is run on a node using a service client setup while the main inverse dynamics control loop (13) 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 Ethernet channel. The attached video shows the experiments performed on Solo12.

The entire BiConMP is implemented in C++. The biconvex dynamics optimizer is implemented from scratch containing all the implementation details discussed in the previous section for ADMM and FISTA. Croccoddyl is used in C++ to compute the IK solutions. The efficient C++ implementation of the BiConMP makes it possible to be used real-time MPC framework on the real robot. The code will be made available upon acceptance of this paper.

A. 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. 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 IK 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 rate of increase seems to be related to the nonlinearity of the underlying motion (bound having the highest angular momentum). The IK solutions also show an almost linear growth with increase in the number of collocation points. The solver does violate this trend at times depending on the termination criteria used in Crocoddyl [40]. However, it does fall back to the average trend of being linear as the number of collocation points are further increased beyond 140. In addition, the IK 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 maintains 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 about 0.9 seconds to generate a jump motion with a horizon of 7 seconds (140 collocation points $\times$ 0.05).

The fourth plot from the top in Fig. 3 shows the percentage of the total solve time consumed by the biconvex solver. Since the biconvex solver takes the most time in the framework, significant decrease in solve times can be achieved by warm starting the solver with pre-computed solutions or by highly optimizing the code [43]. Both these avenues could be explored in future work. The last plot is shown to verify that the solver is terminated only when a good solution is obtained. This is because a low dynamics violation is essential to ensure feasibility of the plan on the real robot and this is almost the same for each particular motion as the number of collocation points are increased.

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 QP in the biconvex ADMM problem is solved until the norm of the gradient falls below $1e-5$. A 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 to the sublinear convergence property of the ADMM algorithm [29]. We would also like to 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 II-C). This allows us to set a maximum number of iterations of the ADMM algorithm in order to guarantee the real-time performance of our framework. We found that in practice there are times where we may hit the maximum number of iterations before our solver finds a solution that satisfy our criteria for acceptable dynamic violation. Even though our solutions don't fall within our predefined threshold for dynamic violation, we find trajectories that are high quality enough to be executed on the robot.

Empirically, we found that setting the maximum number of ADMM iterations to 50 allowed us to run all of the required motions on the robot. In practice, we find that both the aforementioned criteria are sufficient to generate high quality trajectories that can be realized on the robot. The IK sub-

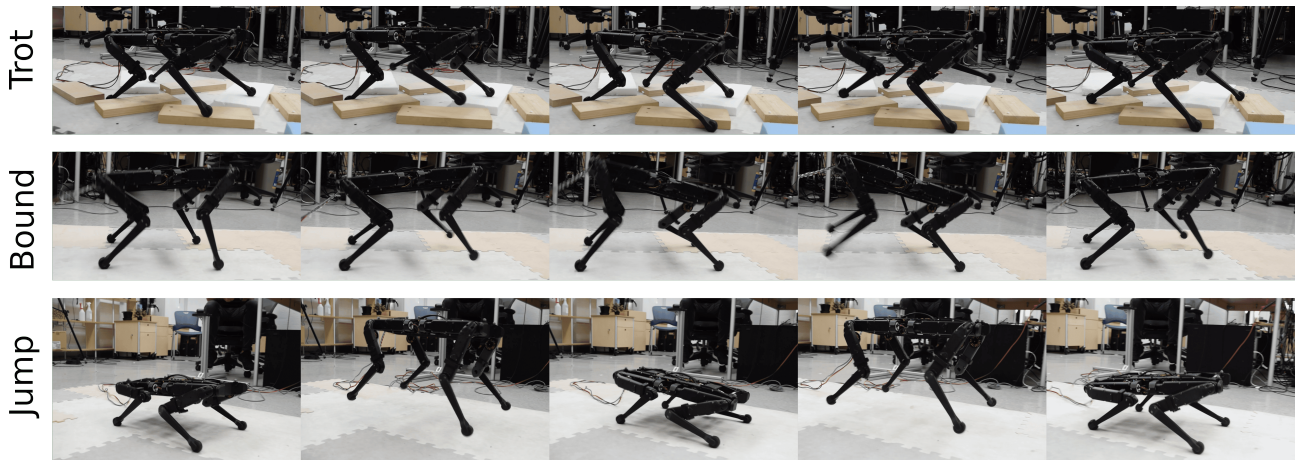

Fig. 2: Different motions demonstrated on real robot and simulation with solo12.

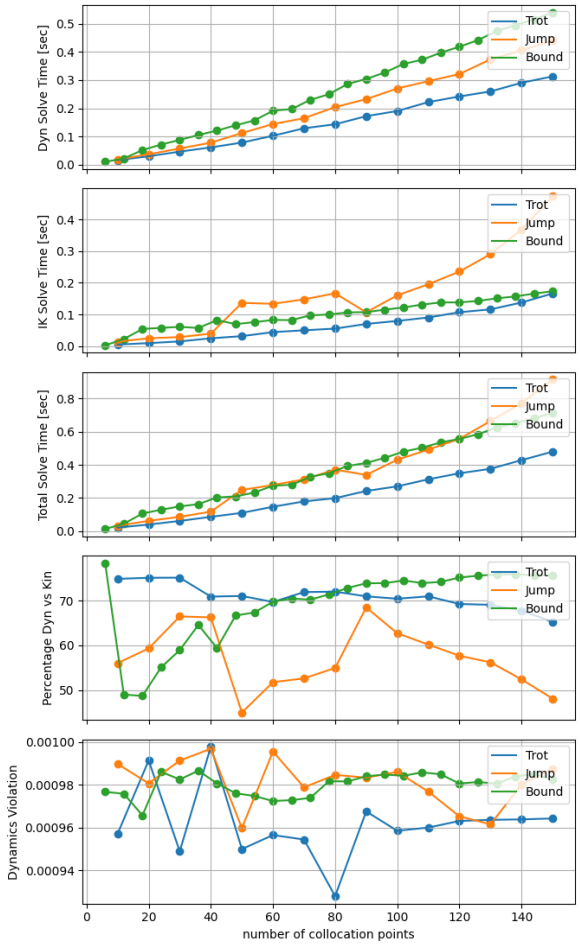

Fig. 3: Solve times vs number of collocation points in the BiConMP.

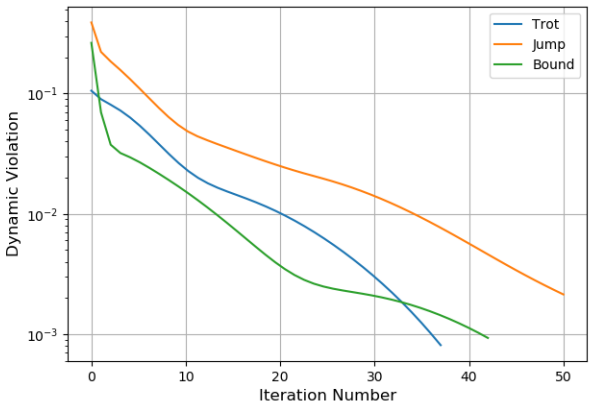

Fig. 4: Dynamic violation as compared to iterations in the dynamics optimizer.

problem is terminated based on the default settings provided by Crocoddyl.

B. Cyclic Gaits

We generated different gaits such as trots, jumps and bounds for the Solo12 quadruped and some snapshots are depicted in Fig. 2 (These motions can also be seen in the accompanying video). Table I outlines the parameters used to design the gaits that were run on the robot. For the trot gait, the gait cycle period was set to 0.3 seconds (time for two steps) and the airtime for each leg was 0.15 seconds. Stable trots were seen when the framework was allowed to plan for horizon of at least one gait period (0.3 seconds). We noted that empirically a smaller horizon 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 [44]. A gait period of 0.5 seconds with an air phase of 0.3 seconds was used to generate repetitive jumping on Solo12. A planning horizon of one gait cycle was sufficient to ensure that the framework does not diverge during run time.

Motion	Stance Time (s)	Gait Time (s)	Collocation Discretization (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.

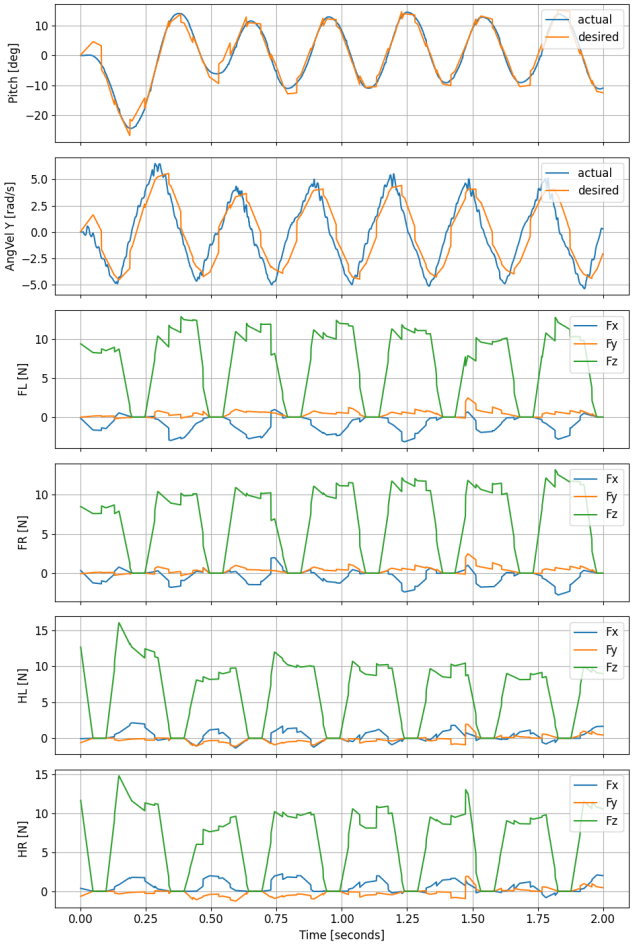

Fig. 5: Angular velocity and forces during bounding gait.

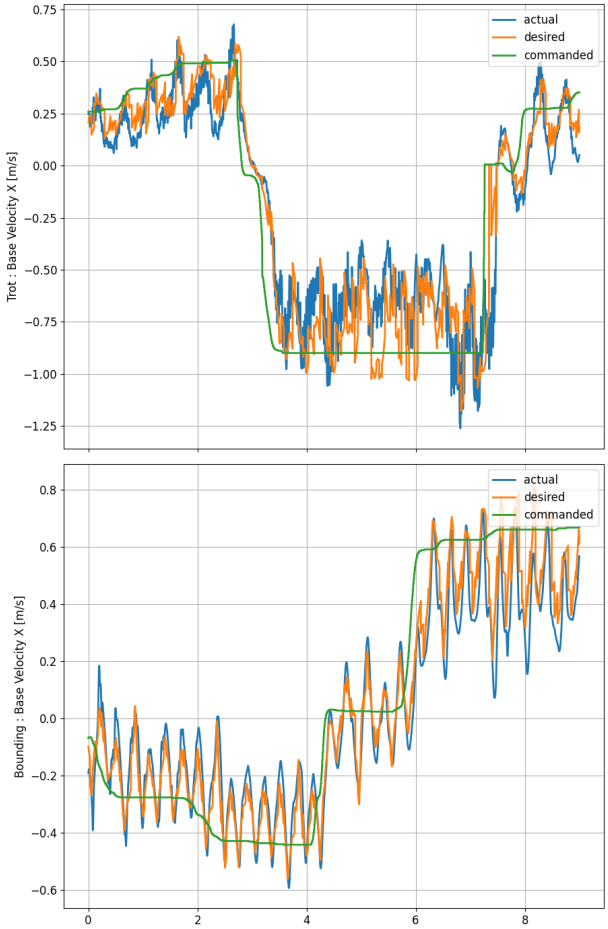

Fig. 6: Velocity tracking for trot and bound gait on the real robot. x axis is the elapsed time in seconds.

For bounding, the gait period of 0.3 seconds with an air phase of 0.15 seconds was used. Figure 5 shows the actual and desired base angular velocity about the Y axis (pitching axis) and the desired forces from the planner. A planning horizon of 2 gait cycles was necessary to ensure stability of the solver. 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 of the framework for the trot and bounding gait 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 for the different gaits.

To evaluate the solve times of the framework, each gait (trot, jump and bound) are run on the robot in 3 different scenarios, i) flat ground with no disturbance, ii) flat ground with external disturbances, iii) uneven terrain (step height of 5-8 cm) without external disturbances. In Fig. 7, we plot the obtained solve times in these scenarios. As can be seen, the solve time does not exceed the replanning frequency of 50 ms and remains real-time regardless of the circumstances. Also, it is interesting to note that the 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 is quite robust to uncertain situations. On the other hand, the solve times with the same gait parameters / weights were lower on an average by about 5-6 milliseconds without sensor noise (such as in simulation). Hence, to achieve higher re-planning frequencies, obtaining clean sensor data becomes important.

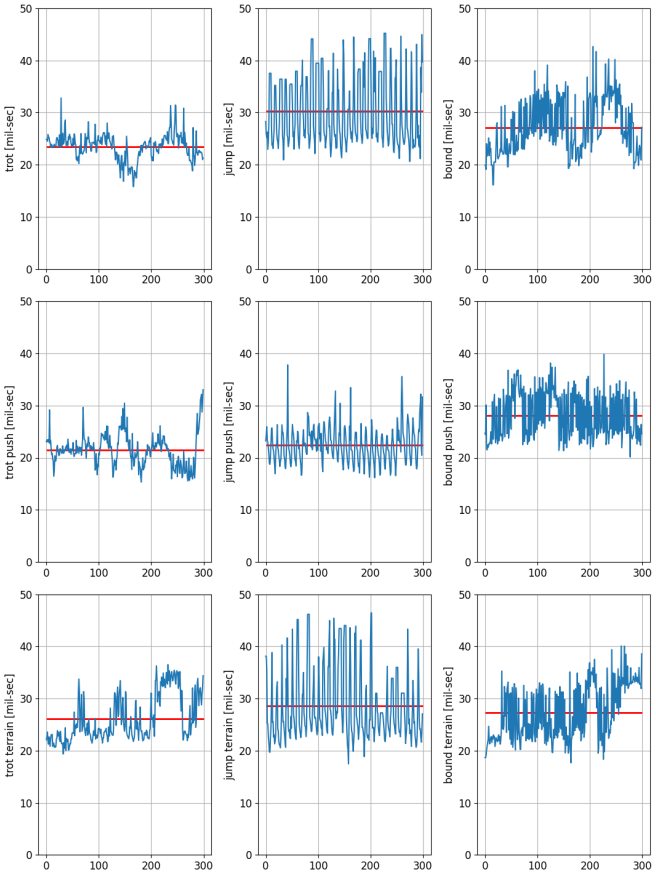

Fig. 7: Comparison of solve times of the framework for different gaits under different scenarios. x axis is the elapsed time in seconds. The horizontal line in each subplot depicts the mean solve times.

The solve times of the framework for different motions usually 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 shown to cause instabilities on real robots during run times and several approaches have been proposed to deal with this issue [45]. During our experiments on Solo12, 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 when compared to simulation where time can be frozen until a new plan is available.

1) 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. 8. The robot is initially in a brief trot phase depicted by the almost constant pitch and base height. After which, the robot

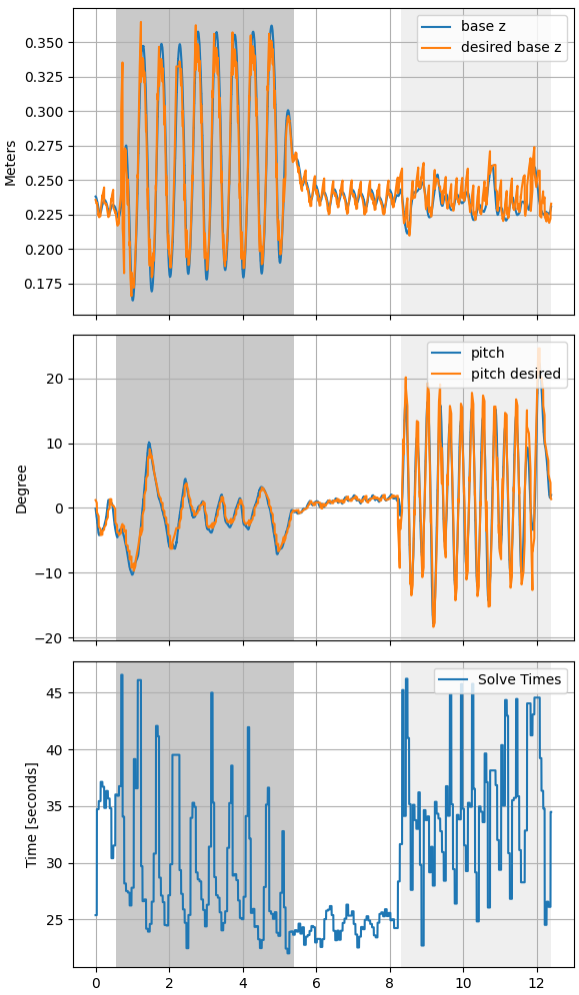

Fig. 8: Solve times during live gait transitions. The white color regions corresponds to the trot gait, the dark grey section represents the jumping gait and the light grey section depicts the bounding phase. The x axis label is the elapsed time in seconds.

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 of these changes between gaits. During all these transitions the framework keeps its solve times below 50 milliseconds. Also, the solve times for each gait remain similar to when there are no transitions (Fig. 7).

C. Acylic Motions

To demonstrate the capability of the BiconMP to generate arbitrary dynamic acyclic trajectories, 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 in order to High-Five at a fixed

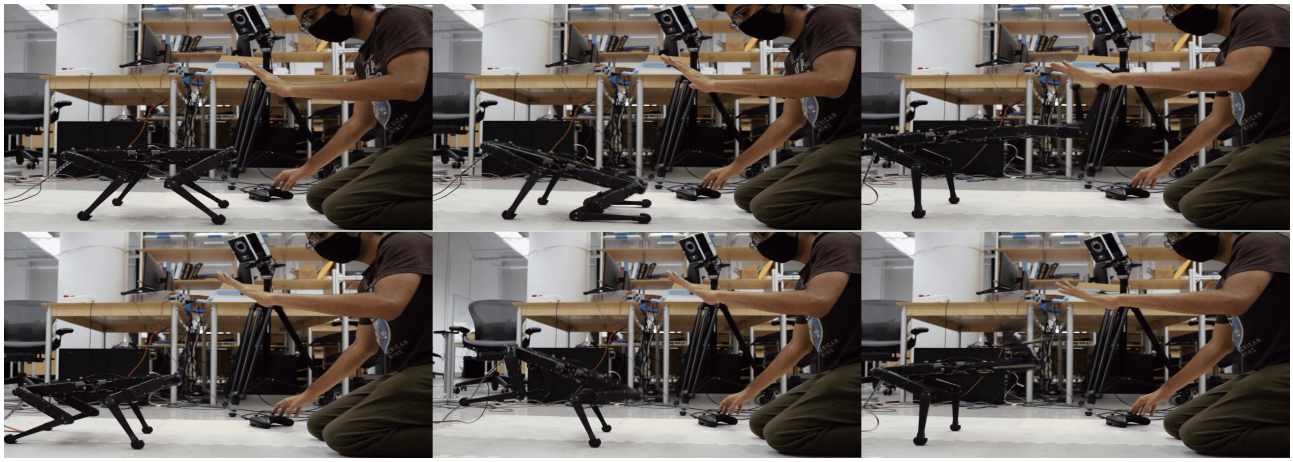

Fig. 9: Acyclic motions.(Top left to bottom left in clock wise direction)

position. Figure 9 shows the High-Five motion generated on the robot. Since Solo12 must balance on its hind legs for a duration of the High-Five, the motion is dynamic and non-trivial to achieve. The motion planner initially gets the robot to crouch before the front two legs lift off in order 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 represent 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 z force becomes zero in the plan in this zone.

During this motion, we would like to highlight that the IK sub-problem finds a non-trivial rotating motion for the front legs after performing the High-Five in order to track the centroidal momentum trajectories (i.e. momentum-obeying)that is provided by the biconvex centroidal dynamics solver. Specifically, the front legs of the robot swing their legs backwards and pivot around the hip joint in order to obey the momentum profile. Such a swing foot trajectory is very difficult to design apriori for a given motion which highlights the advantages of the whole-body motion planner.

The BiConMP framework, can easily be extended to generate robust motions on bipeds and humanoid robots. This is because it is based on the kino-dynamic formulation which has been used to generate complicated motions in previous works [22], [33], [42]. Due to the significant decrease in solve times as compared to these previous approaches, our framework could be used in an MPC setting for bipeds and humanoids. Consequently, our method can be used as unified framework across different robot platforms, which is not possible with simplified dynamics based approaches.

V. MPC Analysis

In this section, we analyse the contribution of the inverse dynamics (RNEA) controller in the stability of the robot and the advantages of using a nonlinear MPC setting on the robot

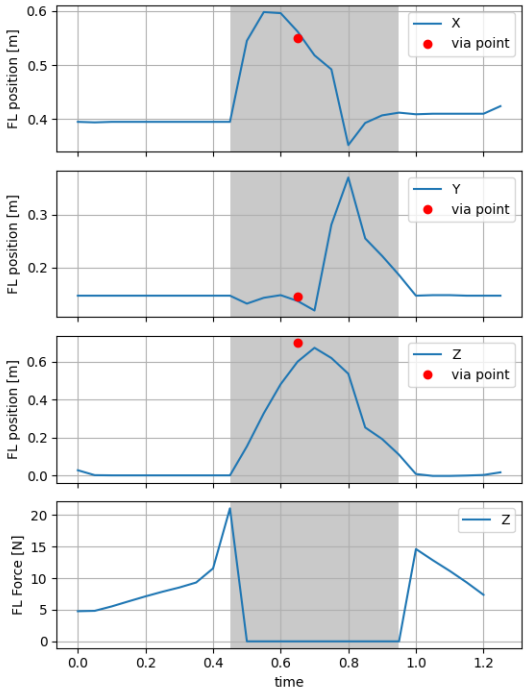

Fig. 10: Swing foot trajectory of front left leg from the IK.

as compared to pure trajectory optimization (when a fixed plan is tracked on the robot). Further, we also discuss the 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 BiConMP takes the dominant position while controlling the motion of the robot during the contact phase. Also, we see a qualitative improvement in compliance of the robot with the MPC framework as compared to a pure centroidal controller [32] during pushes, because the BiConMP is less aggressive in correcting the motion due to the horizon in the controller. The PD controller seems to play a significant role during an air to a contact phase transition where there is an instantaneous jump in the planned contact forces from the planner. During run time on the robot, these forces from the BiConMP are interpolated because the planning discretization (set to 0.05 seconds) is larger than the control frequency. 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, which is why the PD controller seems to contribute more during these transitions.

B. Sim-to-Real Transfer

Initially, the motions discussed in section IV-B and IV-C were first validated in simulation (Raisim [46]). During this stage, the cost functions (weight tuning of the optimization problem) and gait parameters were altered until a desirable motion was observed in 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. This suggests that the MPC framework is able to compensate for model error between sim-to-real transfer as it takes the dominant position during run times as compared to the PD controller. Further, it is interesting to note that the same set of P gains of 3.0 and D gains of 0.05 were used across all the motions presented in the result section for both simulation and real robot experiments. This is usually never the case during pure trajectory optimization [22]. Often different gains are needed across different motions while in simulation and these gains are further different from the gains used on the robot for the same motion. This is a significant benefit provided by the MPC setup as gain tuning on the real robot is often cumbersome.

C. Impact of re-planning frequency on performance

In the second set of experiments we analyse the affect of re-planning frequency on the performance of the MPC. During this experiment, we chose 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 [47], [48] to evaluate the performance of the MPC. We ran this experiment 4 times in which 2 times Solo12 walks on flat ground while

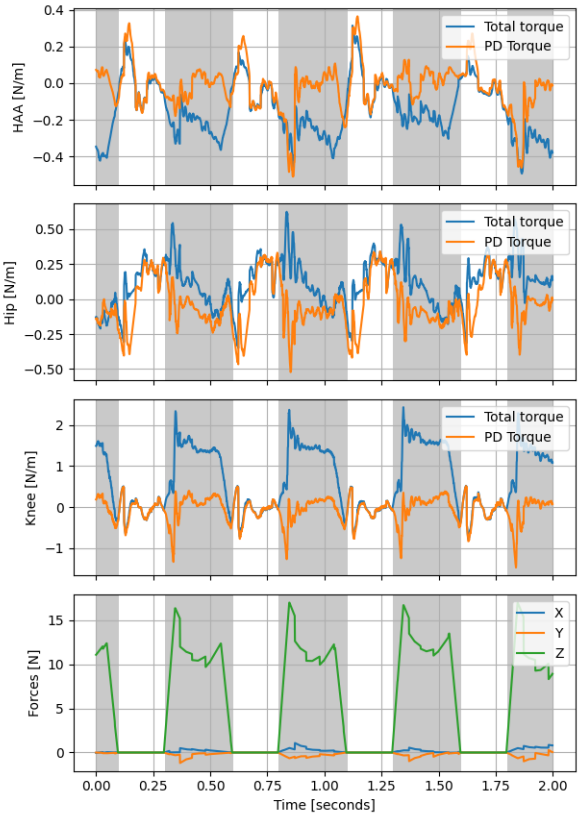

Fig. 11: Contribution of PD gains to the total torques. The three joints are Hip Abduction Adduction (HAA), the hip joint and the knee joint

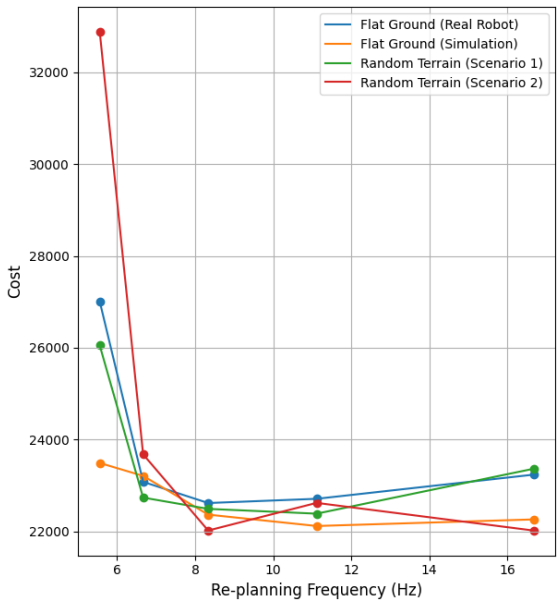

Fig. 12: Comparison of MPC performance vs. replanning frequency. The change in the mean optimal cost returned by the MPC is plotted against replanning frequency.

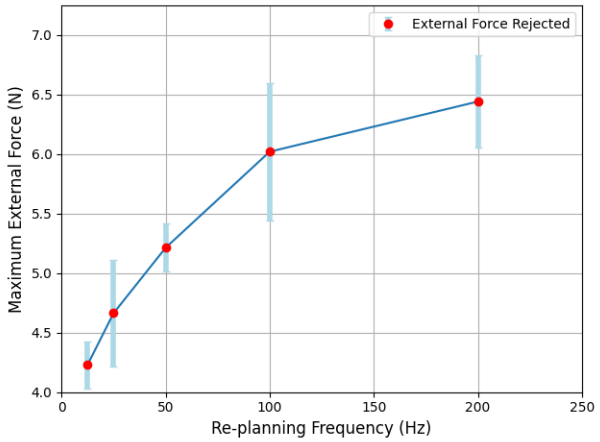

Fig. 13: Comparison of maximum external disturbance vs. re-planning frequency for 3 different motions: trotting, bounding, and jumping. Experiment performed in simulation.

the other two times it walks on random terrain (as shown in Fig. 2). Also, the same experiment is 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 a desired threshold 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 could be seen as a metric that explains the direct transfer of motions from simulation to the real robot without any additional changes.

D. Impact of planning horizon on performance

In the third set of experiments, we analyse how the planning horizon influences the performance of the MPC. The experimental setup was kept identical to the previous subsection (subsection V-C). The only difference here is that planning horizon is changed while 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 present the 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 planning horizon. After a threshold planning horizon is reached, additional increase did not seem to bring any visible benefits. In the presence of terrain noise, a similar result was obtained. Further, the threshold planning horizon for a given re-planning frequency remained the same with or without terrain noise. Consequently, we found that at low re-planning frequency, the stability of the motion could be increased by increasing the horizon. This results is in alignment with theoretical results [48].

E. Impact of re-planning frequency on robustness

In the final set of experiments, we analyse the increase in robustness as the re-planning frequency is increased. To evaluate the robustness, we choose to use the magnitude of external disturbance rejection as a criteria. Since this is hard to measure 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). We pause the simulation until the new solution is available to evaluate robustness. In each run of this experiment, Solo12 is perturbed with an external force at the base at 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 and run each experiment 10 times for each frequency. The resulting plot is 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

Alternative approaches for running MPC problems on legged robots have focused on simplifying the dynamics [25] or warm starting nonlinear solvers [49], [43]. Though these approaches are successful they have been designed for a specific set of motions and it is difficult to scale them to more complex, arbitrary, motions. Instead, by exploiting the structure of the underlying robot dynamics, we show that it is possible to solve the full-body motion planning problem quickly without the need to warm start the solver. Moreover, solving the original nonlinear problem allows the generation of very complicated motions across different robots (quadrupeds, bipeds, humanoids etc.) with the same underlying framework. Throughout all out experiments and tests with the BiConMP, we were able to generate any motion plan that we desired to see on the robot. This suggests that similar performance can be expected for other motions as well.

The main assumption in the kino-dynamic decomposition of the nonlinear robot dynamics is that there exists sufficient torque authority [21], without which computing feasible torques becomes 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 behaviours 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 as discussed in previous work [22].

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. Also, the whole body optimization is able to automatically change swing foot trajectories, base orientation etc., to improve the stability of the robot at higher velocities (like rotating the base pitch while trotting quickly). 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. 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 as we also observe this in practice on Solo12. On the other hand, if higher robustness is desired, a higher re-planning frequency does indeed buy robustness. This could be achieved by exploring avenues such as warm starting the solver and optimizing the code to reduce unwanted computations.

Recently, deep reinforcement learning (DRL) has become an increasingly popular choice to generate robust trajectories for legged robots [50], [51]. This is mainly because speeding up optimization based techniques for MPC frameworks seemed difficult and model free approaches have achieved impressive results. However, with our proposed method the re-planning frequency are now comparable to these methods, with the scope of further speed up in future work. In addition, generating new trajectories with a motion planner 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. 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 in simulation or domain randomization is necessary for successful transfer.

Conclusion

In this work, we proposed a novel nonlinear MPC framework, the BiConMP, which is capable of generating whole body motion plans in real-time for various limbed robots. The BiConMP exploits the biconvex nature of the centroidal dynamics which makes it possible to solve the original nonlinear problem in real-time. Secondly, the nonlinear inverse kinematics problem is formulated as an optimal control problem which can be leveraged by off-the-shelf DDP solvers [40]. We then close the loop by implementing this framework in an MPC fashion to generate several motions such as trotting, bounding, and jumping on Solo12. In addition, we show that we are able to successfully generate arbitrary motions such as a High-Five which leverage the full-body capabilities of the robot. Finally, we analyzed various parameters of our proposed MPC framework such as frequency, cost, and horizon to understand their impact on the performance and robustness on the robot in the real world. In future work, we hope to further speed up the biconvex centroidal solver by exploiting the warm starting capabilities and further optimizing the run-time and performance of our implementation. Secondly, attempts will be made to replace the heuristic Raibert controller with a contact planning algorithm to further reduce the required user input and enable multi-contact behaviours on non co-planar settings.

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] D. H. Jacobson, "New second-order and first-order algorithms for determining optimal control: A differential dynamic programming approach," *Journal of Optimization Theory and Applications*, vol. 2, no. 6, pp. 411–440, 1968.

[13] 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.

[14] 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.

[15] 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.

[16] 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.

[17] 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.

[18] 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.

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

[20] 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.

[21] 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.

[22] 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.

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

[24] 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.

[25] 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.

[26] 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.

[27] 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.

[28] P. Shah, A. Meduri, W. Merkt, M. Khadiv, I. Havoutis, and L. Righetti, "Rapid convex optimization of centroidal dynamics using block coordinate descent," *arXiv preprint arXiv:2108.01797*, 2021.

[29] 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.

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

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

[32] 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.

[33] A. Herzog, N. Rotella, S. Mason, F. Grimminger, S. Schaal, and L. Righetti, "Momentum control with hierarchical inverse dynamics on a torque-controlled humanoid," *Autonomous Robots*, vol. 40, no. 3, pp. 473–491, 2016.

[34] D. E. Orin, A. Goswami, and S.-H. Lee, "Centroidal dynamics of a humanoid robot," *Autonomous robots*, vol. 35, no. 2-3, pp. 161–176, 2013.

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

[36] 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.

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

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

[39] 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.

[40] 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.

[41] 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.

[42] 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.

[43] 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.

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

[45] 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.

[46] 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.

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

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

[49] 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.

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

[51] 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.