A Unified MPC Framework for Whole-Body Dynamic Locomotion and Manipulation

Jean-Pierre Sleiman, *Graduate Student Member, IEEE*, Farbod Farshidian, *Member, IEEE*, Maria Vittoria Minniti, *Graduate Student Member, IEEE*, and Marco Hutter, *Member, IEEE*

***Abstract*—In this letter, we propose a whole-body planning framework that unifies dynamic locomotion and manipulation tasks by formulating a single multi-contact optimal control problem. We model the hybrid nature of a generic multi-limbed mobile manipulator as a switched system, and introduce a set of constraints that can encode any pre-defined gait sequence or manipulation schedule in the formulation. Since the system is designed to actively manipulate its environment, the equations of motion are composed by augmenting the robot's centroidal dynamics with the manipulated-object dynamics. This allows us to describe any high-level task in the same cost/constraint function. The resulting planning framework could be solved on the robot's onboard computer in real-time within a model predictive control scheme. This is demonstrated in a set of real hardware experiments done in free-motion, such as base or end-effector pose tracking, and while pushing/pulling a heavy resistive door. Robustness against model mismatches and external disturbances is also verified during these test cases.**

***Index Terms*—Legged robots, mobile manipulation, multi-contact whole-body motion planning and control, optimization and optimal control.**

I. Introduction

WE OFTEN judge the agility of a poly-articulated robotic system, such as a humanoid or quadruped, by the degree to which it is able to mimic its biological counterpart. This resemblance should appear in the robot's ability to properly coordinate a wide range of complex body movements, and its ability to effectively interact with its environment. Such an interaction could be directed towards moving and balancing the robot's base (locomotion), or towards moving another object (manipulation). The governing dynamics in such problems are hybrid, underactuated, and highly non-linear; this in turn renders the design of controllers for such systems quite challenging.

A broad range of work in the literature relies on a decomposition of the full control problem into two main units, namely a planning module and a tracking module. The latter is responsible for generating the torque commands needed to compliantly track the high-level references computed by the planner. Typically, the tracking controller is based on variants of the standard operational-space inverse dynamics approach [1]. These variants formulate the tracking problem within an optimization setting to properly resolve the system's redundancy while allowing for the incorporation of system constraints [2]–[6]. As for the planning module, which is also the main focus of this letter, it is responsible for generating center-of-mass (CoM) motions, contact locations and forces, as well as limb motion trajectories. These can be computed either simultaneously within the same planner, or separately in decoupled sub-modules, depending on the adopted dynamic model. Generally, these planners rely on simplified template models – with a few notable exceptions that use the full dynamics [7]–[10] – where a wide spectrum of options trades off physical accuracy against computational complexity. For instance, Bellicoso *et al.* [11] demonstrate agile dynamic locomotion on a quadrupedal robot by decomposing their planning framework into the aforementioned three main elements. The CoM trajectories are generated and updated online on the basis of the zero-moment-point (ZMP) dynamic stability criterion [12]. On the other hand, in the works of Di Carlo *et al.* [13] and Villareal *et al.* [14], the quadruped is modeled as a single rigid-body subject to contact patches. With certain assumptions made on the base's orientation and angular velocity, they are able to formulate their planning problem as a convex optimization problem. This is then solved within a Model Predictive Control (MPC) scheme to compute online CoM motions and reference contact forces.

The same template model is also adopted in [15], where a single trajectory optimization (TO) framework is used to compute contact-schedules, swing-leg motions and base trajectories, for locomotion over uneven terrain. Dai *et al.* [16] use a full centroidal dynamic description [17] along with the full kinematics of a humanoid robot within a trajectory optimization setting. Their formulation is also able to discover contacts automatically through the use of complementarity constraints [9]. A similar model is also adopted in [18], where the authors demonstrate real-hardware experiments on a humanoid robot climbing stairs while grabbing a hand-rail, given a predefined contact sequence. However, all of these formulations could not be solved in real-time, and thus do not allow for fast online-replanning. On the other hand, there has been relatively more recent attempts to have a unified locomotion MPC-scheme that takes advantage of fast trajectory-optimization techniques. One such example is presented in [19], where the multi-contact problem is formulated within the framework introduced in [20],

Manuscript received October 15, 2020; accepted February 11, 2021. Date of publication March 25, 2021; date of current version April 13, 2021. This letter was recommended for publication by Associate Editor C. Semini and Editor A. Kheddar upon evaluation of the reviewers' comments. This work was supported in part by the Swiss National Science Foundation through the National Centre of Competence in Research Robotics (NCCR Robotics) and the National Centre of Competence in Research Digital Fabrication (NCCR Dfab), and in part by TenneT. *(Corresponding author: Jean-Pierre Sleiman.)*

The authors are with the Robotic Systems Lab, ETH Zurich, 8092 Zurich, Switzerland (e-mail: jsleiman@ethz.ch; farshidian@mavt.ethz.ch; mariavittoria.minniti@mavt.ethz.ch; mahutter@ethz.ch).

This letter has supplementary downloadable material available at https://doi.org/10.1109/LRA.2021.3068908, provided by the authors.

Digital Object Identifier 10.1109/LRA.2021.3068908

[21]. This work makes use of a similar optimal control problem description.

When it comes to articulated whole-body manipulation, the most notable contributions could be found in the works of Murphy *et al.* [22], where a quadrupedal mobile manipulator is shown performing coordinated athletic motions in a dynamic lifting and throwing task, and in Bellicoso *et al.* [23] where a similar system is used in a door pushing scenario. However, unlike this work, in [22] the manipulation task is planned offline and is separated from the locomotion planner; while in [23], the door-opening task is solved in a reactive fashion by explicitly specifying gripper forces to be tracked by the whole-body controller.

The main contributions of this work are three-fold.

- We exploit the duality between dynamic locomotion and manipulation tasks to formulate a unifying multi-contact optimal control problem. To this end, we adopt a switched-systems perspective when handling contact-making and contact-breaking events. Moreover, we use a single set of constraints to describe any pre-defined gait sequence or manipulation contact-schedule.
- We propose an augmented model consisting of the manipulated-object dynamics, the robot's centroidal dynamics, and the full kinematics. This description enables us to encode any robot-centric or object-centric task in the same cost/constraint function.
- We show, by using the platform in Fig. 1, that the resulting OC framework can resolve a wide variety of free-motion tasks and manipulation problems. By exploiting the whole-body natural dynamics, the planner is able to generate coordinated maneuvers that push our system's performance limits without violating any physical constraints. Most importantly, we manage to solve the OC problem in real-time, in a receding-horizon fashion. This results in an MPC scheme that can easily be deployed on hardware and solved with limited on-board computational power.

To the best of our knowledge, this work is one of the first examples demonstrating a real-hardware application of a whole-body MPC framework that unifies dynamic locomotion and manipulation.

II. Problem Formulation

A. Whole-Body Planner

The nonlinear-MPC framework adopted in this work is based on the solver introduced in [20], [21] which is tailored to handle optimal control problems involving hybrid dynamical systems, by treating them as switched systems with predefined modes. It also offers the possibility to optimize for the switching-times, together with the optimal state and input trajectories. The underlying core algorithm is the Sequential-Linear-Quadratic (SLQ) technique, a continuous-time version of the iterative-LQR (iLQR) method [24]. More specifically, SLQ can be classified as a variant of Differential Dynamic Programming [25], an indirect trajectory optimization approach that relies on the following mechanism: Given a nominal state and input trajectory, perform a forward rollout of the dynamics, compute the quadratic approximations of the cost function and dynamics (SLQ uses a first-order approximation of the dynamics instead), then perform a backward pass on the resulting Riccati equations to finally compute a control policy consisting of a feedforward and feedback term. The full sequence of computations (i.e., one SLQ iteration) has a complexity that is linear with respect to the time horizon, unlike direct TO methods which have cubic complexity.

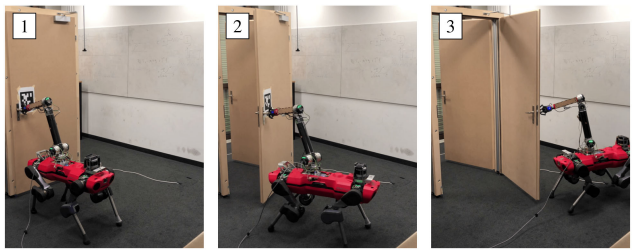


Fig. 1. A quadrupedal mobile manipulator, consisting of an ANYmal C platform equipped with a custom-made robotic arm, pulling a heavy resistive door open.

Furthermore, the unconstrained-SLQ algorithm has been extended to handle state-input and state-only equality constraints [20], as well as inequality path constraints [19], through projections, penalty functions and barrier functions, respectively. Hence, we are able to devise the following optimal control problem

$$\begin{cases} \min\limits_{\boldsymbol{u}(.)} \ \Phi(\boldsymbol{x}(T)) + \int_0^T L(\boldsymbol{x}(t), \boldsymbol{u}(t), t)dt \\ \text{s.t. } \dot{\boldsymbol{x}}(t) = \boldsymbol{f}(\boldsymbol{x}(t), \boldsymbol{u}(t), t) \\ \quad \boldsymbol{g}_1(\boldsymbol{x}(t), \boldsymbol{u}(t), t) = 0 \\ \quad \boldsymbol{g}_2(\boldsymbol{x}(t), t) = 0 \\ \quad \boldsymbol{h}(\boldsymbol{x}(t), \boldsymbol{u}(t), t) \geq 0 \\ \quad \boldsymbol{x}(0) = \boldsymbol{x}_0, \end{cases} \tag{1}$$

where $\boldsymbol{x}(t) \in \mathbb{R}^{n_x}$ and $\boldsymbol{u}(t) \in \mathbb{R}^{n_u}$ are vectors of state and input variables, while $L(\boldsymbol{x}, \boldsymbol{u}, t)$ and $\Phi(\boldsymbol{x}(T))$ are the stage cost (Lagrange term) and terminal cost (Mayer term), respectively. This problem is solved in closed-loop with the SLQ-MPC framework that essentially runs the constrained-SLQ algorithm in a receding horizon fashion within a real-time iteration scheme [26]. To simplify notation, we omit any dependence of (1) on the modes and their corresponding switching-times, which are assumed to be fixed in our case. It is worth mentioning that the switched nature of our dynamic locomotion/manipulation setup is captured in the constraints rather than in the system dynamics. Therefore, this matter will be made clear in Sections II-A2 and II-A3, while the discussion regarding cost function design is deferred to Section IV.

1) System Modeling: A poly-articulated floating-base system, such as the one depicted in Fig. 2, can be properly modeled as an unactuated 3D rigid body to which is attached a set of fully-actuated limbs. The resulting dynamics are therefore governed by the following set of equations:

$$\boldsymbol{M}_u(\boldsymbol{q})\dot{\boldsymbol{\nu}} + \boldsymbol{b}_u(\boldsymbol{q}, \boldsymbol{\nu}) = \boldsymbol{J}_{c_u}^T(\boldsymbol{q})\boldsymbol{F}_c \tag{2a}$$

$$\boldsymbol{M}_a(\boldsymbol{q})\dot{\boldsymbol{\nu}} + \boldsymbol{b}_a(\boldsymbol{q}, \boldsymbol{\nu}) = \boldsymbol{\tau}_a + \boldsymbol{J}_{c_a}^T(\boldsymbol{q})\boldsymbol{F}_c \tag{2b}$$

where $\boldsymbol{q}, \boldsymbol{\nu} \in \mathbb{R}^{6+n_a}$ are the generalized coordinates and generalized velocities, respectively. A ZYX-Euler angle parameterization is assumed to represent the base's orientation. $\boldsymbol{M}$ is the generalized mass matrix, $\boldsymbol{b}$ represents the nonlinear effects (i.e., Coriolis, centrifugal, and gravitational terms), and $\boldsymbol{\tau}_a$ is

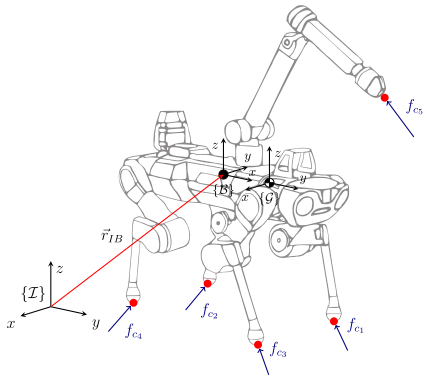


Fig. 2. Illustration of the multi-limbed floating-base system used in this work. The sketch depicts the main reference frames used in our derivation of the equations of motion (inertial $\{\mathcal{I}\}$, base $\{\mathcal{B}\}$, and centroidal $\{\mathcal{G}\}$ frames), in addition to the contact forces acting on the limbs.

the vector of actuation torques. $\boldsymbol{J}_c$ is a matrix of stacked contact Jacobians, while $\boldsymbol{F}_c$ is a vector of stacked contact wrenches. The subscripts u and a correspond to the unactuated and actuated parts of the defined quantities, respectively. Under the mild assumption that one has sufficient control authority in the robot's joints, it would be justifiable to independently consider subsystem (2a) in the TO formulation as a simplified template model. In fact, with the proper transformation applied to (2a), one could equivalently retrieve the Newton-Euler equations applied at the robot's center of mass (CoM), or the centroidal dynamics [27]

$$\dot{\boldsymbol{h}}_{com} = \begin{bmatrix} \sum_{i=1}^{n_c} \boldsymbol{f}_{c_i} + m\boldsymbol{g} \\ \sum_{i=1}^{n_c} \boldsymbol{r}_{com,c_i} \times \boldsymbol{f}_{c_i} + \boldsymbol{\tau}_{c_i} \end{bmatrix}. \tag{3}$$

$\boldsymbol{h}_{com} = (\boldsymbol{p}_{com}, \boldsymbol{l}_{com}) \in \mathbb{R}^6$ is defined as the centroidal momentum, with $\boldsymbol{p}_{com}$ and $\boldsymbol{l}_{com}$ being the linear momentum and angular momentum about the centroidal frame $\mathcal{G}$,[1] respectively. $\boldsymbol{r}_{com,c_i}$ denotes the position of the contact point c_i with respect to the center of mass, while $\boldsymbol{f}_{c_i}$ and $\boldsymbol{\tau}_{c_i}$ are the contact forces and torques applied by the environment on the robot at c_i.

In order to capture the effect of the generalized coordinates' rate of change on the centroidal momentum, we consider the mapping introduced in [17] through the centroidal momentum matrix (CMM) $\boldsymbol{A}(\boldsymbol{q}) \in \mathbb{R}^{6\times(6+n_a)}$ - which is constructed as a function of the system's full kinematic configuration and the multi-body inertias-

$$\boldsymbol{h}_{com} = \underbrace{\begin{bmatrix} \boldsymbol{A}_b(\boldsymbol{q}) & \boldsymbol{A}_j(\boldsymbol{q}) \end{bmatrix}}_{\boldsymbol{A}(\boldsymbol{q})} \begin{bmatrix} \dot{\boldsymbol{q}}_b \\ \dot{\boldsymbol{q}}_j \end{bmatrix}. \tag{4}$$

This could be rearranged in the following form

$$\dot{\boldsymbol{q}}_b = \boldsymbol{A}_b^{-1} \left(\boldsymbol{h}_{com} - \boldsymbol{A}_j \dot{\boldsymbol{q}}_j \right), \tag{5}$$

where $\boldsymbol{q}_b = (\boldsymbol{r}_{IB}, \boldsymbol{\Phi}_{IB}^{zyx}) \in \mathbb{R}^6$ is the base pose with respect to the inertial frame. It is important to note that including (5) in our equations of motion rids our model of the standard massless-limbs assumption, which is inherent to most template models of legged systems. Having laid down the foundations for defining physically consistent dynamics in our motion planner, we are now able to represent the robot as a dynamical system with state vector $\boldsymbol{x}_r = (\boldsymbol{h}_{com}, \boldsymbol{q}_b, \boldsymbol{q}_j) \in \mathbb{R}^{12+n_a}$ and input vector $\boldsymbol{u} = (\boldsymbol{f}_{c_1}, \{\ldots, \boldsymbol{f}_{c_{n_c}}, \boldsymbol{v}_j) \in \mathbb{R}^{3n_c+n_a}$ with

$$\dot{\boldsymbol{q}}_j = \boldsymbol{v}_j, \tag{6}$$

here we have neglected contact torques as inputs by assuming points of contact rather than patches. The system of interest, illustrated in Fig. 2, consists of 16 actuated joints (3 per leg and 4 for the arm) and 5 potential contact points; this entails a total of 28 state variables and 31 inputs.

When dealing with whole-body manipulation problems, having a planner that directly encapsulates the task description and is also aware of the robot-object dynamic coupling is fundamental for effective task-attainment. This could be achieved by augmenting the object state $\boldsymbol{x}_o = (\boldsymbol{q}_o, \boldsymbol{v}_o) \in \mathbb{R}^{2n_o}$ to $\boldsymbol{x}_r$, where the object dynamical system is defined similarly to (2a)

$$\dot{\boldsymbol{x}}_o = \begin{bmatrix} \boldsymbol{v}_o \\ \boldsymbol{M}_o^{-1} \left(-\boldsymbol{J}_{c_o}^T \boldsymbol{f}_{c_5} - \boldsymbol{b}_o \right) \end{bmatrix}. \tag{7}$$

The term $\boldsymbol{b}_0$ captures all position and velocity-dependent generalized forces, such as spring-damper effects. We note that the underlying assumptions that make such a state augmentation possible are that the object model structure and parameters are known, and that the object state is observable as it needs to be continuously fed back to the MPC solver. Finally, the full system flow-map $\dot{\boldsymbol{x}} = \boldsymbol{f}(\boldsymbol{x}, \boldsymbol{u})$ with the augmented state $\boldsymbol{x} = (\boldsymbol{x}_r, \boldsymbol{x}_o)$ could be set up by collecting (3), (5), (6), and (7).

2) Equality Constraints: All of the equality constraints are defined at the level of the potential contact points, which could be in one of two states: open or closed. Thereby, the constraints would depend on a predefined mode-schedule that consists of a mode-sequence coupled with a set of switching times. For the sake of compactness in notation, we denote the set of all closed contacts by $\mathcal{C}$, the set of feet contact points by $\mathcal{F}$, and the set of arm contacts by $\mathcal{A}$ (which in our case is a singleton). Given a fixed mode instance $\mathcal{C}_j$ that starts at time s_j^0 and ends at s_j^f, the state-input equality constraints corresponding to this mode $\boldsymbol{g}_{1_j}(\boldsymbol{x}, \boldsymbol{u}, t)$ can be established as the following: $\forall t \in [s_j^0,\ s_j^f]$ and $\forall i \in \{1, \ldots, n_c\}$

$$\begin{cases} \boldsymbol{v}_{c_i} = \boldsymbol{0} & \text{if } c_i \in \mathcal{C}_j \ \wedge \ c_i \in \mathcal{F} \quad (8a) \\ \boldsymbol{v}_{c_i} \cdot \hat{\boldsymbol{n}} = v^*(t) & \text{if } c_i \in \bar{\mathcal{C}}_j \ \wedge \ c_i \in \mathcal{F} \quad (8b) \\ \boldsymbol{v}_{c_i} - \boldsymbol{J}_{c_o} \boldsymbol{v}_o = 0 & \text{if } c_i \in \mathcal{C}_j \ \wedge \ c_i \in \mathcal{A} \quad (8c) \\ \boldsymbol{f}_{c_i} = \boldsymbol{0} & \text{if } c_i \in \bar{\mathcal{C}}_j, \quad (8d) \end{cases}$$

where $\boldsymbol{v}_{c_i}$ is the absolute linear velocity of contact point c_i expressed in the inertial frame. The constraints presented in (8) have the following implications: forces at open contacts vanish (8d), the foot of a stance leg should not separate or slip with respect to the ground (8a), and a grasped object should remain in contact with the end-effector at the gripping point (8c). Furthermore, all swing legs should track a reference trajectory $v^*(t)$ along the surface normal $\hat{\boldsymbol{n}}$ (8b), thus leaving the remaining orthogonal directions – which determine the stride-length – free for the planner to resolve. It is worth noting that while it is true that a wide variety of manipulation tasks could be handled by supposing a continuous grasp, allowing for an open arm-contact

[1]The centroidal frame is a reference frame attached to the robot's center of mass and aligned with the world frame.

generalizes the planning framework by additionally covering non-prehensile tasks. In such cases, the robot is expected to lose contact with the object during the manipulation period.

3) Inequality Constraints: One of the downsides of using a DDP-based method for trajectory optimization is that inequality constraints are not naturally handled by Riccati-based solvers. As mentioned in the introduction of Section II-A, one remedy has been proposed in [19] which extends the original SLQ formulation by augmenting the inequalities to the cost in the form of a relaxed log-barrier function. In a separate concurrent letter [28], we highlight the issues that could arise from such a method, and we propose a novel inequality-constrained SLQ-MPC algorithm based on an augmented-Lagrangian formulation [29]. This algorithm is applied in this work to properly handle inequality constraints that appear in the planner, while state-input equalities are still handled with the projection technique introduced in [20]. Briefly, the idea would be to transform the inequality constrained problem into an unconstrained one by constructing the augmented Lagrangian function $\mathcal{L}_A$. Then at each iteration, $\mathcal{L}_A$ is minimized with respect to the primal variables $\boldsymbol{x}(t)$ and $\boldsymbol{u}(t)$ through a single call to the equality-constrained SLQ loop

$$(\boldsymbol{x}^*_{k+1},\ \boldsymbol{u}^*_{k+1}) = \underset{\boldsymbol{x}_k, \boldsymbol{u}_k}{\operatorname{argmin}}\ \mathcal{L}_A(\boldsymbol{h}(\boldsymbol{x}_k, \boldsymbol{u}_k), \boldsymbol{\lambda}_k), \tag{9}$$

while the Lagrange dual function is maximized in the direction of the dual variables $\boldsymbol{\lambda}(t)$ through an update rule

$$\boldsymbol{\lambda}^*_{k+1} = \boldsymbol{\Pi}\left(\boldsymbol{\lambda}_k, \boldsymbol{h}(\boldsymbol{x}^*_{k+1}, \boldsymbol{u}^*_{k+1})\right). \tag{10}$$

The algorithm eventually converges to a KKT point $(\boldsymbol{x}^*, \boldsymbol{u}^*, \boldsymbol{\lambda}^*)$ which is a potential primal-dual optimum.

To ensure that our planner generates dynamically feasible motions and forces that would also respect the system's intrinsic operational limits, we introduce the following set of constraints: $\forall t \in [s^0_j,\ s^f_j]$ and $\forall i \in \{1, \ldots, n_c\}$

$$\begin{cases} -\boldsymbol{v}_{j_{max}} \leq \boldsymbol{v}^{arm}_j \leq \boldsymbol{v}_{j_{max}} \\ -\boldsymbol{\tau}_{max} \leq \boldsymbol{J}^{a^T}_{c_i} \boldsymbol{f}_{c_i} + \boldsymbol{g}^a \leq \boldsymbol{\tau}_{max} \quad \text{if } c_i \in \mathcal{C}_j\ \wedge\ c_i \in \mathcal{A} \\ \mu_s f^z_{c_i} - \sqrt{f^{x^2}_{c_i} + f^{y^2}_{c_i} + \epsilon^2} \geq 0 \quad \text{if } c_i \in \mathcal{C}_j\ \wedge\ c_i \in \mathcal{F}. \end{cases} \tag{11}$$

where $\boldsymbol{J}^a$ and $\boldsymbol{g}^a$ are the arm Jacobian and generalized gravitational torques, respectively. These inequalities guarantee that the arm's joint velocity and torque limits are not violated, and that the feet contact forces remain inside the friction cone with friction coefficient μ_s ($\epsilon \neq 0$ is needed to smoothen the constraint). We note that the torque constraint encompasses the dynamic effects of the object in $\boldsymbol{f}_{c_i}$, but neglects the arm dynamics: Inertial terms are omitted since joint accelerations are not accessible in the planner, while velocity-dependent terms are neglected to avoid reductions in the MPC frequency. Since the commanded torques are not computed directly in the MPC layer but rather in the tracking controller, we found this to be a safe and reasonable assumption. Strict feasibility with respect to the real torque limits is indeed imposed in the QP-based controller of Section II-B.

4) Complementary Remarks: In order to compute the necessary kinematic transforms, Jacobians and CMM of our robot, we rely on the fast implementation of rigid-body dynamic algorithms provided by the *Pinocchio* C++ library [30], [31].

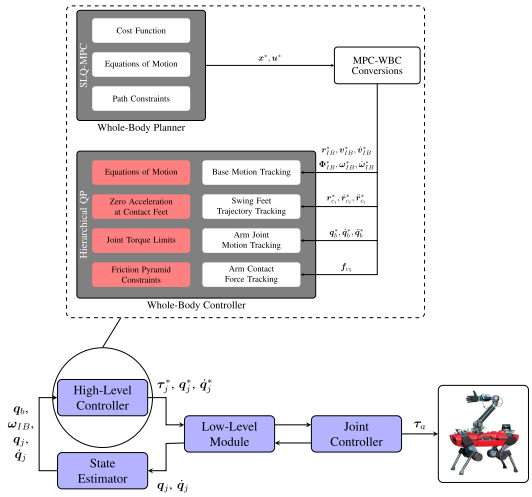


Fig. 3. A schematic diagram depicting the various components of the full control architecture. The high-level control module is highlighted and detailed in the top diagram. This consists of the whole-body planner interacting with the WBC whose high-priority tasks are indicated in red.

We also recall that the SLQ algorithm requires the gradients of the dynamics and constraints to perform the linear-quadratic approximation of (1). Accordingly, we carry out a basic comparison where we evaluate an implementation based on automatic differentiation with CppAD [32], against an implementation based on derived analytical expressions[2], of which the former turns out to be slightly more efficient.

B. Whole-Body Controller

The optimal reference plans for the base and limbs are tracked by a whole-body controller (WBC) that tries to fulfill a set of prioritized tasks. These tasks are formulated in the form of a hierarchical quadratic program (QP) that optimizes for the generalized accelerations and contact forces. Joint torques are then retrieved by inverting the desired dynamics. The objectives with their corresponding priorities are represented in the high-level controller diagram of Fig. 3. For a detailed discussion on how the QP structure is set up and solved, the reader is referred to the relevant letters [5], [11].

It is worth noting that we do not aim to directly track the optimal ground reaction forces; instead, we capture their influence by tracking the reference motion they induce on the base. The reason for that is related to the fact that the WBC imposes stricter conditions on physical correctness than the planner does, as it relies on a more realistic model. Consequently, the forces are adjusted if they violate any higher priority objectives, so by having a base motion task we direct the solver to redistribute the forces and accelerations in such a way that would, at the very least, keep the robot balanced. On the other hand, arm contact forces outputted by the MPC planner are treated differently and sent as direct references to the QP. This is because the WBC has no knowledge of the manipulated object's dynamics. Therefore, it would not be possible to define a prioritized task on the level of the object's motion.

[2]These derivations essentially make use of *Pinocchio*'s computation of the centroidal dynamics and their analytical derivatives.

It is also important to highlight the conversions needed to couple the MPC output solutions with the WBC tracking tasks. To begin with, the arm's reference joint positions, joint velocities, and contact forces are directly accessible from $\boldsymbol{x}^*$ and $\boldsymbol{u}^*$, whereas the joint accelerations $\ddot{\boldsymbol{q}}_j$ are approximated by finite differences. Swing feet trajectories are obtained from basic kinematic transformations, and by using the expression below for task-space accelerations

$$\ddot{\boldsymbol{r}}_{c_i} = \boldsymbol{J}_{c_i}\ddot{\boldsymbol{q}}_j + \dot{\boldsymbol{J}}_{c_i}\dot{\boldsymbol{q}}_j. \tag{12}$$

Similarly, the desired base pose $\boldsymbol{q}_b^*$ is part of the MPC optimal state, while its linear and angular velocities can be extracted from (5) with a simple mapping

$$\begin{bmatrix}\boldsymbol{v}_{IB}\\ \boldsymbol{\omega}_{IB}\end{bmatrix} = \begin{bmatrix}\mathbb{I}_{3\times3} & \boldsymbol{0}_{3\times3}\\ \boldsymbol{0}_{3\times3} & \boldsymbol{T}(\boldsymbol{\Phi}_{IB}^{zyx})\end{bmatrix}\dot{\boldsymbol{q}}_b, \tag{13}$$

where $\boldsymbol{T} \in \mathbb{R}^{3\times3}$ maps derivatives of ZYX-Euler angles to angular velocities. As for the feedforward base accelerations, we start by deriving the expression for $\ddot{\boldsymbol{q}}_b$ from (4) by taking the time-derivative on both sides and rearranging terms:

$$\ddot{\boldsymbol{q}}_b = \boldsymbol{A}_b^{-1}\left(\dot{\boldsymbol{h}}_{com} - \dot{\boldsymbol{A}}\dot{\boldsymbol{q}} - \boldsymbol{A}_j\ddot{\boldsymbol{q}}_j\right) \tag{14}$$

where $\dot{\boldsymbol{h}}_{com}$ is given by (3) and $\dot{\boldsymbol{A}}\dot{\boldsymbol{q}}$ is retrieved from the Recursive Newton-Euler Algorithm (RNEA) by setting the joint accelerations to zero and transforming the resulting base wrench into a centroidal wrench. Finally, the computation of the accelerations trivially follows from Eqs. (13) and (14)

$$\begin{bmatrix}\dot{\boldsymbol{v}}_{IB}\\ \dot{\boldsymbol{\omega}}_{IB}\end{bmatrix} = \begin{bmatrix}\mathbb{I}_{3\times3} & \boldsymbol{0}_{3\times3}\\ \boldsymbol{0}_{3\times3} & \boldsymbol{T}\end{bmatrix}\ddot{\boldsymbol{q}}_b + \begin{bmatrix}\boldsymbol{0}_{3\times3} & \boldsymbol{0}_{3\times3}\\ \boldsymbol{0}_{3\times3} & \dot{\boldsymbol{T}}\end{bmatrix}\dot{\boldsymbol{q}}_b. \tag{15}$$

III. System Description

The framework described in Section II is robot-agnostic and is thus capable of encompassing, with the possibility of minor modifications, a wide variety of multi-limbed systems designed for locomotion and manipulation purposes. In this work, we demonstrate the validity of our approach through a set of experiments performed on a quadrupedal mobile manipulator consisting of the ANYmal C platform equipped with the DynaArm, a custom-made 4-DoF robotic arm. The latter is a newly developed torque-controllable manipulator, which comprises four powerful actuators that allow for highly dynamic maneuvers and a high payload capability of 7 kg. The distinguishing feature of the DynaArm is that the elbow flexion joint is driven by an actuator that is situated at the shoulder through a belt transmission mechanism. This has the advantage of reducing the torque load on the shoulder flexion joint, since the reaction torque caused by the elbow drive is now directly transmitted to the arm's base.

The full control architecture is depicted in the schematic diagram of Fig. 3. Apart from the joint controller, all of the modules run on the robot's onboard computer (Intel Core i7-8850H CPU@4 GHz hexacore processor). With a time horizon of $T = 1$ s, the MPC loop computes feedforward trajectories at an average update rate of 70 Hz (in free-motion, when no object state is augmented to the MPC formulation). Both the whole body planner and controller obtain their feedback from a state-estimator that fuses encoder readings and IMU measurements to estimate the base pose. The WBC along with the state-estimator constitute the main control loop which runs at 400 Hz. Finally, the low-level module communicates back-and-forth data with the drive controller, where the actuator torque commands are generated at a frequency of 2.5 kHz:

$$\boldsymbol{\tau}_a = \boldsymbol{\tau}_j^* + \boldsymbol{K}_p(\boldsymbol{q}_j^* - \boldsymbol{q}_j) + \boldsymbol{K}_d(\dot{\boldsymbol{q}}_j^* - \dot{\boldsymbol{q}}_j). \tag{16}$$

IV. Experimental Results

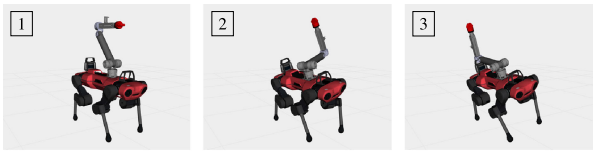
We perform various experiments both in simulation (visualizations of the centroidal dynamics under the MPC policy) and on real hardware. The experiments are divided into two main categories, ones done in free-motion and others involving object-manipulation tasks. All of the examples discussed in this letter are also included in the supplementary video submission (*https://youtu.be/uT4ypNDzUvI*). We start by presenting the generic cost function used in the upcoming test cases:

$$\begin{aligned} L(\boldsymbol{x},\boldsymbol{u},t) = {} & \alpha_1 \cdot \left(||\boldsymbol{r}_{IE} - \boldsymbol{r}_{IE}^{ref}||^2_{\boldsymbol{Q}_{ee_p}} + ||\boldsymbol{\zeta}_{IE}||^2_{\boldsymbol{Q}_{ee_o}}\right)\\ & + \alpha_2 \cdot ||\boldsymbol{x}_r - \boldsymbol{x}_r^{ref}||^2_{\boldsymbol{Q}_r} + \alpha_3 \cdot ||\boldsymbol{x}_o - \boldsymbol{x}_o^{ref}||^2_{\boldsymbol{Q}_o}\\ & + ||\boldsymbol{u} - \boldsymbol{u}^{ref}||^2_{\boldsymbol{R}} \end{aligned} \tag{17}$$

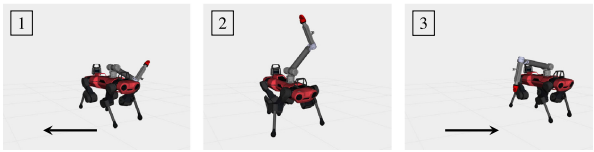
where $\boldsymbol{R}$ is positive definite and $\boldsymbol{Q}_r$, $\boldsymbol{Q}_o$, $\boldsymbol{Q}_{ee_p}$, $\boldsymbol{Q}_{ee_o}$ are positive semi-definite weighting matrices. The vectors $\boldsymbol{r}_{IE}, \boldsymbol{\zeta}_{IE} \in \mathbb{R}^3$ correspond to the arm's end-effector position and the orientation error (represented with exponential coordinates), respectively. Whereas the parameters $\alpha_1, \alpha_2, \alpha_3 \in \{0, 1\}$ are used to switch between different objectives depending on the nature of the task.

A. Free-Motion

In this section, we set $\alpha_3 = 0$ for all experiments. The first set of examples focuses on commanding the base motion which is already part of the state; thus we set $\alpha_1 = 0$ and $\alpha_2 = 1$ initially. We start with a simple test on the real system, where we demonstrate two different dynamic gaits, namely a trot and flying trot, whose mode sequences are depicted in the video. A high penalty is imposed on the arm joint positions and velocities to keep it at a nominal configuration, thereby leading the planner to treat it as a lumped mass with respect to the base. It is clear that in such cases the arm does not play any role with regards to balancing. Alternatively, reducing the joint weights allows the planner to exploit the base-limb coupling by using the arm as a "tail" that contributes in balancing the robot. This behavior is showcased in Fig. 4(a) for a static balancing scenario, where we command the base with a 30 $^\circ$ roll-angle while in stance mode, then we lift the left-fore leg. As a result, the arm moves so as to shift the whole-body center of mass in a direction that would redistribute the contact forces equally. Another scenario is shown in Fig. 4(b), where we command the robot to trot sideways with a relatively high velocity in both directions. As the robot accelerates in one direction, the arm swings in the opposing direction, and it does so by rotating about the x-axis of the base frame. Apart from force redistribution, this dynamic movement helps the base accelerate, while additionally counteracting the rolling centroidal angular-momentum induced by the lateral forces (as a consequence of (3) and (5)). This in turn helps regularize the base roll at the desired zero set-point. In fact, we compare the cases of a "rigid" arm and a "tail" arm for a reference lateral displacement, and plot the trajectories for the base roll in Fig. 5. We note a 66.8% reduction in the $\mathcal{L}_2$-norm of the low-weight signal with respect to the high-weight signal. A similar behavior is also exhibited when commanding accelerations in



(a) **1.** Robot starts in a nominal stance mode. **2.** Base is commanded to roll with a 30 degree angle. **3.** Left fore leg is lifted.



(b) **1.** Robot is trotting laterally in one direction. **2.** Robot is commanded to switch directions. **3.** Robot is trotting laterally in the other direction.

Fig. 4. Snapshots demonstrating the use of the arm as a "tail" that aids in balancing. The sequences correspond to (a) static and (b) dynamic balancing scenarios.

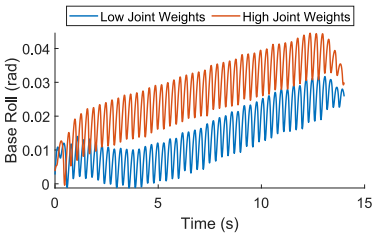


Fig. 5. Plots showing the base's roll during a lateral trot in two scenarios: The arm is treated as a rigidly attached mass (high joint weights) – The arm acts as a balancing "tail" (low joint weights).

the longitudinal directions where the arm's swinging motion helps regularize the base pitch. These examples are presented in simulation only to avoid potential collisions between the arm and base that could damage the platform.

In the second set of experiments we focus on commanding the manipulator's end-effector. Accordingly, a task-space objective is assigned by setting $\alpha_1 = 1$ and $\alpha_2 = 1$, where the quadratic state-cost is used as a regularization term. To begin with, we carry out a task involving the manipulator reaching out to grasp an object or to place it. The perceived behavior is one where the torso clearly adapts its pose in coordination with the arm's movement (see attached video). This indeed highlights the importance of including the robot's full kinematic model in the planning phase. By that we obtain motion plans that would optimally exploit the kinematic redundancy in our system to help achieve the end-effector tracking task. The degrees of freedom that are predominantly employed are determined by the relative weights in the matrix $\boldsymbol{Q}_r$. A second scenario consists of sending references to both the base and the gripper simultaneously. The gripper's position with respect to the inertial frame is fixed, while the base is commanded to trot back and forth. This is done while having the robot carry an unmodeled 2 kg load to illustrate the robustness of our control framework in its ability to overcome modeling errors.

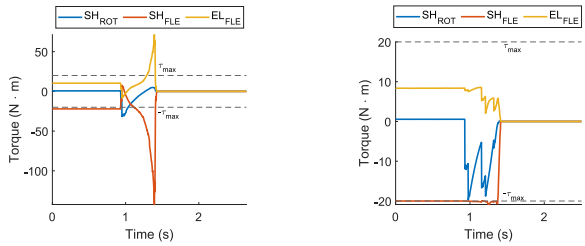


Fig. 6. Plots showing the arm's joint torques for the dynamic throwing task without torque limits *(left)* and with torque limits *(right)*.

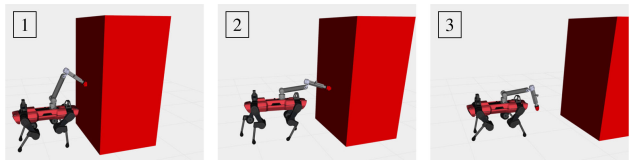


Fig. 7. Snapshots of the robot dynamically pushing a 10 kg load towards a target position.

B. Object-Manipulation

In this section, we adopt an object-centered perspective in our task-specification, meaning that the objective is defined on the level of the object's state, and the robot's optimal motion/force trajectories are generated accordingly. Therefore, we set $\alpha_1 = 0$, $\alpha_2 = 1$ and $\alpha_3 = 1$ during our next experiments. The first set of examples consists of simulations of the following manipulation tasks (see attached video): Dragging (pulling) a 10 kg load, dynamically throwing a 3 kg load, and dynamically pushing a 10 kg load, all towards a target position. The first one has a continuously closed contact state, while the other two have a closed contact initially, followed by an open-contact after a switching time of $t_s = 0.6$ s from the start of the manipulation sequence. The load-dragging video highlights the importance of imposing arm joint-torque limits in the MPC formulation. This is essential for avoiding arbitrarily large and unattainable contact forces at any arm configuration. We note that in both cases (with and without torque constraints) the task is achieved; however, the constrained version results in the robot naturally extending its arm closer to a singular configuration to be able to apply the forces required to drag the load. Moreover, we verify the adequacy of the new inequality-handling algorithm through the torque plots corresponding to the object-throwing task (see Fig. 6). In fact, the system is operated at the boundary of the feasible region without any algorithmic instability issues. Finally, Fig. 7 shows the frame sequences of the dynamic pushing task, where the loss of contact throughout the manipulation period is displayed. It is worth mentioning that the mode schedule for this task could be easily adapted to include three modes (open - closed - open) where the first transition occurs by having the robot strictly follow a pre-defined trajectory towards the object.

In this final example, we perform real hardware experiments on this letter's central manipulation task, namely opening (pushing and pulling) heavy resistive doors. We model the door – which is equipped with a door-closer – as a rotational mass-damper system with a constant recoil torque acting on it. The door angle, which is fed back to the whole-body planner, is estimated by relying on the gripper's position and the door's

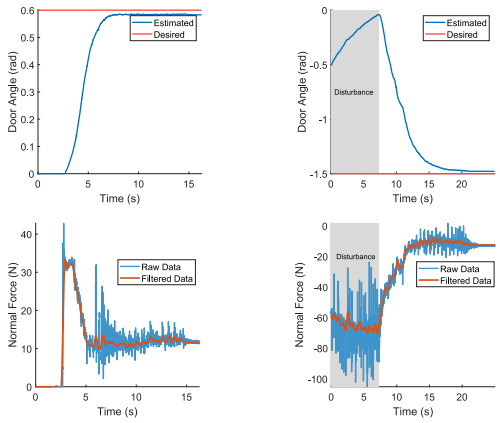


Fig. 8. Plots showing the estimated door angle and the normal force applied to the door for the pushing *(left)* and pulling *(right)* experiments. The door is disturbed during the pulling task.

kinematic parameters. Moreover, due to our planner's inability to inherently account for potential collisions between the robot and the door, we shape a desired robot behavior in the cost function to guide the solver towards collision-free motions. For instance, when pushing the door, we impose a higher penalty on the base's lateral movement to avoid collisions with the door frame. The results for the optimal force trajectories and the estimated door angle are presented in Fig. 8 for the door pushing and pulling tasks. In both scenarios, we observe that the planner discovers physically consistent motion/force plans that drive the door to its desired angle regardless of any model mismatches. The steady-state error could in fact be diminished, and the transient response made faster, by simply increasing the object-state tracking weights and reducing the regularization of the arm's contact forces. To further demonstrate the robustness of our approach, we apply an external disturbance that tends to close the door while the robot is already in the process of pulling it open. During this period, it is clear how the planner still adjusts the contact forces according to the current door angle. When the disturbance vanishes, the robot manages to pull the door to the assigned 90° set-point.

C. Comparative Study

The purpose of this section is to further support the importance of encoding the dynamic coupling between the base, arm, and manipulated object in the MPC formulation. To this end, we compare the centroidal dynamics model adopted in this letter to the single rigid body dynamics (SRBD) template model used in previous works [15], [19], both with and without augmenting the object dynamics. The SRBD model is obtained by assuming a fixed full-body inertia around a nominal robot configuration, and by neglecting the contributions of the joint velocities on the base motion in (5). In all four cases the object dynamic effects are incorporated in the tracking controller, and the MPC cost weights are similar. The comparison is based on a dynamic object-lifting task with a displacement of 1.25m, where various desired lifting times are specified for each case. The times below which the task execution fails are reported in Table I, along with the true settling times, and the average computational times of each MPC iteration. We note from the table and from the corresponding videos that including the dynamic effects of the arm and object in the planner allows for a wider range of fast lifting

TABLE I
COMPARATIVE RESULTS FOR 4 TEMPLATE MODELS DURING THE DYNAMIC LIFTING TASK

	Centroidal with Object	Centroidal without Object	SRBD with Object	SRBD without Object
Minimum Reference Lifting Time (s)	0.2	1.0	1.5	2.0
Settling Time (s)	0.77	3.64	6.75	6.90
Average MPC Computational Time (ms)	19.9	17.2	19.2	14.5

motions. Moreover, we notice a slight decrease in the average computational times when excluding the object's state from the system flow map and/or when using the SRBD approximation. Nevertheless, the best performance was still obtained by the richest model, even if the MPC solver in this case runs at a lower update rate. We note that in this experiment, the failure cases were not caused by any constraint violations, but rather by the SLQ solver's inability to converge to an acceptable solution (i.e., satisfying the assigned tolerances) within the allowable number of function calls. This occurs because at some point, the optimal input trajectory computed at the last MPC iteration leads to a diverging state trajectory during the forward rollout phase of the current iteration. The bigger the discrepancy between the template model and the actual model, the greater the mismatch between the future predicted behavior under the optimal inputs and the measured one, and the more likely it is that the solver fails to converge. Finally, it is worth mentioning that in the case of simply commanding the robot in free-motion with a fixed nominal configuration for the arm and a nominal base orientation, the SRBD model is sufficient.

V. CONCLUSION AND FUTURE WORK

In this work, we proposed, to the best of our knowledge, the first holistic MPC framework that plans whole-body motion/force trajectories for tasks combining dynamic locomotion and manipulation. The underlying multi-contact optimal control problem was formulated as a constrained and switched system that is solved in real-time with a dedicated SLQ algorithm. We defined an extended system model that augments the manipulated-object dynamics to the robot's centroidal dynamics and full kinematics. By that, we were able to account for the coupling effects between the base, limbs, and object in the same planning framework. We demonstrated the effectiveness of our approach on a broad range of test cases with a quadrupedal mobile manipulator performing either free-motion tasks or object-manipulation tasks. We verified that the computed optimal trajectories are physically tractable and respect the system's operational limits as they can be easily tracked by our whole-body controller, and thus easily deployed on the real system. We also validated that the MPC generates solutions fast enough to provide our closed-loop system with robustness properties, which allow it to mitigate the effects of model mismatches or external disturbances. One of the natural extensions to this work would be an MPC framework that is aware of the different body geometries and is thereby able to implicitly plan for an optimal contact schedule based on the potential collisions between these bodies. Another interesting direction would be to make the planner adaptive with respect to the object's dynamic properties. This would be done by fusing the current method with online system-identification techniques in order to resolve the parametric uncertainties in the object's model.

REFERENCES

[1] O. Khatib, "A unified approach for motion and force control of robot manipulators: The operational space formulation," *IEEE J. Robot. Automat.*, vol. 3, no. 1, pp. 43–53, Feb. 1987.

[2] O. Kanoun, F. Lamiraux, and P. Wieber, "Kinematic control of redundant manipulators: Generalizing the task-priority framework to inequality task," *IEEE Trans. Robot.*, vol. 27, no. 4, pp. 785–792, Aug. 2011.

[3] A. Escande, N. Mansard, and P. Wieber, "Hierarchical quadratic programming: Fast online humanoid-robot motion generation," *Int. J. Robot. Res.*, vol. 33, no. 7, pp. 1006–1028, 2014.

[4] S. Kuindersma, F. Permenter, and R. Tedrake, "An efficiently solvable quadratic program for stabilizing dynamic locomotion," *IEEE Int. Conf. Robot. Automat.*, May 2014, pp. 2589–2594.

[5] C. D. Bellicoso, C. Gehring, J. Hwangbo, P. Fankhauser, and M. Hutter, "Perception-less terrain adaptation through whole body control and hierarchical optimization," in *Proc. 16th IEEE-RAS Int. Conf. Humanoid Robots, Humanoids*, 2016, pp. 558–564.

[6] S. Fahmi, C. Mastalli, M. Focchi, and C. Semini, "Passive whole-body control for quadruped robots: Experimental validation over challenging terrain," *IEEE Robot. Automat. Lett.*, vol. 4, no. 3, pp. 2553–2560, Jul. 2019.

[7] M. Neunert *et al.*, "Whole-body nonlinear model predictive control through contacts for quadrupeds," *IEEE Robot. Autom. Lett.*, vol. 3, no. 3, 2018, pp. 1458–1465.

[8] J. Koenemann *et al.*, "Whole-body model-predictive control applied to the HRP-2 humanoid," in *Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst.*, 2015, pp. 3346–3351.

[9] M. Posa and R. Tedrake, "Direct trajectory optimization of rigid body dynamical systems through contact," in *Algorithmic Foundations Robot. X*, 2013, pp. 527–542.

[10] C. Mastalli *et al.*, "Crocoddyl: An efficient and versatile framework for multi-contact optimal control," in *Proc. IEEE Int. Conf. Robot. Automat,* 2020, pp. 2536–2542.

[11] C. D. Bellicoso, F. Jenelten, P. Fankhauser, C. Gehring, J. Hwangbo, and M. Hutter, "Dynamic locomotion and whole-body control for quadrupedal robots," in *Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst.*, 2017, pp. 3359–3365.

[12] P. Sardain and G. Bessonnet, "Forces acting on a biped robot. center of pressure-zero moment point," *IEEE Trans. Syst. Man Cybern. Part A*, vol. 34, no. 5, pp. 630–637, Sep. 2004.

[13] J. D. Carlo, P. M. Wensing, B. Katz, G. Bledt, and S. Kim, "Dynamic locomotion in the MIT cheetah 3 through convex model-predictive control," in *Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst*, 2018, pp. 1–9.

[14] O. Villarreal, V. Barasuol, P. M. Wensing, D. G. Caldwell, and C. Semini, "MPC-based controller with terrain insight for dynamic legged locomotion," in *Proc. IEEE Int. Conf. Robot. Automat,* 2020, pp. 2436–2442.

[15] A. Winkler, D. Bellicoso, M. Hutter, and J. Buchli, "Gait and trajectory optimization for legged systems through phase-based end-effector parameterization," *IEEE Robot. Automat. Lett.*, vol. 3, no. 3, pp. 1560–1567, Jul. 2018.

[16] H. Dai, A. Valenzuela, and R. Tedrake, "Whole-body motion planning with centroidal dynamics and full kinematics," in *Proc. 14th IEEE-RAS Int. Conf. Humanoid Robots*, 2014, pp. 295–302.

[17] D. E. Orin, A. Goswami, and S. Lee, "Centroidal dynamics of a humanoid robot," *Auton. Robots*, vol. 35, no. 2/3, pp. 161–176, 2013.

[18] M. Kudruss *et al.*, "Optimal control for whole-body motion generation using center-of-mass dynamics for predefined multi-contact configurations," in *Proc. 15th IEEE-RAS Int. Conf. Humanoid Robots*, 2015, pp. 684–689.

[19] R. Grandia, F. Farshidian, R. Ranftl, and M. Hutter, "Feedback MPC for torque-controlled legged robots," in *Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst.*, 2019, pp. 4730–4737.

[20] F. Farshidian, M. Neunert, A. W. Winkler, G. Rey, and J. Buchli, "An efficient optimal planning and control framework for quadrupedal locomotion," in *Proc. IEEE Int. Conf. Robot. Automat.*, 2017, pp. 93–100.

[21] F. Farshidian, E. Jelavic, A. Satapathy, M. Giftthaler, and J. Buchli, "Real-time motion planning of legged robots: A model predictive control approach," in *Proc. 17th IEEE-RAS Int. Conf. Humanoid Robot.*, 2017, pp. 577–584.

[22] M. P. Murphy, B. Stephens, Y. Abe, and A. A. Rizzi, "High degree-of-freedom dynamic manipulation," *Proc. Int. Soc. Opt. Eng.*, vol. 8387, pp. 339–348, 2012.

[23] C. D. Bellicoso *et al.*, "ALMA - Articulated locomotion and manipulation for a torque-controllable robot," in *Proc. Int. Conf. Robot. Automat.*, 2019, pp. 8477–8483.

[24] E. Todorov and W. Li, "A generalized iterative LQG method for locally-optimal feedback control of constrained nonlinear stochastic systems," in *Proc. Amer. Control Conf.*, vol. 1, 2005, pp. 300–306.

[25] D. H. Jacobson and D. Q. Mayne, Differential Dynamic Programming, *(ser. Modern Analytic and Computational Methods in Science and Mathematics)*. New York, NY, USA: North-Holland, 1970.

[26] M. Diehl, H. G. Bock, and J. P. Schlöder, "A real-time iteration scheme for nonlinear optimization in optimal feedback control," *SIAM J. Control. Optim.*, vol. 43, no. 5, pp. 1714–1736, 2005.

[27] P.-B. Wieber, *Holonomy and Nonholonomy in the Dynamics of Articulated Motion*, *Fast Motions Biomech. Robot.*, pp 411–425, 2006.

[28] J.-P. Sleiman, F. Farshidian, and M. Hutter, "Constraint handling in continuous-time ddp-based model predictive control," 2021, *arXiv:2101.06067*.

[29] J. Nocedal and S. J. Wright, *Numerical Optimization*. Berlin, Germany: Springer, 2006.

[30] J. Carpentier *et al.*, "Pinocchio: Fast forward and inverse dynamics for poly-articulated systems," 2015, pp. 614–619,. [Online]. Avilabel: https://stack-of-tasks.github.io/pinocchio

[31] J. Carpentier *et al.*, "The pinocchio C++ library: A fast and flexible implementation of rigid body dynamics algorithms and their analytical derivatives," in *Proc. IEEE Int. Symp. Syst. Integrations*, 2019, pp. 614–619.

[32] B. Bell, "CppAD: A package for C algorithmic differentiation,"2019. [Online]. Avilable: http://www.coin-or.org/CppAD