A simple yet effective whole-body locomotion framework for quadruped robots

*Gennaro Raiola, +Enrico Mingo Hoffman, *Michele Focchi, +Nikos G. Tsagarakis and *Claudio Semini

***Abstract*—In the context of legged robotics, many criteria based on the control of the Center of Mass (CoM) have been developed to ensure stable and safe robot locomotion. Defining a whole-body framework with the control of the CoM requires a planning strategy, often based on a specific type of gait and reliable state-estimation. In a whole-body control approach, if the CoM task is not specified, the consequent redundancy can still be resolved by specifying a postural task that sets references for all the joints. Therefore, the postural task can be exploited to keep a well behaved, stable kinematic configuration. In this work, we propose a generic locomotion framework which is able to generate different kind of gaits, ranging from very dynamic gaits such as the trot, to more static gaits like the crawl, without the need to plan the CoM trajectory. Consequently, the whole-body controller becomes planner-free and it does not require the estimation of the floating base state, which is often prone to drift. The framework is composed of a priority-based whole-body controller that works in synergy with a walking pattern generator. We show the effectiveness of the framework by presenting simulations on different types of simulated terrains, including rough terrain, using different quadruped platforms.**

***Index Terms*—whole-body control, legged robots, planning, optimization**

I. Introduction

Over the last few years in the context of legged robots, a lot of effort has been devoted to designing controllers and planners for locomotion. However, most of the time these two elements are considered separately [1], [2], [3]. Typically the controller requires that the trajectory of the CoM is specified to ensure the stability during locomotion[1]. In a different manner from the task that controls the orientation of the base, for which an Inertial Measurement Unit (IMU) can provide reliable measurements, the planning and tracking of the CoM requires a state-estimation algorithm to obtain its linear position and velocity [5], [6]. Even though these algorithms achieve good results by fusing different sources (e.g. leg odometry, vision and inertial measurements) their estimation has the potential to drift due to bias in the sensors, feet slippage, visual occlusions and compliance of the mechanical structure. Moreover, designing trajectories for the CoM is not a trivial task, because, despite satisfying stability constraints, consideration must be taken of the specific kinematic properties of the robot beforehand. For instance, a certain CoM position could correspond to an undesirable kinematic configuration: close to the kinematic limits of the robot, and/or with low leg mobility [7]. To avoid inconvenient kinematic configurations while walking, it is crucial to provide enough mobility and prevent progressive degeneration of the support polygon and thus also keep the joint efforts limited. As a matter of fact, if the support polygon shrinks, the robustness decreases because the legs can lose mobility for future steps. Differently from the trunk orientation task, where the reference orientation usually does not change so frequently (and in general for common gaits like walk and trot is bounded in a way that the joints are always inside their limits), planning a feasible trajectory for the CoM requires a more complex procedure, often involving numerical optimization techniques.

A. Related work

Priorities are a strategy to deal with conflicting tasks where some of them are more critical. This strategy ensures the achievement of high priority tasks at the expense of other tasks with lower priority. In robotics, hierarchical approaches based on priorities were originally introduced for inverse kinematics problems with the works of Whitney [8] and Liegeois [9], and successively by Nakamura [10] and Siciliano [11]. Kathib et al. [12] also implemented them for inverse dynamics control of redundant manipulators involving two task levels: a first level to control the position of the end-effector and another to control the redundant joints. Later on, Sentis et al. [13] extended this approach to humanoid robots in contact with the environment with an arbitrary number of tasks. However, all these works are projection-based[2] and do not allow the enforcement of *explicitly inequality* constraints. On the other hand, the tasks' completion is often bounded by the robot's workspace and its own technological limits that are typically expressed as inequality constraints (e.g. joint limits, actuation limits and friction limits). To take inequality constraints into account, optimization techniques have been introduced to cast the control problem as an optimization problem [14], [15], [16], [17]. In these approaches, the robot dynamics can be imposed as an *equality* constraint to ensure a physically consistent evolution of the robot state variables. Given the quadratic nature of the cost functions involved[3] and the linearity of the constraints, to render the inverse dynamics controller, the resulting optimization problem is typically

*Dynamic Legged Systems Lab, Istituto Italiano di Tecnologia (IIT), Genova, Italy. gennaro.raiola@gmail.com michele.focchi@iit.it claudio.semini@iit.it

+Humanoid and Human Centred Mechatronics Lab, Istituto Italiano di Tecnologia (IIT), Genova, Italy, enrico.mingo@iit.it nikos.tsagarakis@iit.it

[1]Here the term "stability" is intended in the sense defined by Pang and Trinkle [4].

[2]Priorities are achieved by projecting low priority task Jacobians by means of linear operators (i.e. the null-space projector of higher priority task Jacobians).

[3]The cost functions often quantify the error between a desired and a measured value and present it in the form of a squared euclidean norm.

expressed as a Quadratic Program (QP). Alternatively Wensing et al. [18] avoided the linear approximation of the friction cones and encoded them as Second Order Cones leading to the SOQP formulation. More recently, *hard-priorities* have been introduced for these inverse dynamics formulations and different efficient implementations have been proposed by Del Prete [19] and Herzog [20]. Herzog was the first to demonstrate with experiments on humanoid robots the validity of this approach and later the approach was extended on quadruped robots by Bellicoso [21]. Koolen et al. [22] implemented *soft-priorities* and extensively tested this approach on humanoid robots at the Darpa Robotics Challenge (DRC). Salini et al. [23] also implemented soft-priorities providing an effective way to avoid torque discontinuities when the relative importance of tasks was modified or when constraints appeared or disappeared. This made their controller able to adapt to dynamically changing environments. The advantage of hard-priorities is that they ensure a perfect achievement of the highest priority task at the price of a high computational time (e.g. one QP is solved for each priority level), but it can happen that there is no redundancy left to achieve lower priority tasks. Indeed strict priorities can be sometimes so conservative that they may completely block lower-priority tasks. On the other hand, with soft-priorities the program is solved only once, and the computation time mainly depends on the dimension of the set of constraints. However, it is not always easy to define the relative weights for the tasks because a good trade-off must be found between the different terms in the cost function. In addition a scaling of the weights must be considered to account for the different size of the SI units in the cost variables. Finally, the authors have recently proposed the formalisation of prioritized whole-body Cartesian impedance control as a cascade of QPs [24] together with the post-optimization of contact forces [25] in the case of floating-base systems. All these approaches require the specification of the CoM task in (at least) the X and Y directions (see Fig. 1 for frame definition).

B. Proposed Approach and Contribution

Our approach builds on top of previous works [25], [24] on hierarchical Cartesian impedance control with QP optimization. We extended these works by proposing a novel locomotion *framework* that: 1) avoids the specification of a task for the CoM both in terms of planning and control, making the framework *planner-free*, 2) consequently, does not require inputs from a state-estimation algorithm; 3) keeps the robot in a kinematically "appealing" configuration (e.g. far from joint limits, with a good leg mobility); 4) is robust on rough terrain. The approach achieves a synergy between the planner and the controller by exploiting the *hierarchical* nature of our whole-body optimization. Referring to Table I for the priority order of the tasks, we placed the postural task at the lowest priority. The postural task will exploit the Degrees of Freedom remaining from the higher priority tasks to keep the robot close to a preferable *nominal* kinematic configuration (see Fig. 2). This generates a connection between the motion of the trunk and the location of the contacts. Therefore, the postural task acts as a set of "elastic linkages", and determines the *linear* motion of the base, aligning that with the feet while trying to maintain a "nominal" configuration of the robot. Consequently, it eliminates the complexity of designing a CoM trajectory that takes into account the changing shape of the support polygon during locomotion, making the proposed approach planner-free. The locomotion is driven by a terrain-consistent (haptic) *stepping strategy* that selects the footholds to realize a desired velocity command for the robot base. Instead, the orientation of the base is controlled in a separate task at a *higher* priority and will be accommodated by the postural task being this at *lower* priority level. The price to pay for the absence of the CoM task is that the locomotion stability is no longer guaranteed (e.g. the Zero Moment Point (ZMP) could end-up on the boundary of the support polygon). However, this is not a big issue for quadrupeds, if the swing phases are fast enough [26].

To summarize, the contributions of the present work are:

- A *planner-free* locomotion framework for rough terrain that can be implemented in *real-time*. The framework is composed of a hierarchical whole-body inverse dynamics optimization and a walking pattern generator for omni-directional motions. It can handle different gaits such as crawl and trot, and requires only desired base velocities as high-level inputs from the operator. It does not require the specification of any CoM task and it allows to decouple the base orientation from the generation of the swing trajectories.
- an *experimental* contribution where we demonstrate the effectiveness of our locomotion framework in simulation on different quadruped platforms such as Hydraulically actuated Quadruped (HyQ) and ANYmal (ANYmal). Preliminary results were carried out on the real HyQ platform.

C. Outline

The rest of this paper is structured as follows: in Section II we describe the Hierarchical Whole-Body Operational Space formulation that we use to enforce priorities in our locomotion framework. In Section III we present the walking pattern generator while Section IV discusses some details useful for the implementation of the whole-body framework on the real robot. In Section V we present both simulation and preliminary experimental results, and we conclude with Section VI.

TABLE I: Order of Priorities

Task Name	Symbol	Priority
Contact Task	${}^{\mathcal{W}}\mathcal{T}_{c_i}^{\rightarrow}$	1
Trunk Orientation Task	${}^{\mathcal{W}}\mathcal{T}_{bl}^{\angle}$	2
Postural Task	$\mathcal{T}_{\ddot{q}}$	3

II. Whole-Body Operational Space Controller

In this section we introduce the formulation of the *Whole-Body Operational Space Controller* approach we developed.

A. Model, Tasks and Constraints

We describe the configuration of our robotic system using $f + n$ joint variables:

$$\mathbf{q} = \begin{bmatrix} \mathbf{q}_u \\ \mathbf{q}_a \end{bmatrix}, \tag{1}$$

where the values of the first f virtual joints $\mathbf{q}_u$ represent the pose of the (under-actuated) floating-base, using a particular parameterization for the orientation in $SO(3)$[4] and the last $n = 12$ are the angular positions of the actuated joints ($\mathbf{q}_a \in \mathbb{R}^n$). We describe with $\dot{\mathbf{q}} = [\dot{\mathbf{q}}_u^T \quad \dot{\mathbf{q}}_a^T]^T \in \mathbb{R}^{6+n}$ the vector of generalized velocities; the linear and angular velocities of the base, $\mathbf{v}_{fb} \in \mathbb{R}^6$, are provided by a proper floating-base Jacobian:

$$\mathbf{J}_{fb}\dot{\mathbf{q}}_u = \mathbf{v}_{fb}. \tag{2}$$

The dynamic model of the floating-base system in contact with the environment is given by the following:

$$\mathbf{M}(\mathbf{q})\,\ddot{\mathbf{q}} + \mathbf{h}(\mathbf{q}, \dot{\mathbf{q}}) = \mathbf{S}^T\boldsymbol{\tau} + \mathbf{J}_c(\mathbf{q})^T\,\mathbf{F}_c. \tag{3}$$

According to our parameterization, $\mathbf{M} \in \mathbb{R}^{(6+n)\times(6+n)}$ is the joint space inertia matrix, $\ddot{\mathbf{q}} \in \mathbb{R}^{n+6}$ are the joint accelerations, $\mathbf{h} \in \mathbb{R}^{n+6}$ are torques which account for non-linear terms in the dynamics, $\mathbf{S} = [0_{n\times 6} \;\; I_{n\times n}]$ is a selection matrix that accounts for the fact that the floating-base is not actuated, $\boldsymbol{\tau} \in \mathbb{R}^n$ are the actuated torques, finally $\mathbf{J}_c \in \mathbb{R}^{3c\times(n+6)}$ are the Jacobians of the contacts[5] and $\mathbf{F}_c \in \mathbb{R}^{3c}$ is the vector of contact forces expressed in the world $\mathcal{W}$ frame, where c is the number of contacts. We will enforce in our Whole-Body Control (WBC) formulation, the first 6 rows of (3) as a constraint $\mathcal{C}_{fb}$ to have accelerations $\ddot{\mathbf{q}}$ and contact forces $\mathbf{F}_c$ consistent with the dynamics of the system:

$$\mathcal{C}_{fb}: \quad \mathbf{M}_{fb}\ddot{\mathbf{q}} + \mathbf{h}_{fb} = \mathbf{J}_{c,fb}^T\mathbf{F}_c. \tag{4}$$

where $\mathbf{M}_{fb} \in \mathbb{R}^{6\times n+6}$, $\mathbf{h}_{fb} \in \mathbb{R}^6$ and $\mathbf{J}_{c,fb} \in \mathbb{R}^{3c\times 6}$ are the first six rows extracted from (3).

In this work, we aim to simplify the gait generation and to make the whole-body controller independent, as much as possible, from the floating-base state estimation. In order to do so, the world frame $\mathcal{W}$ origin will always be attached to the base origin and we express the orientation tasks w.r.t. this frame. Additionally for the generation of Cartesian feet trajectories (see Section III-B) we define the $\mathcal{H}$ frame. This is the same as the base frame B (e.g. moves with the robot) but with the Z axis parallel to gravity, i.e. aligned to the inertial frame $\mathcal{W}$ (see Fig. 1). This frame is also known as *horizontal frame* [27], [7]. The base frame and horizontal frames can be extrapolated from IMU measurements.

In our whole-body formulation we intend to consider generalized accelerations and contact forces as optimization variables. Therefore a generic Cartesian (6D) acceleration should be expressed in terms of these variables as:

$$\ddot{\mathbf{x}} = \mathbf{J}(\mathbf{q})\,\ddot{\mathbf{q}} + \dot{\mathbf{J}}(\mathbf{q}, \dot{\mathbf{q}})\,\dot{\mathbf{q}}, \tag{5}$$

[4] In particular we use Euler angles to represent the orientation of the base, so $f = 6$. Note that Euler angles can have singularity at 90° pitch, which is an orientation we never consider in our tasks.

[5] We assume our robot establishes *point* contacts with the environment.

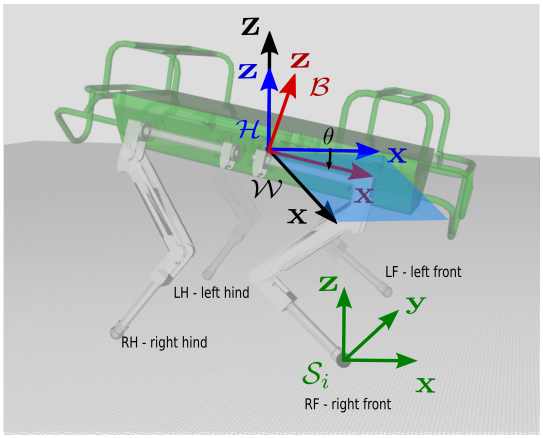


Fig. 1: The figure shows the horizontal frame $\mathcal{H}$, placed at the base link bl but aligned with gravity. The X-axes of the world and of the horizontal frame are co-planar but with different yaw orientations. The swing frame $\mathcal{S}_i$ for the right front leg of the robot, is located at the foot and is aligned with the horizontal frame (on flat terrain). θ is the pitch of the base link.

where $\ddot{\mathbf{x}} \in \mathbb{R}^6$ is a Cartesian acceleration, $\mathbf{J}$ is the task Jacobian and the term $\dot{\mathbf{J}}\dot{\mathbf{q}} \in \mathbb{R}^6$ accounts for the accelerations due to joint velocities. We can define a Cartesian tracking task for (5) as a quadratic cost function $\mathcal{T}$:

$$\begin{aligned} \mathcal{T}: \quad & \|\mathbf{J}\ddot{\mathbf{q}} - \ddot{\mathbf{x}}_r\|^2_{\mathbf{W}}, \\ \ddot{\mathbf{x}}_r = \ddot{\mathbf{x}}_d - \dot{\mathbf{J}}\dot{\mathbf{q}} + \mathbf{K}_P(\mathbf{x}_d - \mathbf{x}) + \mathbf{K}_D(\dot{\mathbf{x}}_d - \dot{\mathbf{x}}), \end{aligned} \tag{6}$$

where $\ddot{\mathbf{x}}_r \in \mathbb{R}^6$ is a reference Cartesian acceleration vector composed by the feed-forward terms $\ddot{\mathbf{x}}_d - \dot{\mathbf{J}}\dot{\mathbf{q}} \in \mathbb{R}^6$ and the feedback terms $\mathbf{K}_P(\mathbf{x}_d - \mathbf{x}) + \mathbf{K}_D(\dot{\mathbf{x}}_d - \dot{\mathbf{x}}) \in \mathbb{R}^6$ that aim to drive the position and velocity tracking error to zero, $\mathbf{K}_P, \mathbf{K}_D \in \mathbb{R}^{6\times 6}$ are positive definite feedback gains matrices. The proper computation of the orientation error between the two poses will be explicitly discussed later on in this section. The matrix $\mathbf{W} \in \mathbb{R}^{6\times 6}$ is the weight matrix associated to the cost function[6].

Considering the generic Cartesian acceleration task in (6), we can define the *contact task* ${}^{\mathcal{W}}\mathcal{T}^{\rightarrow}_{c_i}$ for each foot i in contact:

$$\begin{aligned} {}^{\mathcal{W}}\mathcal{T}^{\rightarrow}_{c_i}: \quad & \|\mathbf{J}_{c,i}\ddot{\mathbf{q}} - \ddot{\mathbf{x}}_r\|^2_{\mathbf{W}}, \\ \ddot{\mathbf{x}}_r = -\dot{\mathbf{J}}_{\mathbf{c,i}}\dot{\mathbf{q}}. \end{aligned} \tag{7}$$

In the contact task we set the reference acceleration to be zero (i.e. the feet in contact do not move). The task is defined w.r.t the world frame $\mathcal{W}$ while the superscript $\rightarrow$ means that only the position part of the task is considered (because we have point feet assumption). We do not set any feedback gain at this level because we do not want to have dependency on the state

[6] Note that, in general, the size of the weight matrix $\mathbf{W}$ depends on the size (number of *rows*) of the task.

estimation, that is often prone to drift [7], while the inter-feet distance will be ensured by the postural task. Contact tasks at the feet are needed to ensure the emergence of the contact forces in (4) needed to compensate for the gravity load acting on the floating base of the robot.

We consider another Cartesian task ${}^{\mathcal{W}}\mathcal{T}_{bl}^{\angle}$ to control the orientation of the *base* of the robot:

$$\begin{aligned} {}^{\mathcal{W}}\mathcal{T}_{bl}^{\angle}: \quad & \|\mathbf{J}_{bl}\ddot{\mathbf{q}} - \ddot{\mathbf{x}}_r\|_{\mathbf{W}}^2, \\ \ddot{\mathbf{x}}_r = \dot{\boldsymbol{\omega}}_d - \dot{\mathbf{J}}_{bl}\dot{\mathbf{q}} - \mathbf{K}_{P,o}\mathbf{e_o} + \mathbf{K}_{D,o}&\left(\boldsymbol{\omega}_d - \boldsymbol{\omega}\right). \end{aligned} \tag{8}$$

Where $\mathbf{e_o}$ is the orientation error computed through quaternions[8]. The task is defined w.r.t the world frame $\mathcal{W}$ while the superscript $\angle$ means that only the orientation part of the task is considered. In (8), $\boldsymbol{\omega}_d, \boldsymbol{\omega} \in \mathbb{R}^3$ are the desired and measured angular velocity, respectively and $\dot{\boldsymbol{\omega}}_d$ is the desired angular acceleration.

To track posture references, we define a postural task $\mathcal{T}_{\ddot{q}}$:

$$\begin{aligned} \mathcal{T}_{\ddot{q}}: \quad & \|\ddot{\mathbf{q}}_a - \ddot{\mathbf{q}}_r\|_{\mathbf{W}}^2, \\ \ddot{\mathbf{q}}_r = \mathbf{K}_{P,p}&\left(\mathbf{q}_{a,d} - \mathbf{q}_a\right) - \mathbf{K}_{D,p}\left(\dot{\mathbf{q}}_{a,d} - \dot{\mathbf{q}}_a\right), \end{aligned} \tag{9}$$

defined just for the *actuated* part of (1). The posture references aim to keep the robot in a well behaved kinematic configuration as in Figure 2. These references can be changed if the user wants to set a different height for the robot [9]. To ensure

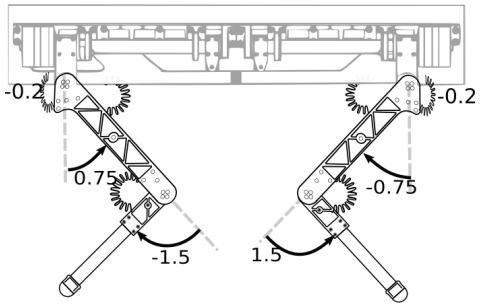


Fig. 2: Nominal configuration used as reference for the postural task (values shown in radians).

contact stability, it is common to constrain the contact forces to lie in a *linearized friction cone* $\mathcal{C}_{fc_i}$ for each contact:

$$\mathcal{C}_{fc_i}: \quad \begin{cases} F_{c_i,\mathrm{n}} \geq 0, \\ |\mathbf{F}_{c_i,\mathrm{t}}| \leq \frac{\sqrt{2}}{2}\mu_i F_{c_i,\mathrm{n}}, \end{cases} \tag{10}$$

[7]Even in the case the CoM position is estimated via *relative* measurements (based solely on relative encoders readings that are quite accurate), the occurrence of slippage in the foot chosen as reference, can make the estimation suffer from drift.

[8]The orientation error can be computed from the quaternion error $\boldsymbol{\alpha_e} = \left[\eta_e, \boldsymbol{\epsilon_e}\right]$ as $\mathbf{e_o} = 2arccos(\eta_e)\frac{\boldsymbol{\epsilon_e}}{\|\boldsymbol{\epsilon_e}\|}$. The quaternion error is computed from a desired orientation quaternion $\boldsymbol{\alpha_d} = [\eta_d, \boldsymbol{\epsilon_d}]$ and the actual orientation quaternion $\boldsymbol{\alpha_a} = [\eta_a, \boldsymbol{\epsilon_a}]$ as in [28], $\boldsymbol{\alpha_e} = \left[\eta_a\eta_d + \boldsymbol{\epsilon_a}^T\boldsymbol{\epsilon_d}, \eta_a\boldsymbol{\epsilon_d} - \eta_d\boldsymbol{\epsilon_a} - \boldsymbol{\epsilon_a}\times\boldsymbol{\epsilon_d}\right]$.

[9]Note that this configuration works well for flat or moderate terrain inclinations and might pose stability issues for bigger inclinations; to improve robustness, it would be possible to make the robot lean forward when climbing up a ramp, as presented in [7]. Improving the postural task in this regard is part of future work.

where $F_{c_i,\mathrm{n}}$ is the normal component, $\mathbf{F}_{c_i,\mathrm{t}} \in \mathbb{R}^2$ are the tangential components of the contact force at foot i and μ_i is the friction coefficient. We also set some bounds on the contact forces:

$$\mathcal{C}_{F_c} := \underline{\mathbf{F}}_c \leq \mathbf{F}_c \leq \overline{\mathbf{F}}_c, \tag{11}$$

and on the joint accelerations:

$$\mathcal{C}_{\ddot{q}} := \underline{\ddot{\mathbf{q}}} \leq \ddot{\mathbf{q}} \leq \overline{\ddot{\mathbf{q}}}. \tag{12}$$

Notice that the limits in (11) are chosen to be feasible upper and lower bounds w.r.t. the limits in (10).

B. Inequality Hierarchical Quadratic Programming (iHQP)

With all the *ingredients* presented before we set-up a cascade of constrained QP problems in the variables $\mathbf{x} = \left[\ddot{\mathbf{q}}^T,\ \mathbf{F}_C^T\right]^T$:

$$\begin{aligned} \mathbf{x}_{\mathrm{k}}^* = \underset{\mathbf{x}_{\mathrm{k}}}{\operatorname{argmin}}\ & \|\mathbf{A}_{\mathrm{k}}\mathbf{x}_{\mathrm{k}} - \mathbf{b}_{\mathrm{k}}\|_w^2 + \lambda\|\mathbf{x}_{\mathrm{k}}\|^2 \\ \text{subject to} \\ & \underline{\mathbf{c}}_{\mathrm{k}} \leq \mathbf{C}\mathbf{x}_{\mathrm{k}} \leq \overline{\mathbf{c}}_{\mathrm{k}} \\ & \underline{\mathbf{u}}_{\mathrm{k}} \leq \mathbf{x}_{\mathrm{k}} \leq \overline{\mathbf{u}}_{\mathrm{k}} \\ & \mathbf{A}_{\mathrm{j}}\mathbf{x}_{\mathrm{j}}^* = \mathbf{A}_{\mathrm{j}}\mathbf{x}_{\mathrm{k}} \end{aligned} \tag{13}$$

given a generic task $\mathbf{A}_{\mathrm{k}}\mathbf{x} = \mathbf{b}_{\mathrm{k}}$ subject to the constraint $\mathbf{C}_{\mathrm{k}}$ and bounds, for the k-th level of priority. The equality constraint enforces the priorities from all the previous j levels, with $j = 0, \ldots, k-1$. Notice that $\mathbf{x}_{\mathrm{j}}^*$ is the solution given by the previously solved QPs. The second term in the cost function is a regularization term for the k-th level through the λ gain. A regularization term on the ground reaction forces is mandatory to prevent ill-conditioning of the Hessian, avoiding instability in the solution.

In addition, the regularization of the contact forces can be used to prevent the solver from generating a solution with unnecessarily high forces or to increase robustness. For instance, in [29] regularisation is used to find a solution where the forces try to be far away from the friction cone boundaries while in [10] are inserted as a last priority task to minimize internal forces.

C. Stack of Tasks

The iHQP problem in (13) is used to solve two different *Stack of Tasks*, composed of the tasks and the constrains we introduced in the previous section. Despite the fact that the introduced whole-body control framework is generic w.r.t. the type of tasks and the number of priorities, we focus on only two kinds of stacks.

We first introduce the stack $\mathcal{S}_3$ constituted by 3 levels of priorities:

$$\mathcal{S}_3 := \begin{pmatrix} \sum_{i\in I_{st}} {}^{\mathcal{W}}\mathcal{T}_{c_i}^{\rightarrow} \,/ \\ {}^{\mathcal{W}}\mathcal{T}_{bl}^{\angle} \,/ \\ \mathcal{T}_{\ddot{q}} \end{pmatrix} \ll \mathcal{C}_{fb} \ll \mathcal{C}_{fc} \ll \mathcal{C}_{\ddot{q}} \ll \mathcal{C}_{F_c}, \tag{14}$$

where the "/" symbol implies a null-space relation between the cost functions (*hard* hierarchy) and the " $\ll$ " symbol

considers the insertion of constraints, in this case, to all the priority levels. Notice that we enforce contacts (I_{st} is the set of the indexes of the stance feet) as the first priority level, while in the secondary level we control the orientation of the base. Finally a postural task attracts the posture of the whole robot to a *nominal* reference. All these tasks are subject to be consistent with dynamics, friction cones, joint acceleration limits and force limits.

The second stack $\mathcal{S}_1$ consists of a single level of priority constituted by a constrained weighted sum of tasks (*soft* hierarchy):

$$\mathcal{S}_1 := \left(\sum_{i \in I_{st}} {}^{\mathcal{W}}\mathcal{T}_{c_i}^{\rightarrow} + {}^{\mathcal{W}}\mathcal{T}_{bl}^{\swarrow} + \mathcal{T}_{\ddot{q}} \right) \ll \mathcal{C}_{fb} \ll \mathcal{C}_{fc} \ll \mathcal{C}_{\ddot{q}} \ll \mathcal{C}_{F_c}, \tag{15}$$

where the $+$ operator is used to sum cost functions.

As a matter of fact, strict hierarchies in $\mathcal{S}_3$ eliminate inconsistencies which may be generated by employing a single priority level, for example breaking contacts due to motion of the trunk. However, they can sometimes be too conservative so that they may completely block lower-priority tasks. On the other hand, classic implementation of strict priorities, as in $\mathcal{S}_3$, rely on resolving a cascade of QPs which augment the computational cost w.r.t a single priority level. Furthermore, up to a certain extent, it is possible to tune relative weights in $\mathcal{S}_1$ so that the behaviour will be similar, but not exactly the same, as in $\mathcal{S}_3$ (if the ratio of the weights of different priority levels is sufficiently high, i.e. at least 10^3 [10]). Therefore, $\mathcal{S}_1$ is preferable for a real-time compatible implementations but, due to the presence of the weights, it becomes harder to fine-tune the different task contributions.

It is important to note that the presented approach does not rely on explicit control of the CoM of the robot. For stability purposes we instead rely on the postural task which acts in the final layer of $\mathcal{S}_3$, or at low priority in $\mathcal{S}_1$. The postural task will move the robot to a nominal configuration by exploiting the remaining Degrees of Freedom from the higher priority tasks, thus resulting in a motion that aligns the base with the stance feet. Avoiding direct control of the position of the CoM of the robot also has the advantage of achieving automatic adjustment of the base in the presence of uneven terrain, as will be shown later.

The outputs of the QP, used to solve the problem described in (14) and (15), are optimal joint accelerations $\ddot{\mathbf{q}}^*$ and contact forces $\mathbf{F}_c^*$ that, plugged in (3), return the reference torques $\boldsymbol{\tau}^*$. These will be the inputs of a low-level torque controller active at the joints of the robot. Figure 3 shows a block diagram of the components of the framework.

It is worth pointing out that the role of the inertia matrix $\mathbf{M}$, which multiplies the joint accelerations $\ddot{\mathbf{q}}^*$ (the controllers are written at the acceleration level), works as a time-varying, non-linear and non-diagonal gain matrix acting on the feedback gains of the tasks, which are, most of the time, diagonal. This has the final effect of creating coupling between joints. A Cartesian task further increases the coupling because a Cartesian error is spread on several joints. This is an issue in the case where the tracking of a joint is worse than the others[11]. However, through a particular choice of the weight matrix and the feedback gains of the tasks it is possible to get back the diagonal gains achieving an equivalent Cartesian impedance controller, see Appendix A.

[10]This is a rule of "thumb" that works well in practice considering the tasks' normalized costs in SI units.

D. State Estimation and Contact Estimation

1) Independence of Constraints from State Estimation: As stated in the previous section, to be robust w.r.t state estimation drift, our intention is to drop the dependency from the *position* of the floating base w.r.t the inertial frame. However, the orientation of the floating base and its angular velocity are still estimated using an IMU sensor. Neglecting the linear position (and velocity) of the base is analogous to continuously resetting the world frame origin to the base origin.

However, equation (7) for the contact, is a constraint $\mathbf{J}_c\ddot{\mathbf{q}} = -\dot{\mathbf{J}}_c\dot{\mathbf{q}}$ that should be written in an inertial frame (not in the base frame), because the velocity of a foot depends not only on joint velocities but also on the motion of the floating base. Therefore, we are considering the floating base part in the $\mathbf{J}_c$ Jacobian. Apparently, it might seem that the linear part $\dot{\mathbf{x}}_b$ of the base twist in $\dot{\mathbf{q}} = [\dot{\mathbf{x}}_b^T \quad \boldsymbol{\omega}^T \quad \dot{\mathbf{q}}_j^T]^T$ is required (while we consider it to be zero).

If we carefully inspect the structure of $\dot{\mathbf{J}}_c$, we notice that this matrix has columns of zeros multiplying the $\dot{\mathbf{x}}_b$ variables, because $\mathbf{J}_c$ depends only on base orientation and on $\mathbf{q}_j$ but not on base linear position. This makes the term $\dot{\mathbf{J}}_c\dot{\mathbf{q}}$ dependent on $\boldsymbol{\omega}$ but not on $\dot{\mathbf{x}}_b$. Therefore considering $\dot{\mathbf{x}}_b = \mathbf{0}$ is not affecting the validity of equation (7). The $\dot{\mathbf{x}}_b$ seems to appear also in the dynamic equation (3) (inside $\dot{\mathbf{q}}$) that we enforce as equality constraint. However the term $\mathbf{h}(\mathbf{q}, \dot{\mathbf{q}})$ is also independent from the base linear velocity making also this constraint unaffected. For the above reasons, the theoretical foundation of our approach is still perfectly valid even if we consider both $\mathbf{x}_b$, $\dot{\mathbf{x}}_b = \mathbf{0}$, making our locomotion independent from the need of a state-estimation algorithm.

2) Floating base height estimation: To be able to control the height of the robot it is necessary to obtain an estimation of it. Differently from the X and Y coordinates, the base height can be considered as the average *relative* position of the feet in the $\mathcal{H}$ frame, which can reliably be estimated through the forward kinematics of the feet ${}^{\mathcal{B}}\mathbf{x}_{c_i}$.

$${}^{\mathcal{H}}\mathbf{x}_{c_i} = {}^{\mathcal{H}}\mathbf{R}_{\mathcal{B}}\, {}^{\mathcal{B}}\mathbf{x}_{c_i}, \tag{16}$$

$${}^{\mathcal{H}}\mathbf{x}_{bl,z} = -\frac{1}{N} \sum_{i \in I_{st}} {}^{\mathcal{H}}\mathbf{x}_{c_i,z} \tag{17}$$

where the rotation matrix ${}^{\mathcal{H}}\mathbf{R}_{\mathcal{B}}$ encodes the orientation of the base w.r.t the $\mathcal{H}$ frame which can be easily measured with

[11]In the case of the knee joint of HyQ, due to the low inertia of the link, the torque sensor barely measures any torque during the swing (when there is no contact), resulting in a open feedback loop. For further details see Section IV.

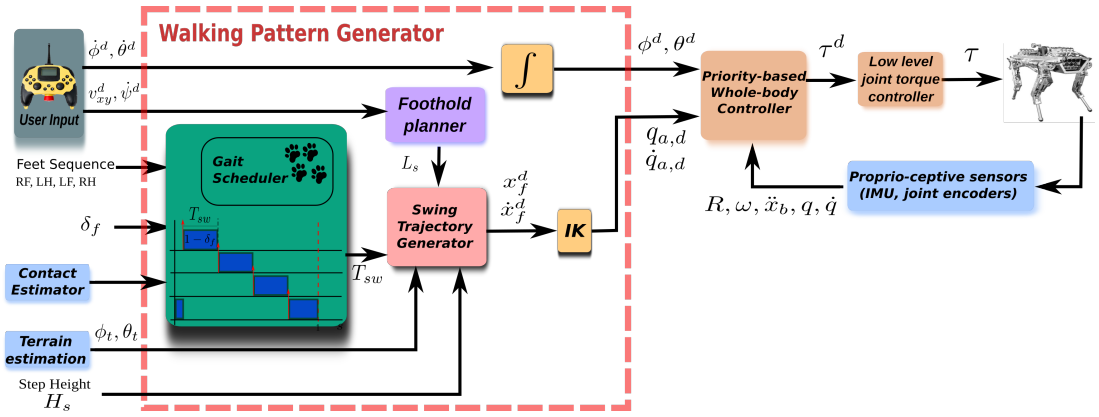


Fig. 3: Block diagram of the locomotion framework. The user gives high-level velocity inputs to the foothold planner, that, on its behalf, is triggered by the gait scheduler. The swing trajectory generator computes the trajectories for the swinging feet, given the step length $\boldsymbol{\Delta}\bar{\boldsymbol{L}}_{xy}$ provided by the foothold planner for each swing leg.

the IMU, and I_{st} is the set of the indexes of the stance feet and N their number.

3) Contact estimation: In order to understand which are the active contacts, we rely on contact force estimation based on torque readings extracting the leg equation from (3)[12]:

$$\mathbf{F}_{c_i} = -\mathbf{J}_{c_i}^{-T}(\boldsymbol{\tau}_{c_i} - \mathbf{h}_i), \tag{18}$$

where $\mathbf{F}_{c_i} \in \mathbb{R}^3$ is the estimated contact force of one leg, $\mathbf{J}_{c_i} \in \mathbb{R}^{3\times 3}$ is the leg Jacobian and $\boldsymbol{\tau}_{c_i}$ are the measured torques in one leg and $\mathbf{h}_i \in \mathbb{R}^3$ is the Coriolis/Centrifugal and gravity bias. When the projection of $\mathbf{F}_{c_i}$ along the normal to the terrain overcomes a certain threshold [7], we consider the leg to be in contact with the environment.

III. Walking Pattern Generator

The walking pattern generator receives desired twist commands for the base of the robot from an external source such as an operator device or a high level planner[13], and transforms these into swing trajectories for the legs given a specific type of gait. In order to do so, the walking pattern generator is composed by A) a gait scheduler that sequences the footsteps based on the gait, B) a foothold planner which transforms the desired base twist commands into footholds by using the horizontal frame $\mathcal{H}$ as reference frame, C) a swing trajectory generator which takes the foothold coordinate and the desired step height as inputs, and calculates the swing trajectory, D) an inverse kinematics transformation to map the leg's trajectories from Cartesian to joint space. Note that we decided to implement the swing trajectory in the joint space rather than in the Cartesian space because of the coupling issues described in the previous section.

[12] For the sake of simplicity, we discard the acceleration terms because their influence on the force computation is very low. However, in case the acceleration terms can not be neglected, they can be incorporated in the equation after filtering. The filtering is necessary due the presence of quantization noise, which would otherwise be amplified by the double differentiation of the encoder measurements.

[13] In our experiments we used a joy-pad connected to the operator's pc.

A. Gait scheduler

Each different gait can be simply defined as a timed sequence of footsteps. Therefore, given a type of gait, the role of the scheduler is to trigger the sequence of leg swings, see Fig. 4.

A state machine is associated to each leg to keep track of its state. The possible states and transitions are depicted in Fig. 5.

B. Foothold planner

The foothold planner calculates the desired foothold coordinates (X and Y) in the $\mathcal{H}$ frame (see Fig. 6). Choosing such reference frame for the foothold selection makes the swing trajectory generation *independent* from the roll and pitch orientation of the base. Henceforth, unless specified, we assume all the vectors are expressed in that frame. The foothold coordinates are computed starting from the *desired* linear $\begin{bmatrix} v_x^d & v_y^d \end{bmatrix}$ and yaw angular velocity $\dot{\psi}^d$ of the base. These two velocities are transformed into foot displacements $\boldsymbol{\Delta}\mathbf{L}_{xy0} \in \mathbb{R}^2$, (see Fig. 7 and equations (19), (20)). These deltas are not added to the previous foot position but to a *virtual foothold* offset that is computed with respect to the *actual* position of the base. This "robo-centric" foothold selection is an important feature to increase the robustness when dealing with rough terrain because it avoids accumulation of errors that would appear if the steps are taken w.r.t the previous foot positions and allows to keep the robot close to a preferred kinematic configuration (the absence of this mechanism would make the robot legs stretch or compress).

The linear part of mapping of the desired velocity command is:

$$\boldsymbol{\Delta}\mathbf{L}_{xy0} = \frac{1}{f_{sw}} \begin{bmatrix} v_x^d \\ v_y^d \\ 0 \end{bmatrix}, \tag{19}$$

which represents the displacement of the foot produced by a linear velocity command, f_{sw} is the step frequency. For the

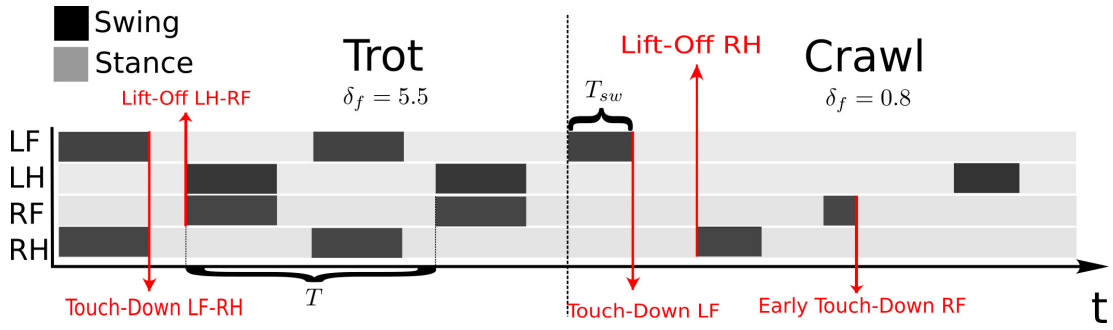


Fig. 4: The left side of the image shows the schedule of a trot gait, while the right side shows a crawl gait. Given the feet sequence, the scheduler triggers the lift-off events for the legs. The touch-down events instead, are triggered if a real contact is detected (*haptic touchdown*) by the contact estimator. The time taken to complete a step cycle is defined as T. The ratio between the stance duration and the cycle duration is defined as the duty factor parameter $\delta_f = T_{st}/T$. Consequently, the swing duration is computed as $T_{sw} = T(1 - \delta_f)$ and the swing frequency as $f_{sw} = 1/T_{sw}$. In this example, the trot has a $\delta_f = 0.55$ while the crawl has $\delta_f = 0.8$, therefore, the crawl keeps the feet for a longer time on the ground. Indeed, a high value for δ_f can be useful for slower gaits such as the crawl to give time to the whole-body controller to recover the posture. Note that during the stance the contacts of the legs are enforced by the whole-body controller. The scheduler can trigger the swing only if the corresponding leg has completed a step cycle.

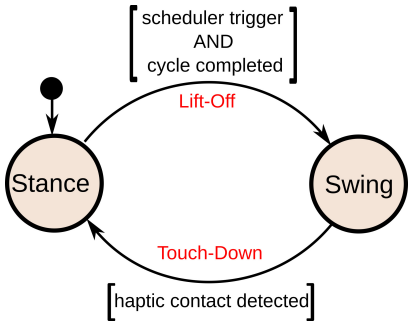


Fig. 5: Each leg starts in the **Stance** state, waiting for the scheduler to trigger the swing. When triggered, the state machine switches from **Stance** to **Swing**. The state machine switches from **Swing** to **Stance**, if a contact is detected by the contact estimator. The step cycle duration T determines the time the leg has to be in the **Stance** state, before it can be triggered by the scheduler for a new swing.

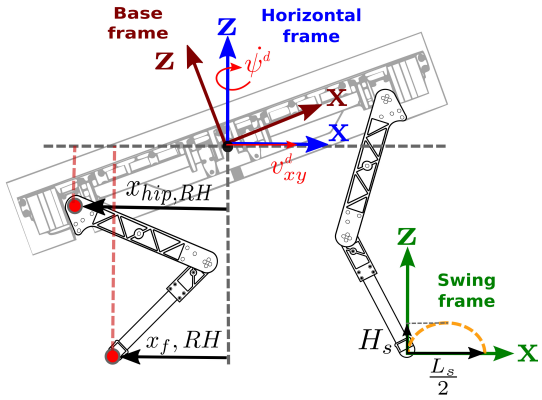


Fig. 6: Frames used by the foothold planner. Desired base twist and foothold positions are expressed in the horizontal frame. Each step is taken with respect to a "virtual foothold" that moves with the base and represents the nominal size of the stance of the gait cycle (e.g. when the desired twist is set to zero)

angular part instead, we have the following:

$$\mathbf{\Delta L}_{h0} = \frac{1}{f_{sw}} \begin{bmatrix} 0 \\ 0 \\ \dot{\psi}^d \end{bmatrix} \times \mathbf{x}_{\text{hip}}, \tag{20}$$

where $\mathbf{x}_{\text{hip}}$ represents the position of the hip of the swinging leg, in the horizontal frame. By summing these two quantities we obtain $\mathbf{\Delta L}_{xy}$:

$$\mathbf{\Delta L}_{xy} = \mathbf{\Delta L}_{h0} + \mathbf{\Delta L}_{xy0} \tag{21}$$

To keep the robot close to a preferred stance configuration, and avoid accumulation of errors, we compute the difference between a preferred *virtual foothold* location defined as $\mathbf{x}_{\text{f}}^0$ and the current one $\mathbf{x}_{\text{f}}$. Therefore, we obtain the following offset:

$$\mathbf{vf}_{xy} = \mathbf{x}_f^0 - \mathbf{x}_f, \tag{22}$$

which is then summed to (21) to obtain the total foot displacement:

$$\mathbf{\Delta \bar{L}}_{xy} = \mathbf{\Delta L}_{xy} + \mathbf{vf}_{xy}. \tag{23}$$

Note that in absence of base command velocities, the effect of the offset (22) is to align the feet to the virtual foothold configuration.

Finally we extract from $\mathbf{\Delta \bar{L}_{xy}}$ the step length and L_s the angle ψ_s of the swing trajectory plane as follows:

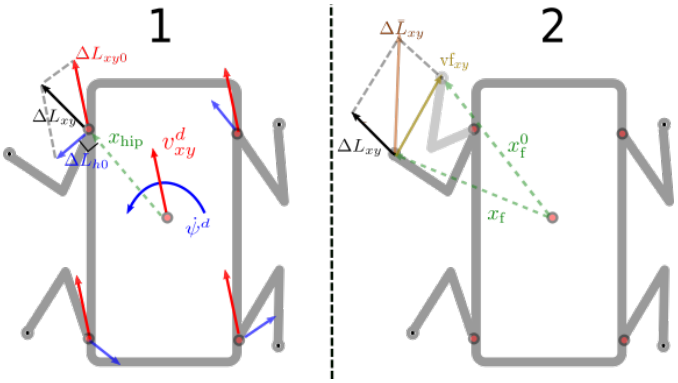


Fig. 7: Top view of the robot with the geometrical explanation of the foot displacements computation: 1) starting from the desired linear $\mathbf{v}^d_{xy}$ and angular $\dot{\psi}^d$ velocities, we compute the corresponding deltas $\mathbf{\Delta L}_{xy0}$ and $\mathbf{\Delta L}_{h0}$. 2) The resulting vector $\mathbf{\Delta L}_{xy}$ is then summed to the virtual foothold vector $\mathbf{vf}_{xy}$ to produce the total foot displacement $\mathbf{\Delta \overline{L}}_{xy}$.

$$L_s = \sqrt{\Delta \overline{L}_x{}^2 + \Delta \overline{L}_y{}^2} \tag{24}$$

$$\psi_s = atan2(\Delta \overline{L}_y, \Delta \overline{L}_x) \tag{25}$$

C. Swing trajectory generator

The swing is generated in the swing frame $\mathcal{S}$ (see Fig. 6 for frame definitions) and is defined as an ellipse built up by mean of sine functions. This has the advantage to easily create a reaching motion and re-plan the trajectory if the user decides to change the swing duration (e.g. by changing the duty cycle or the cycle duration). The swing frame is the same as the horizontal frame (e.g. shares the same yaw orientation) but it is aligned with the terrain and has the origin in the swinging foot. We assume that a terrain estimator is providing the local inclination of the terrain [7] in roll ϕ_t and pitch θ_t. Thanks to the continuous re-computation of the step length L_s and the step heading ψ_s, it is also possible to constantly re-plan the trajectory based on the desired base twist and current base configuration[14] as explained in the previous section III-B.

The trajectory for the swing (expressed in the swing frame $\mathcal{S}$) is computed as follows:

$$\begin{aligned} {}^{\mathcal{S}}x_{f,x} &= \frac{L_s}{2}(1 - cos(\pi f_{sw} t)), \\ {}^{\mathcal{S}}x_{f,y} &= 0, \\ {}^{\mathcal{S}}x_{f,z} &= H_s sin(\pi f_{sw} t), \end{aligned} \tag{26}$$

where L_s and H_s are respectively the step length and the step height, and $t \in [0, T_{sw}]$. While the step length L_s is defined by the foothold planner, the step height can be arbitrarily chosen based on the presence of obstacles or the type of terrain. After mapping (26) to the inertial frame, we obtain:

[14]Assuming smooth input changes.

$${}^{\mathcal{W}}\mathbf{x}_f = {}^{\mathcal{W}}\mathbf{R}_{\mathcal{S}} \; {}^{\mathcal{S}}\mathbf{x}_f, \tag{27}$$

where ${}^{\mathcal{W}}\mathbf{R}_{\mathcal{S}}$ maps vectors from the swing to the inertial frame, and it is defined as a rotation of $\psi_s + \psi$ [15] along the Z axis and a rotation about the X an Y axes of ϕ_t and θ_t (if an estimation of the terrain inclination is available). Since the trajectory for the velocity is defined in a moving frame, its derivative must be computed taking the chain rule derivatives of (27) and (26):

$${}^{\mathcal{W}}\dot{\mathbf{x}}_f = {}^{\mathcal{W}}\dot{\mathbf{R}}_{\mathcal{S}} \; {}^{\mathcal{S}}\mathbf{x}_f + {}^{\mathcal{W}}\mathbf{R}_{\mathcal{S}} \; {}^{\mathcal{S}}\dot{\mathbf{x}}_f, \tag{28}$$

with ${}^{\mathcal{S}}\dot{\mathbf{x}}_f$ defined as:

$$\begin{aligned} {}^{\mathcal{S}}\dot{x}_{f,x} &= \pi f_{sw} \frac{L_s}{2} sin(\pi f_{sw} t), \\ {}^{\mathcal{S}}\dot{x}_{f,y} &= 0, \\ {}^{\mathcal{S}}\dot{x}_{f,z} &= \pi f_{sw} H_s cos(\pi f_{sw} t). \end{aligned} \tag{29}$$

While the timing of the lift-off is dictated by the scheduler, the touch-down is triggered haptically [7] to be sure that the stance is only triggered when a stable foothold is established. During the swing down phase, the occurrence of the contact with the terrain is continuously checked. The swing can continue beyond the planned foothold (reaching motion) until the contact is detected when $t \geq T_{sw}$. The foot is considered in contact with the ground when the ground reaction force overcomes a certain threshold in the direction normal to the terrain. The *haptic* touch-down is a crucial feature to address rough terrain because it prevents to inject destabilizing forces to the base created by the tracking of trajectories that are not terrain consistent.

D. Inverse kinematics

To transform the swing trajectories from Cartesian to joint space we use the Closed Loop Inverse Kinematics (CLIK) algorithm [30]. Therefore, for each swinging leg i we can express the joint velocity as:

$$\dot{\mathbf{q}}_{a_i,d} = \mathbf{J}_{c_i}^{-1} \left[{}^{\mathcal{W}}\dot{\mathbf{x}}_{f_i,d} + \mathbf{P}({}^{\mathcal{W}}\mathbf{x}_{f_i,d} - {}^{\mathcal{W}}\mathbf{x}_{f_i}) \right], \tag{30}$$

where $\mathbf{P} \geq 0$ is the CLIK proportional gain, ${}^{\mathcal{W}}\mathbf{x}_{f_i}$ represents the current foot position w.r.t the inertial frame, ${}^{\mathcal{W}}\mathbf{x}_{f_i,d}$ and ${}^{\mathcal{W}}\dot{\mathbf{x}}_{f_i,d}$ are the desired velocity and position reference provided by the swing trajectory generator through the equations (27) and (28) respectively. $q_{a_i,d}$ is found by integration. In order to control the base height, it is possible to reconfigure the legs in stance. For example, in order to increase the height of the base, the robot's legs must be stretched, while to decrease it, the legs must be retracted. Therefore, we can map the desired base height to the joint positions for the legs i in stance as:

$$\dot{\mathbf{q}}_{a_i,d} = \mathbf{J}_{c_i}^{-1} \; \mathbf{P} \; {}^{\mathcal{W}}\mathbf{\Delta H}_{bl}, \tag{31}$$

[15]we remember that we computed ψ_s w.r.t. the $\mathcal{H}$ frame so we need to consider also the orientation ψ of this frame w.r.t. the $\mathcal{W}$ frame

where ${}^{\mathcal{W}}\mathbf{\Delta H}_{bl} = \left[0, 0, -{}^{\mathcal{W}}\Delta H_{bl,z}\right]$ is a vector that defines the desired change of height for the base of the robot. The value ${}^{\mathcal{W}}\Delta H_{bl,z}$ is defined as:

$$ {}^{\mathcal{W}}\Delta H_{bl,z} = {}^{\mathcal{W}}x_{bl,z,d} - {}^{\mathcal{W}}x_{bl,z}, \tag{32} $$

where ${}^{\mathcal{W}}x_{bl,z,d}$ represents the desired base height and ${}^{\mathcal{W}}x_{bl,z}$ is the actual base height estimated with (17) as described in Section II-D2[16]. Finally, to track the joint space references both for the feet in stance and in swing, we use the postural task $\mathcal{T}_{\ddot{q}}$ introduced in (9).

IV. Implementation details

In this section we present some implementation details and remarks on the final control scheme that we are employing on the real platform.

Swing task: the swing task can be implemented as a Cartesian task or a joint task. From a theoretical point of view, a Cartesian space formulation is more sound because it allows us to set the gains in the same space the trajectories are defined. Conversely, a joint space formulation provides an anisotropic and tilted impedance ellipsoid at the feet, making the legs more compliant in a direction than in another depending on the leg configuration, even with a constant joint stiffness. However, in the implementation on the real HyQ platform, we found issues with the Cartesian implementation of the swing task. The reason is that the Jacobian matrix *couples* the tracking errors of all the joints. This is not a problem if they are *all* able to perform a good tracking. However, with our platform HyQ, the distal limbs of the legs (lower-legs links) are very light and their load-cells measure barely zero-torque during the swing[17] and consequently, the feed-back loop opens. Therefore, the only way to make the joint move, is to create a position error that is big enough to increase the desired torque even if the actual torque remains zero. When implementing Cartesian impedance control algorithms for the swing legs, this peculiarity of the lower leg joint affects performance as well the other joints in the leg. Conversely, with the joint space implementation we are able to avoid this coupling between the joints. For this reason, we have chosen a joint space implementation and the swing references are sent directly to the postural task. However, with this implementation, the coupling due to the inertia matrix is still present, and the matrices $\mathbf{K}_P$, and $\mathbf{K}_D$ assume the meaning of acceleration gains rather than joint impedance gains. To avoid this problem, it is possible to pre-multiply these gains with the inverse of the inertia matrix, giving them the physical meaning of joint *stiffness* and *damping* (see Appendix A). This formulation turns out to be beneficial to improve the tracking of the swing legs.

Force and acceleration bounds: To handle contact transitions, during stance and swing phases, we impose contact force constraints to switch between a maximum allowed value and zero. This allows to keep the size of the QP problem constant during the whole locomotion phase, which can be useful in a hard real-time implementation. To prevent torque discontinuities it is possible to implement a smooth unloading / loading by setting a time-varying upper bound on the contact force as in [29]. During preliminary experiments performed on the real robot, we found that there is a strong influence between the acceleration bounds and the tracking accuracy. In particular, setting the limits too low results in an *overshoot* with the tracking of the desired trajectory at the touchdown (when there is the biggest deceleration). This problem appears only on the real robot and is not present in simulation, because in this second case, the tracking errors are smaller.

TABLE II: Parameters used in the controller

Hessian Regularization Factor	η	$1e-6$
Force Max X	f_x	1000 [N]
Force Max Y	f_y	1000 [N]
Force Max Z	f_z	1000 [N]
Force Min X	f_x	1000 [N]
Force Min Y	f_y	1000 [N]
Force Min Z	f_z	20 [N]
Accel Max	$\ddot{q}_{max}$	500 $[rad/s^2]$
CLIK gain	P	10 $[1/s]$
Swing Frequency	T_{sw}	2 $[Hz]$
Step Height	H_s	0.1 $[m]$

The acceleration and the force limits (active only during the stance phase), are summarized in Table II together with the other parameters set in the controller.

Haptic Touch-Down event: To keep spurious contact estimations from triggering a premature touch-down in the leg's state machine, it is possible to disable the haptic contact detection during the swing up phase (half of the swing time). To detect the touchdown the threshold on the ground reaction forces is set to 50N (see Section II-D).

Loop frequency: The output of the whole-body controller is given as a desired torque to the low-level torque controller, in a cascade loop architecture. Both the whole-body controller and the low-level torque controller run at 1 kHz. In the implementation on the real robot, the trunk controller damping is limited to a max of 400 Nmd/rad to avoid instabilities, because the loop frequency is known to limit the maximum value for the damping [31].

Solver and Computation time: To solve the stack S_3, we used the solver qpOASES [32], leveraging on the whole-body control framework OpenSoT [33]. With an Intel Quad-Core i5-4440 CPU @ 3.10GHz (onboard) machine, it requires on average 1180$\pm$20 μs to solve the 3 layers. Conversely, with the single stack S_1 the computation time drops to 830$\pm$20 μs, making this implementation preferable to be run at 1 kHz. It is worth noting that most of the optimization time is spent in calculating the Hessian.

Terrain estimator: If a terrain estimation algorithm [7] is available, a reference can be given to the orientation task to align the base with the slope of the ground and prevent

[16]Both values can be expressed either in the world or horizontal frame since the two differ only for a rotation around the Z axis.

[17]Load-cells and torque sensors in legged robots are sized in order to be able to measure big torques during the stance, for this reason they lack accuracy during the swing.

reaching the kinematic limits of the leg. However, in the absence of a terrain estimator, the base orientation task can be removed from the stack and the postural task can be used to achieve some sort of terrain adaption, because it will attempt to align the base with the feet.

V. Experiments

In this section we present some experiments to demonstrate the effectiveness of our whole-body framework for quadruped robots (see the accompanying video[18] and Fig. 8 for a summary of all the experiments). The simulations have been carried out with the Robot Operating System (ROS) in a Gazebo environment[19] that uses the ODE physics engine [34]. A friction coefficient of $\mu = 0.8$ was set (unless specified) in all the experiments.

We tested our approach on two different quadruped platforms (HyQ and ANYmal) of different sizes and weights. The porting to a different platform required only a slight tuning of the gains of the postural and of the trunk orientation tasks. In a first simulation performed with HyQ we show in the video that the robot is able to seamlessly switch between a crawl and trot. The robot is traversing a rough terrain area made of ruins and cobble-stones, moving omni-directionally. Notice that the robot is *blind* and not aware of the status of the terrain. To demonstrate the motion decoupling capability of our framework, the robot performs a walk on flat terrain while changing the base orientation and the height. Fig. 9 shows the tracking of the base orientation and of the height in the upper plots, while in the lower plots is reported the tracking in Cartesian space for the LF and RH feet. The gains used for the Cartesian and postural tasks are reported below. For the base orientation (8) we set $\mathbf{K}_P{=}diag([1000.0,\ 1000.0,\ 1000.0])$ and $\mathbf{K}_D{=}diag([100.0,\ 100.0,\ 100.0])$. For the postural task (9) the gains are scheduled depending on the walking phase: for the *swing* phase we set $\mathbf{K}_{P_{sw}}{=}diag([300.0,\ 300.0,\ 300.0])$, $\mathbf{K}_{D_{sw}}{=}diag([8.0,\ 12.0,\ 5.0])$, while for the *stance* phase $\mathbf{K}_{P_{st}}{=}diag([500.0,\ 500.0,\ 500.0])$, $\mathbf{K}_{D_{st}}{=}diag([20.0,\ 20.0,\ 20.0])$. To improve tracking for the swing phase it is possible to pre-multiply the gains for the inverse of the inertia matrix (of the leg) (see Appendix A).

The ground truth coming from Gazebo is used to obtain the measurements in the world frame. In both cases, good tracking without steady errors is achieved; indeed the swing tasks and the base orientation task are not conflicting with each other because the latter is written in the horizontal frame which is independent from the base orientation. For completeness we present the mean and the standard deviation of the tracking errors during the whole experiment, in Table III.

We carried out preliminary experiments on the real platform HyQ showing a $2Hz$ trot on flat terrain, in a second moment we control the base orientation to follow some operator desired reference commands given by mean of a joy-pad interface while an external disturbance acts on the robot (see the video). The tracking error has an average of 0.0101 rad with a standard deviation of 0.0102 rad, see Fig. 10.

Remarks: To achieve a successful implementation on the real robot we had to modify the original formulation of the optimization problem, see Section IV.

TABLE III: Mean and standard deviation of the errors

Measurements	mean	std
Roll	0.0050 [rad]	0.0145 [rad]
Pitch	0.0072 [rad]	0.0187 [rad]
Yaw	0.0017 [rad]	0.0043 [rad]
Height	0.0071 [m]	0.0145 [m]
RH - X	0.0303 [m]	0.0186 [m]
RH - Y	0.0203 [m]	0.0128 [m]
RH - Z	0.0016 [m]	0.0027 [m]
LF - X	0.0235 [m]	0.0152 [m]
LF - Y	0.0202 [m]	0.0201 [m]
LF - Z	0.0006 [m]	0.0022 [m]

VI. Conclusions

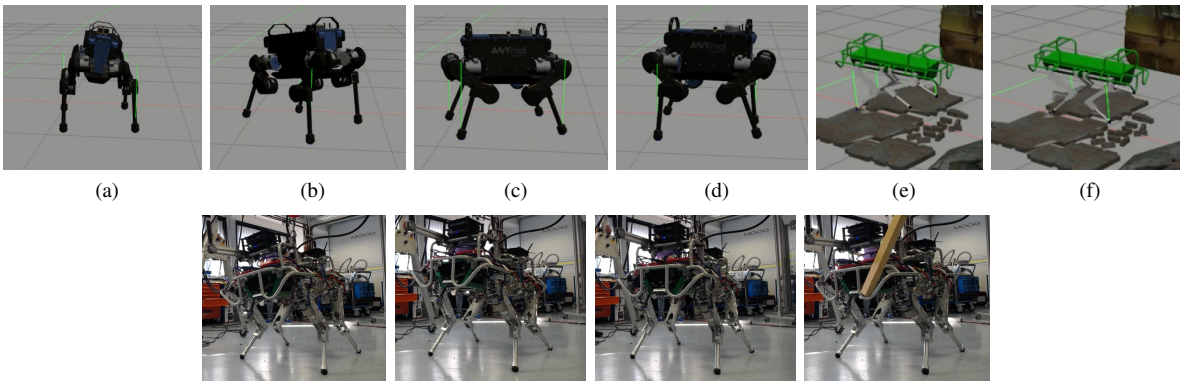
In this work we present a novel locomotion framework for quadrupedal robots that merges a walking pattern generator, acting only at the foot level, with a prioritized whole-body inverse dynamics controller. One of the advantages of the proposed framework is to avoid estimating the linear position and velocity of the floating base, while maintaining the ability to effectively tackle moderately rough terrain. This has been achieved by leveraging the postural task acting in the whole-body controller as a sort of elastic element. Consequently, the robot's base *follows* the feet, resulting in a motion of the trunk that adapts naturally to the foot stance configuration while trying to keep a well-behaved kinematic configuration. To increase the robustness of the proposed approach, the foothold selection is done w.r.t. a *virtual foothold* defined in the *horizontal frame* of the robot making the footstep strategy independent from the base orientation. In this way, no CoM planning is required to implement various types of gaits. However, despite the fact that presented framework is capable to handle uneven terrain, it relies on a particular posture which in turn may need to be properly tuned according to the particular type of terrain being traversed. As part of future work, we plan to further extend the proposed approach by taking into account the presence of a manipulator mounted on the robot's trunk. This would allow operation with complex loco-manipulation tasks. Since our approach is based on mixed *hard-* and *soft*-priorities, we will consider using machine learning techniques in order to properly find optimal weights between the different tasks.

ACKNOWLEDGMENT

We would like to express our gratitude to Dr. Matteo Parigi Polverini and Dr. Arturo Laurenzi for their help and for their valuable suggestions during the preparation. The research

[18]The video with experiments is also available at https://youtu.be/--jpFMhez9g

[19]The controller can be tested at this repository: https://github.com/graiola/wbc-setup



(a) (b) (c) (d) (e) (f)

Fig. 8: First row: snapshots of the locomotion simulations: ANYmal tracking a (a) roll reference, a (b) pitch reference, (c),(d) changing the robot height. HyQ during a (e) crawl swinging only one leg at a time and a trot (f) swinging two legs at the time. Second row: snapshots of the trunk orientation experiments with HyQ. The robot is tracking an orientation reference while being disturbed by an external interaction.

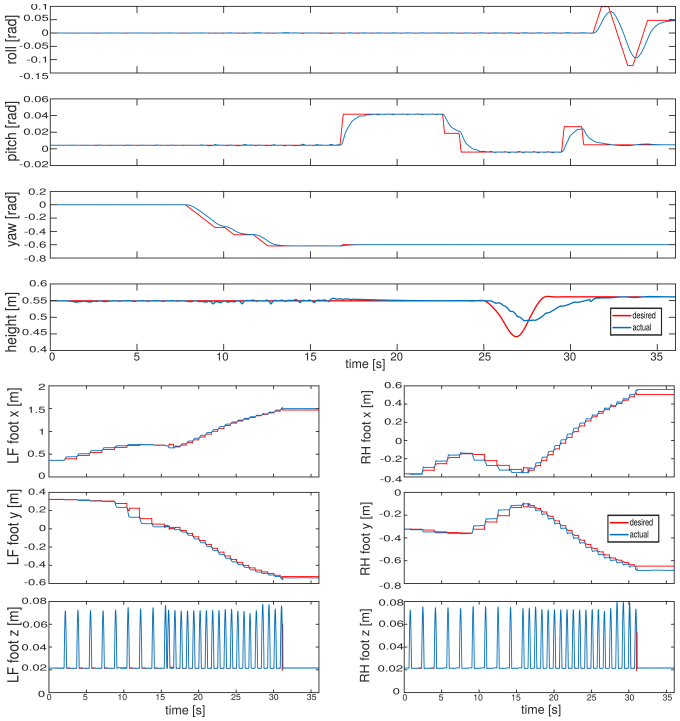


Fig. 9: Simulation - HyQ walking on flat terrain: The upper plots show the tracking of the base orientation (roll, pitch and yaw) and of the base height. Note that the height does not attain its reference, because it is implemented via the postural task that is in the null-space of the orientation task. In the lower plots instead, the tracking of the left front and right hind foot in the X, Y and Z coordinates is shown. After approximately 15 s, the gait is switched from a crawl to a trot.

leading to these results has received funding from the INAIL - Teleoperation Project.

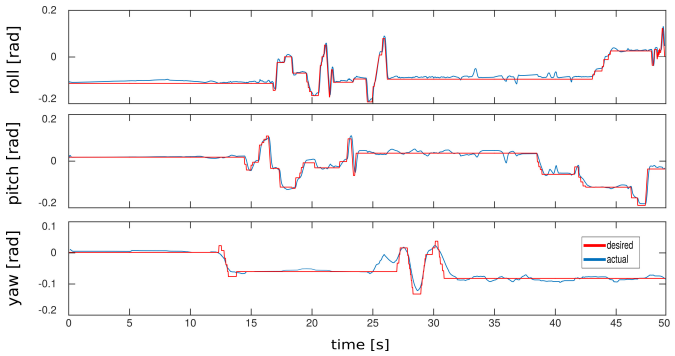


Fig. 10: Real hardware - HyQ changing the base orientation in roll, pitch and yaw.

APPENDIX A
RELATING INVERSE DYNAMICS TO IMPEDANCE CONTROL

This appendix will show how it is possible to relate inverse dynamics to Cartesian impedance control by opportunely selecting tasks gains.

Let us first consider a *naive* example with a single postural task:

$$\ddot{\mathbf{q}}^* = \underset{\ddot{\mathbf{q}}}{\operatorname{argmin}} \; \|\ddot{\mathbf{q}} - \ddot{\mathbf{q}}_r\|, \tag{33}$$

with solution:

$$\ddot{\mathbf{q}}^* = \ddot{\mathbf{q}}_r = \mathbf{K}\left(\mathbf{q}_d - \mathbf{q}\right). \tag{34}$$

To neglect the inertia matrix, we just need to choose a particular gain for the $\ddot{\mathbf{q}}_\mathbf{r}$:

$$\ddot{\mathbf{q}}_\mathbf{r} = \mathbf{M}^{-1}\mathbf{K}'\left(\mathbf{q}_d - \mathbf{q}\right) = \mathbf{M}^{-1}\boldsymbol{\tau}_r. \tag{35}$$

If we plug (35) in (3), neglecting non-linear terms, we obtain:

$$\boldsymbol{\tau} = \mathbf{M}\ddot{\mathbf{q}}_\mathbf{r} = \boldsymbol{\tau}_r, \tag{36}$$

obtaining the classic joint impedance control where the inertia matrix does not appear.

We now consider the optimization in (13) for the unconstrained case of a single, full rank, Cartesian task:

$$\ddot{\mathbf{q}}^* = \underset{\ddot{\mathbf{q}}}{\operatorname{argmin}} \ \|\mathbf{J}\ddot{\mathbf{q}} - \ddot{\mathbf{x}}_r\|, \tag{37}$$

with $\mathbf{J} \in \mathbb{R}^{6\times 6}$. If we compute the Lagrangian from (37) we obtain:

$$\mathcal{L} = \frac{1}{2}\ddot{\mathbf{q}}^T\mathbf{J}^T\mathbf{J}\ddot{\mathbf{q}} - \left(\mathbf{J}\ddot{\mathbf{q}}\right)^T\ddot{\mathbf{x}}_r + \ddot{\mathbf{x}}_r^T\ddot{\mathbf{x}}_r. \tag{38}$$

To solve (37) we derive the Lagrangian:

$$\frac{\partial \mathcal{L}}{\partial \ddot{\mathbf{q}}} = \mathbf{J}^T\mathbf{J}\ddot{\mathbf{q}} - \mathbf{J}^T\ddot{\mathbf{x}}_r = \mathbf{0}. \tag{39}$$

With the hypothesis of $\mathbf{J}$ full rank, the matrix $\mathbf{J}^T\mathbf{J}$, is invertible and the solution of (37) is given by:

$$\ddot{\mathbf{q}} = \left(\mathbf{J}^T\mathbf{J}\right)^{-1}\mathbf{J}^T\ddot{\mathbf{x}}_r = \mathbf{J}^{-1}\ddot{\mathbf{x}}_r = \mathbf{J}^{-1}\left(\mathbf{K}\left(\mathbf{x}_d - \mathbf{x}\right) - \dot{\mathbf{J}}\dot{\mathbf{q}}\right). \tag{40}$$

In this case, we can neglect the inertia imposing:

$$\begin{aligned}\ddot{\mathbf{x}}_r = \mathbf{K}\left(\mathbf{x}_d - \mathbf{x}\right) - \dot{\mathbf{J}}\dot{\mathbf{q}} &= \mathbf{J}\mathbf{M}^{-1}\mathbf{J}^T\mathbf{K}'\left(\mathbf{x}_d - \mathbf{x}\right) - \dot{\mathbf{J}}\dot{\mathbf{q}} \\ &= \mathbf{\Lambda}^{-1}\mathbf{F}_r - \dot{\mathbf{J}}\dot{\mathbf{q}},\end{aligned} \tag{41}$$

where $\mathbf{\Lambda} = \left(\mathbf{J}\mathbf{M}^{-1}\mathbf{J}^T\right)^{-1}$ is the Cartesian inertia matrix of the task. (41) plugged in (3) returns:

$$\boldsymbol{\tau} = \mathbf{J}^T\mathbf{F}_r - \mathbf{M}\mathbf{J}^{-1}\dot{\mathbf{J}}\dot{\mathbf{q}}. \tag{42}$$

Notice that (42) is equivalent to a Cartesian impedance controller [35].

Finally, let us consider the final level of a hyerarchical controller. Again we consider the optimization in (13):

$$\begin{aligned}\ddot{\mathbf{q}}^* = \underset{\ddot{\mathbf{q}}}{\operatorname{argmin}} \ &\|\ddot{\mathbf{q}} - \ddot{\mathbf{q}}_r\|_{\mathbf{W}} \\ \text{subject to} \\ &\mathbf{J}\ddot{\mathbf{q}} = \ddot{\mathbf{x}}^*,\end{aligned} \tag{43}$$

with $\mathbf{J} \in \mathbb{R}^{m\times n}$ containing all the Jacobians from the previous levels and $\ddot{\mathbf{x}}^* \in \mathbb{R}^m$ all the optimal accelerations obtained at the previous levels. The Lagrangian is given by:

$$\mathcal{L} = \frac{1}{2}\ddot{\mathbf{q}}^T\mathbf{W}\ddot{\mathbf{q}} - \ddot{\mathbf{q}}^T\mathbf{W}\ddot{\mathbf{q}}_r + \ddot{\mathbf{q}}_r^T\mathbf{W}\ddot{\mathbf{q}}_r + \boldsymbol{\lambda}^T\left(\mathbf{J}\ddot{\mathbf{q}} - \ddot{\mathbf{x}}^*\right), \tag{44}$$

which leads to the following optimal conditions:

$$\begin{aligned}\tfrac{\partial \mathcal{L}}{\partial \ddot{\mathbf{q}}} &= \mathbf{W}\ddot{\mathbf{q}} - \mathbf{W}\ddot{\mathbf{q}}_r + \mathbf{J}^T\boldsymbol{\lambda} = \mathbf{0}, \\ \tfrac{\partial \mathcal{L}}{\partial \boldsymbol{\lambda}} &= \mathbf{J}\ddot{\mathbf{q}} - \ddot{\mathbf{x}}^* = \mathbf{0}.\end{aligned} \tag{45}$$

The final optimal accelerations are given by:

$$\begin{aligned}\ddot{\mathbf{q}}* = \mathbf{W}^{-1}\mathbf{J}^T\left(\mathbf{J}\mathbf{W}^{-1}\mathbf{J}^T\right)^{-1}\ddot{\mathbf{x}}_r^* \\ + \left(\mathbf{I} - \mathbf{W}^{-1}\mathbf{J}^T\left(\mathbf{J}\mathbf{W}^{-1}\mathbf{J}^T\right)\mathbf{J}\right)^{-1}\ddot{\mathbf{q}}_r,\end{aligned} \tag{46}$$

it is well known that is possible to achieve the dynamically consistent inverted Jacobian [12] posing $\mathbf{W} = \mathbf{M}$:

$$\begin{aligned}\ddot{\mathbf{q}}* &= \bar{\mathbf{J}}^\dagger\ddot{\mathbf{x}}_r^* + \left(\mathbf{I} - \bar{\mathbf{J}}^\dagger\mathbf{J}\right)\ddot{\mathbf{q}}_r \\ \bar{\mathbf{J}}^\dagger &= \mathbf{M}^{-1}\mathbf{J}^T\left(\mathbf{J}\mathbf{M}^{-1}\mathbf{J}^T\right)^{-1}.\end{aligned} \tag{47}$$

We now plug (35) and (41) into (47):

$$\ddot{\mathbf{q}}* = \mathbf{M}^{-1}\mathbf{J}^T\mathbf{F}_r - \mathbf{M}^{-1}\mathbf{J}^T\mathbf{\Lambda}\dot{\mathbf{J}}\dot{\mathbf{q}} + \left(\mathbf{I} - \bar{\mathbf{J}}^\dagger\mathbf{J}\right)\mathbf{M}^{-1}\boldsymbol{\tau}_r \tag{48}$$

which plugged into (3) returns:

$$\boldsymbol{\tau} = \mathbf{J}^T\mathbf{F}_r - \mathbf{J}^T\mathbf{\Lambda}\dot{\mathbf{J}}\dot{\mathbf{q}} + \left(\mathbf{I} - \mathbf{J}^T\left(\mathbf{J}\mathbf{M}^{-1}\mathbf{J}^T\right)^{-1}\mathbf{J}\mathbf{M}^{-1}\right)\boldsymbol{\tau}_r \tag{49}$$

which again corresponds to a Cartesian impedance controller with dynamically consistent null-space projection [35].

REFERENCES

[1] M. Kalakrishnan, J. Buchli, P. Pastor, M. Mistry, and S. Schaal, "Learning, planning, and control for quadruped locomotion over challenging terrain," *The International Journal of Robotics Research*, vol. 30, no. 2, pp. 236–258, nov 2010.

[2] A. Winkler, C. Mastalli, M. Focchi, D. G. Caldwell, and I. Havoutis, "Planning and Execution of Dynamic Whole-Body Locomotion for a Hydraulic Quadruped on Challenging Terrain," *IEEE International Conference on Robotics and Automation (ICRA)*, pp. 5148–5154, Seattle, 2015.

[3] P. Fankhauser, C. Dario Bellicoso, C. Gehring, R. Dubé, A. Gawel, and M. Hutter, "Free Gait - An architecture for the versatile control of legged robots," *IEEE-RAS International Conference on Humanoid Robots*, pp. 1052–1058, Cancun, 2016.

[4] J.-S. Pang and J. Trinkle, "Stability characterizations of rigid body contact problems with coulomb friction," *ZAMM-Journal of Applied Mathematics and Mechanics/Zeitschrift für Angewandte Mathematik und Mechanik*, vol. 80, no. 10, pp. 643–663, 2000.

[5] M. Bloesch, M. Hutter, M. A. Hoepflinger, S. Leutenegger, C. Gehring, C. D. Remy, and R. Siegwart, "State estimation for legged robots-consistent fusion of leg kinematics and imu," *Robotics*, vol. 17, pp. 17–24, Cambridge, 2013.

[6] S. Nobili, M. Camurri, V. Barasuol, M. Focchi, D. Caldwell, C. Semini, and M. Fallon, "Heterogeneous Sensor Fusion for Accurate State Estimation of Dynamic Legged Robots," *Robotics: Science and Systems XIII*, Cambridge, 2017.

[7] M. Focchi, R. Orsolino, M. Camurri, V. Barasuol, C. Mastalli, D. G. Caldwell, and C. Semini, *Heuristic Planning for Rough Terrain Locomotion in Presence of External Disturbances and Variable Perception Quality*. Springer International Publishing, 2020, pp. 165–209.

[8] D. E. Whitney, "Resolved motion rate control of manipulators and human prostheses," *IEEE Transactions on man-machine systems*, vol. 10, no. 2, pp. 47–53, 1969.

[9] A. Liegeois *et al.*, "Automatic supervisory control of the configuration and behavior of multibody mechanisms," *IEEE transactions on systems, man, and cybernetics*, vol. 7, no. 12, pp. 868–871, 1977.

[10] Y. Nakamura, H. Hanafusa, and T. Yoshikawa, "Task-priority based redundancy control of robot manipulators," *The International Journal of Robotics Research*, vol. 6, no. 2, pp. 3–15, 1987.

[11] B. Siciliano and J. J. E. Slotine, "A general framework for managing multiple tasks in highly redundant robotic systems," in *ICAR, Fifth International Conference on Advanced Robotics*. IEEE, Pisa, 1991, pp. 1211–1216.

[12] O. Khatib, "A unified approach for motion and force control of robot manipulators: The operational space formulation," *Journal of Robotics and Automation*, vol. 3, no. 1, pp. 43–53, 1987.

[13] L. Sentis and O. Khatib, "Task-oriented control of humanoid robots through prioritization," *International Journal of Humanoid Robotics*, pp. 1–16, 2004.

[14] W. Dece, R. Smits, H. Bruyninckx, and J. De Schutter, "Extending iTaSC to support inequality constraints and non-instantaneous task specification," *Proceedings - IEEE International Conference on Robotics and Automation*, pp. 964–971, Kobe, 2009.

[15] M. De Lasa and A. Hertzmann, "Prioritized optimization for task-space control," in *IEEE/RSJ International Conference on Intelligent Robots and Systems*, vol. 3. Ieee, St. Louis, 2009, pp. 5755–5762.

[16] L. Righetti, J. Buchli, M. Mistry, and S. Schaal, "Control of legged robots with optimal distribution of contact forces," *2011 11th IEEE-RAS International Conference on Humanoid Robots*, pp. 318–324, Bled, 2011. [Online]. Available: http://ieeexplore.ieee.org/lpdocs/epic03/wrapper.htm?arnumber=6100832

[17] L. Saab, O. E. Ramos, N. Mansard, P. Soueres, and J.-y. Fourquet, "Dynamic Whole-Body Motion Generation under Rigid Contacts and other Unilateral Constraints," *IEEE Transactions on Robotics*, vol. 29, no. 2, pp. 346–362, 2013.

[18] P. M. Wensing and D. E. Orin, "Generation of dynamic humanoid behaviors through task-space control with conic optimization," in *2013 IEEE International Conference on Robotics and Automation*. IEEE, may Karlsruhe, 2013, pp. 3103–3109.

[19] A. Del Prete, F. Nori, G. Metta, and L. Natale, "Prioritized motion-force control of constrained fully-actuated robots: "task Space Inverse Dynamics"," *Robotics and Autonomous Systems*, vol. 63, no. P1, pp. 150–157, 2015. [Online]. Available: http://dx.doi.org/10.1016/j.robot.2014.08.016

[20] 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.

[21] D. Bellicoso, C. Gehring, J. Hwangbo, P. Fankhauser, and M. Hutter, "Perception-Less Terrain Adaptation through Whole Body Control and Hierarchical Optimization," in *IEEE-RAS International Conference on Humanoid Robots*, Cancun, 2016.

[22] T. Koolen, S. Bertrand, G. Thomas, T. de Boer, T. Wu, J. Smith, J. Englsberger, and J. Pratt, "Design of a Momentum-Based Control Framework and Application to the Humanoid Robot Atlas," *International Journal of Humanoid Robotics*, vol. 13, no. 01, p. 1650007, 2016.

[23] J. Salini, V. Padois, and P. Bidaud, "Synthesis of complex humanoid whole-body behavior: A focus on sequencing and tasks transitions," in *2011 IEEE International Conference on Robotics and Automation*. IEEE, Shanghai, 2011, pp. 1283–1290.

[24] E. M. Hoffman, A. Laurenzi, L. Muratore, N. G. Tsagarakis, and D. G. Caldwell, "Multi-Priority Cartesian Impedance Control Based on Quadratic Programming Optimization," *Proceedings - IEEE International Conference on Robotics and Automation*, pp. 309–315, Brisbane, 2018.

[25] A. Laurenzi, E. Mingo Hoffman, M. Parigi Polverini, and N. G. Tsagarakis, "Balancing Control Through Post-Optimization of Contact Forces," *IEEE-RAS International Conference on Humanoid Robots*, vol. 2018-Novem, pp. 320–326, 2019.

[26] A. Sproewitz, L. Kuechler, A. Tuleu, M. Ajallooeian, M. D'Haene, R. Mockel, and A. Ijspeert, "Oncilla robot: a light-weight bio-inspired quadruped robot for fast locomotion in rough terrain," *Symposium on adaptive motion of animals and machines (AMAM)*, Hyogo, 2011.

[27] V. Barasuol, J. Buchli, C. Semini, M. Frigerio, E. R. De Pieri, and D. G. Caldwell, "A reactive controller framework for quadrupedal locomotion on challenging terrain," *Proceedings - IEEE International Conference on Robotics and Automation*, pp. 2554–2561, Karlsruhe. 2013.

[28] J. S. Yuan, "Closed-loop manipulator control using quaternion feedback," *IEEE Journal on Robotics and Automation*, vol. 4, no. 4, pp. 434–440, Aug 1988.

[29] M. Focchi, A. del Prete, I. Havoutis, R. Featherstone, D. G. Caldwell, and C. Semini, "High-slope terrain locomotion for torque-controlled quadruped robots," *Autonomous Robots*, vol. 41, no. 1, pp. 259–272, Jan 2017.

[30] B. Siciliano, L. Sciavicco, L. Villani, and G. Oriolo, *Robotics: Modelling, Planning and Control*, ser. Advanced Textbooks in Control and Signal Processing. Springer, 2009.

[31] M. Focchi, G. A. Medrano-Cerda, T. Boaventura, M. Frigerio, C. Semini, J. Buchli, and D. G. Caldwell, "Robot impedance control and passivity analysis with inner torque and velocity feedback loops," *Control Theory and Technology*, pp. 1–16, 2016.

[32] H. Ferreau, C. Kirches, A. Potschka, H. Bock, and M. Diehl, "qpOASES: A parametric active-set algorithm for quadratic programming," *Mathematical Programming Computation*, vol. 6, no. 4, pp. 327–363, 2014.

[33] E. Mingo Hoffman, A. Rocchi, A. Laurenzi, and others., "Robot Control for Dummies: Insights and Examples using OpenSoT," in *IEEE-RAS International Conference on Humanoid Robots*, 2017, pp. 736–741.

[34] S. Chitta, E. Marder-Eppstein, W. Meeussen, V. Pradeep, A. Rodríguez Tsouroukdissian, J. Bohren, D. Coleman, B. Magyar, G. Raiola, M. Lüdtke, and E. Fernández Perdomo, "ros_control: A generic and simple control framework for ros," *The Journal of Open Source Software*, 2017. [Online]. Available: http://www.theoj.org/joss-papers/joss.00456/10.21105.joss.00456.pdf

[35] C. Ott, *Cartesian impedance control of redundant and flexible-joint robots*. Springer, 2008.