Model Predictive Control with Environment Adaptation for Legged Locomotion

NIRAJ RATHOD[1,2], ANGELO BRATTA[2,3], MICHELE FOCCHI [2], MARIO ZANON[1], OCTAVIO VILLARREAL[2], CLAUDIO SEMINI[2] AND ALBERTO BEMPORAD[1]

[1]The authors are with the IMT School for Advanced Studies Lucca, Lucca, Italy (e-mail: name.surname@imtlucca.it)

[2]The authors are with the Dynamic Legged Systems Lab, Istituto Italiano di Tecnologia (IIT), Genova, Italy (e-mail: name.surname@iit.it)

[3]Dipartimento di Informatica, Bioingegneria, Robotica e Ingegneria dei Sistemi (DIBRIS), Università di Genova, Genova, Italy

Corresponding author: Niraj Rathod (e-mail: niraj.rathod@imtlucca.it).

ABSTRACT Re-planning in legged locomotion is crucial to track the desired user velocity while adapting to the terrain and rejecting external disturbances. In this work, we propose and test in experiments a real-time Nonlinear Model Predictive Control (NMPC) tailored to a legged robot for achieving dynamic locomotion on a variety of terrains. We introduce a mobility-based criterion to define an NMPC cost that enhances the locomotion of quadruped robots while maximizing leg mobility and improves adaptation to the terrain features. Our NMPC is based on the real-time iteration scheme that allows us to re-plan online at $25\,\mathrm{Hz}$ with a prediction horizon of 2 seconds. We use the single rigid body dynamic model defined in the center of mass frame in order to increase the computational efficiency. In simulations, the NMPC is tested to traverse a set of pallets of different sizes, to walk into a V-shaped chimney, and to locomote over rough terrain. In real experiments, we demonstrate the effectiveness of our NMPC with the mobility feature that allowed IIT's $87\,\mathrm{kg}$ quadruped robot HyQ to achieve an omni-directional walk on flat terrain, to traverse a static pallet, and to adapt to a repositioned pallet during a walk.

INDEX TERMS Legged locomotion, Mobility, Nonlinear Model Predictive Control, Online re-planning

NOMENCLATURE

The list of most commonly used symbols used in this article.

Acronyms:

CoM	Center of Mass.
GRFs	Ground Reaction Forces.
NMPC	Nonlinear Model Predictive Control.
RTI	Real-time Iteration.
SRBD	Single Rigid Body Dynamics.
VFA	Vision-based Foothold Adaptation.
WBC	Whole-Body Control.
ZMP	Zero Moment Point.

Notation:

n_x	Number of states.
n_u	Number of control inputs.
n_a	Number of model parameters.
T	Prediction horizon.
N	Number of control intervals.
μ	Friction coefficient.
$\mathbf{x}^{\mathrm{p}} \in \mathbb{R}^{n_x \times (N+1)}$	Predicted states by NMPC.
$\mathbf{u}^{\mathrm{p}} \in \mathbb{R}^{n_x \times N}$	Optimal control inputs from NMPC.

$\mathbf{x}^{\mathrm{ref}} \in \mathbb{R}^{n_x \times (N+1)}$	Reference states.
$\mathbf{u}^{\mathrm{ref}} \in \mathbb{R}^{n_x \times N}$	Reference control inputs.
$\mathbf{p}_{\mathrm{c}} \in \mathbb{R}^3$	Robot's CoM position.
$\mathbf{v}_{\mathrm{c}} \in \mathbb{R}^3$	Robot's CoM velocity.
$\boldsymbol{\Phi} \in \mathbb{R}^3$	Orientation of robot's base.
$\boldsymbol{\omega} \in \mathbb{R}^3$	Angular velocity of robot's base.
$\boldsymbol{f}_i \in \mathbb{R}^3$	GRF at i^{th} foot.
$\mathbf{p}_{\mathrm{f},i} \in \mathbb{R}^3$	Foot position of ith foot.
$\boldsymbol{\delta} \in \mathbb{R}^4$	Contact status.
$_{\mathcal{C}}\mathbf{p}_{\mathrm{hf}} \in \mathbb{R}^{12}$	Hip-to-foot distance in CoM frame.

I. INTRODUCTION

The main advantage of legged robots with respect to their wheeled counterpart is their ability to traverse complex and unstructured environment such as forests, obstacles, and debris. However, the control of legged robots poses complex problems related to underactuation (the body is controlled only indirectly through the legs), and to the hybrid nature of the forces required to generate motion, since the robot needs to establish and interrupt contact between its feet and

the ground. The control design for legged robots was initially dealt with by using heuristic approaches which yielded successful results such as in the walking machines from Raibert [1], the virtual model control of Pratt et al. [2] and in the heuristic locomotion planning for quadrupedal robots by Focchi et al. [3]. However, heuristic approaches have several limitations, for example: 1) they cannot be easily generalized to all kinds of terrain and motions; 2) they cannot account for the future state of the robot hence, they have no possibility to guarantee physical feasibility of the planned trajectories. The challenge of avoiding these undesirable *myopic* behaviors in heuristic planning approaches has motivated the research towards new optimization-based predictive locomotion planning.

Formulating the locomotion planning as an optimization problem allows one to represent high-level locomotion tasks as cost functions and system dynamics using constraints. Besides robot dynamics, the locomotion tasks should also respect the contact dynamics such as unilateral force and friction cone constraints, that are critical to stabilize the locomotion. The use of optimization techniques to design Whole-Body Control (WBC) has enabled legged robots to traverse soft terrains [5] and to be versatile in terms of type of gait and motions that a legged robot can achieve [6]. The aforementioned examples are based on the solution of a Quadratic Program that only considers the instantaneous [7] effects of the joint torques on the robot's base. Further, similar to heuristic approaches mentioned earlier, these approaches do not consider the information about the future states of the robot and hence cannot assure recursive feasiblity.

In order to address this issue, several approaches make use of Trajectory Optimization (TO)-based locomotion planning considering the full dynamics of the robot [8, 9]. However, these approaches usually suffer from high computational time hence they are often restricted to *offline* (open-loop) use. In general, offline planners [10, 11] neither adapt to quick terrain changes nor cope with state drifts and uncertainties. To address this issue the concept of *online* re-planning can be used. Online re-planning can intrinsically cope with the problem of error accumulation in planned motion that is common in real scenarios.

For online re-planning, MPC has gained broad interest in the robotics community for legged locomotion. Moreover, the intrinsic feedback mechanism offered by MPC can compensate for modeling errors and disturbances acting on the system provided that the MPC is executed at a sufficiently high rate in closed-loop. A careful choice of the dynamic model inside MPC formulation is typically required to achieve a desired re-planning frequency in closed-loop, given the limited computational resources available for on-line computations. For example, using a full dynamics model of legged robots inside MPC [12, 13] with long prediction horizon may result in an optimization problem which requires excessive computations for real-time deployment at high sampling rates. Using approximate models is a way to reduce the complexity of the optimization, trading the
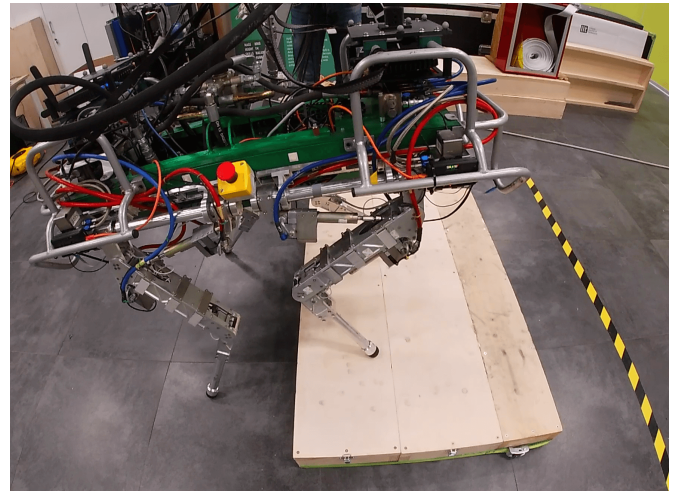

FIGURE 1. IIT's quadruped robot HyQ traversing a pallet with the mobility enhanced real-time NMPC.

accuracy with computational efficiency. Following this line, [14] used a Centroidal Dynamics (CD) plus a full-kinematic model to enforce the kinematic limits in TO to plan complex behaviors on the humanoid robot Atlas. The CD model considers contact forces as input and links the linear and angular momentum of the robot to the external wrench [15].

A simplified version of the CD model is the Single Rigid Body Dynamics (SRBD) model where the inertia of the legs is neglected (assumption of massless legs) and the robot's body and legs are lumped into a single rigid body. This model is well suited for quadrupeds, since they usually concentrate their mass and inertia in the robot base, unlike humanoids. The SRBD model was used for TO [16] and MPC [17] to jointly optimize for footholds, Center of Mass (CoM) trajectories and contact forces. By further linearizing the angular part of the dynamics of the SRBD, [18] was able to achieve a variety of quadrupedal gaits in experiments but their approach was not suitable for motions that involve large variations from the horizontal orientation. The simplest among all the approximate models mentioned earlier is the Linear Inverted Pendulum Model (LIPM) and it has been used inside MPC for quadruped [19] and biped [20] locomotion. However, there are two main limitations in LIPM, namely it neglects angular dynamics and assumes constant robot height. Additionally, it does not account for friction cones, so that the contact stability on non-flat terrain can not be guaranteed.

While models play an important role in obtaining computationally light MPC formulations, the choice of solution method is also paramount to achieve fast online re-planning with MPC. A Differential Dynamic Programming (DDP) based approach demonstrated the real-time performance with whole-body MPC [13] on HRP-2 humanoid. Recently, [21] proposed a DDP-based MPC using a kinodynamic model which re-plans at a frequency of $15\,\mathrm{Hz}$ with a prediction horizon of $1\,\mathrm{s}$ on a quadruped. The main drawbacks of

DDP based approaches is the difficulty in implementing hard-inequality and switching constraints. Hard-inequality constraints need to be implemented as penalties (e.g., with relaxed-barriers) [22] while the switching constraints are formulated using Augmented Lagrangian methods [23, 24]. Though not obvious at first sight, these methods are essentially equivalent to direct optimal control based on multiple-shooting [25] in combination with some form of Nonlinear Programming (NLP) solvers using barrier functions in the real-time iteration scheme [26]. One such framework is provided by `acados` [27].

In addition to DDP-based MPC, there also exist a few implementations of NLP-based MPC for legged robots. One such NMPC implementation with CoM dynamics plus full kinematic model was demonstrated in [28] using a Sequential Linear Quadratic (SLQ) algorithm for a trotting gait on flat terrain. Neunert et al. [12] achieved a fast re-planning frequency of 80-170 Hz for a small prediction horizon of 0.5 s (125 nodes) with their NMPC using the full dynamics of the robot, and optimizing foot locations, swing timing, and locomotion sequences along with full body dynamics. However, in the real experiments they have only demonstrated slow trotting on flat terrain. Moreover, since their approach does not consider the map of the terrain, it has limited application on uneven terrain conditions. An interesting observation is that they did not see a noticeable degradation in the closed-loop performance of the NMPC when the re-planning frequency was dropped until 30 Hz, demonstrating that the predictive nature of the MPC empowers the robot to tolerate much lower re-planning frequency. A similar observation was made in [29] with an MPC scheme which optimizes foot locations, but requires a heuristic conditioning of the cost function. In their experiments, the robot is stable if the re-planning occurs at 20 Hz, unstable for lower frequencies and the performance improvement is observed over 40 Hz.

The aforementioned approaches have been successful in controlling legged robots, but neglected an important aspect of these robots, which is usually referred to as *mobility*. In this paper, we define the mobility as the attitude of the robot leg to arbitrarily change its foot position [47]. We noticed that maximizing mobility improves terrain adaptation hence it is advantageous to account for it in the motion planning of legged robots. Furthermore, as discussed in Section IV-A, adding mobility in the NMPC cost eliminates the need to specify references for the roll, pitch and height of the robot.

To achieve kinematically suitable configurations for the legs, a common heuristic is to align the robot base with the terrain inclination (estimated in [3] via fitting an averaging plane through the stance feet). This approach aims to bring the legs as close as possible to the middle of their workspaces in order to avoid the violation of the kinematic limits. Optimization of mobility allows to achieve a similar behaviour in an automatic way. Fankhauser et al. in [30] maximized mobility by encoding it in a cost function that penalizes the distance with respect to a default foot position. Recently, Cebe et al. [31] implemented TO using an SRBD model

and also incorporating the feet positions in the optimization. They re-plan only at the feet touchdowns due to the high computation demand of their TO algorithm and showed experimental results on uneven terrain. Since their planner does not plan during the swing phase of the legs, they do not run their planner in an MPC fashion. Apart from the previously mentioned contributions, to the best of our knowledge no prior work has addressed the mobility with MPC in legged locomotion.

A. PROPOSED APPROACH AND CONTRIBUTION
In this work, we demonstrate in experiments with our 87 kg Hydraulically actuated Quadruped (HyQ) robot [32] that a suitably formulated NMPC can tackle rough terrain locomotion, account for leg mobility, and provide the optimal base orientation, while being real-time feasible. Indeed, optimizing for leg mobility allows our NMPC to devise a robot base orientation and height that improves locomotion on rough terrain.

This is particularly useful to achieve *environment adaptation* on rough terrains. Another advantage is that minimal heuristics is required from the user i.e., no reference trajectory for the robot's height, and its base roll and pitch orientation is needed.

This work is a system integration on the same line of our previous work [33]. However, while in [33] only offline optimization was performed, here we achieve real-time feasible online replanning in an MPC fashion. To achieve this goal:

a) We consider a simplified SRBD model that describes the angular and translational dynamics of the robot base but neglects the dynamics of legs.

b) We employ the real-time iteration (RTI) scheme [26, 34, 35] that allows us to run our NMPC online with the prediction horizon of 2 s (50 nodes) as opposed to the 0.5 s (125 nodes) used by [12]. Differently from [31] (that re-plans at each foot touchdown event), we continuously re-plan at the rate of 25 Hz.

c) We run our NMPC on a single computer along with the rest of our locomotion framework[1] unlike in [12, 31] where they use dedicated computers for their TO and NMPC, respectively.

We show in simulation the robot traversing a set of pallets of different dimensions placed relatively at varying distances, walking into a V-shaped chimney and lastly over a randomly generated rough terrain. We present *Experimental* results that demonstrate the capability of our NMPC to generate an omnidirectional walk and to traverse a pallet for our quadruped robot HyQ (see Fig. 1). We tested the re-planning capability of our approach by pushing a pallet in front of the robot while walking, such that the control algorithm has to re-plan online in order to adapt to a dynamically changing environment. To summarize, the contributions of our work (in order of importance) are as follows:

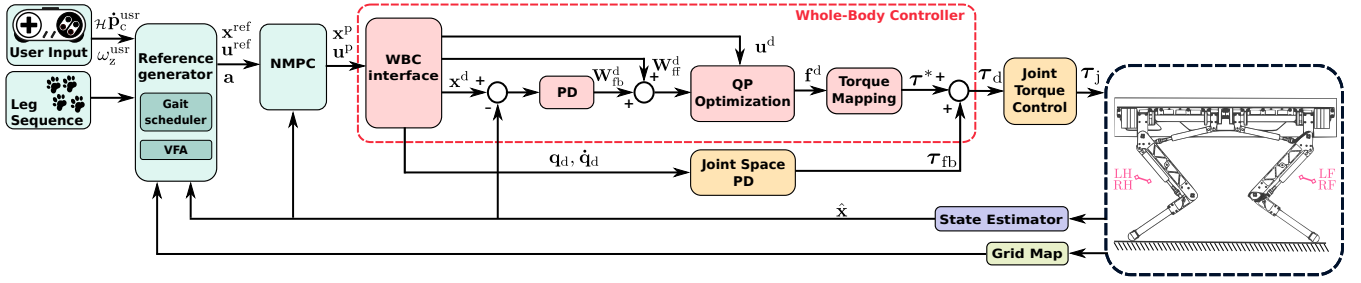[1]Except the perception related modules that run on a dedicated computer

FIGURE 2. Block diagram of the planning pipeline with the NMPC in our locomotion framework. The reference generator provides the references ($\mathbf{x}^{\text{ref}}$, $\mathbf{u}^{\text{ref}}$) to NMPC after receiving the user inputs. Then, the NMPC passes optimal state $\mathbf{x}^{\text{p}}$ and control $\mathbf{x}^{\text{p}}$ trajectories to the Whole-Body Controller. The torque $\boldsymbol{\tau}_{\text{d}}$ is given as reference to low level joint torque controller $\boldsymbol{\tau}_{\text{j}}$. The state estimator provides the state estimation $\hat{\mathbf{x}}$ to the required blocks. Finally, the heightmap is generated by Grid Map and given to the reference generator.

- Major *experimental* results are presented in Section VIII where we demonstrate the capability of our NMPC planner to generate an omni-directional walk and rough terrain locomotion for our quadruped robot HyQ, by exploiting an online evaluation of the map of the terrain and using on-board state estimation.

- Additional (minor) contributions are:

  a) We introduce a cost term which accounts for *mobility* by penalizing hip-to-foot positions that do not provide the highest mobility. To the best of our knowledge, this is the first time that mobility has been addressed with MPC.

  b) The generation of the reference trajectory for the NMPC takes into account premature and delayed touchdown of the feet as well as continuously adjusting the footholds according to the robot body motion and the terrain features.

  c) We use a parametric robot model that results in smaller NMPC formulation.

B. OUTLINE

The paper is organized as follows: Section II gives an overview of our planning pipeline whereas Section III describes the NMPC setup. The leg mobility and other features are explained in Section IV, whereas the generation of the references and the WBC are detailed in Sections V and VI, respectively. We then summarize the RTI scheme for our NMPC in Section VII. Further, Section VIII illustrates simulation and experimental results with the HyQ robot. Finally, we draw the conclusions in Section IX.

II. LOCOMOTION FRAMEWORK

Fig. 2 illustrates the planning pipeline of our locomotion framework. The *reference generator*, as discussed in Section V, takes the user input (longitudinal, lateral and angular velocity), schedule of the gait (e.g., a crawl) timing, the initial state of the robot, and a map of the terrain to generate reference trajectories for the state $\mathbf{x}^{\text{ref}}$ and control input $\mathbf{u}^{\text{ref}}$ required by the NMPC. The reference generator also provides a vector of parameters $\mathbf{a}$ to the NMPC, that includes foot locations and sequences of contact status. The NMPC

running at $25\,\text{Hz}$, delivers the optimal trajectories of the state $\mathbf{x}^{\text{p}}$ and control input $\mathbf{u}^{\text{p}}$, as detailed in Section III.

All the components of the Whole-Body controller (highlighted with dashed box in Fig. 2) are discussed in Section VI. The *WBC interface* interpolates the optimal state $\mathbf{x}^{\text{p}}$ at a rate of $250\,\text{Hz}$ to generate a desired signal $\mathbf{x}^{\text{d}}$ for a Cartesian virtual impedance controller [4]. The WBC interface also computes the feedforward wrench $\mathbf{W}_{\text{ff}}^{\text{d}}$ that is added to a feedback wrench $\mathbf{W}_{\text{fb}}^{\text{d}}$ that renders the Cartesian impedance. Moreover, the WBC interface provides the joint position $\mathbf{q}_{\text{d}}$ and velocities $\dot{\mathbf{q}}_{\text{d}}$ to a Joint Space PD controller running at $1\,\text{kHz}$. After acquiring the feedback and feed-forward wrenches, a Quadratic Programming (QP) optimization computes the vector of desired Ground Reaction Forces (GRFs) $\mathbf{f}^{\text{d}}$ accounting for the friction cone constraints and penalizing the the difference between $\mathbf{f}^{\text{d}}$ and $\mathbf{u}^{\text{p}}$ coming from the NMPC solution. Then, $\mathbf{f}^{\text{d}}$ is mapped to the torque vector $\boldsymbol{\tau}^*$ that is added to the Joint Space PD torques $\boldsymbol{\tau}_{\text{fb}}$ resulting into the total desired torque $\boldsymbol{\tau}_{\text{d}}$. Ultimately, $\boldsymbol{\tau}_{\text{d}}$ is passed to a low-level joint torque controller as reference [36].

An online state estimator [37] that runs at $500\,\text{Hz}$ provides the estimation of the robot state $\hat{\mathbf{x}}$ to all the components inside our locomotion framework that require it. A dedicated on-board computer takes inputs from an RGB-D camera (RealSense) mounted in front of the robot and generates a 2.5D heightmap at the rate of $30\,\text{Hz}$ using the *Grid Map* library from [38]. This heightmap is later sent to the reference generator.

III. NMPC

In our planning algorithm, we choose a real-time NMPC formulation because it has the ability to handle both the nonlinear system dynamics and the constraints, explicitly. NMPC is based on solving an Optimal Control Problem (OCP) given the current state $\hat{\mathbf{x}}_0$ of the system. Only the first element of the optimized input trajectory is applied to the system, then the state is measured and the OCP is solved again based on the new state measurement to close the loop.

We define the decision variables as the predicted state and control input with $\mathbf{x}^{\text{p}} = \{\mathbf{x}_0, \ldots, \mathbf{x}_N\}$ and $\mathbf{u}^{\text{p}} = \{\mathbf{u}_0, \ldots, \mathbf{u}_{N-1}\}$, respectively, such that an NLP formulation

can be stated as:

$$\min_{\mathbf{x}^{\mathrm{P}}, \mathbf{u}^{\mathrm{P}}} \quad \sum_{k=0}^{N-1} \ell\left(\mathbf{x}_k, \mathbf{u}_k, \mathbf{a}_k\right) + \ell_{\mathrm{T}}\left(\mathbf{x}_N\right) \tag{1a}$$

$$\text{s.t.} \quad \mathbf{x}_0 = \hat{\mathbf{x}}_0, \tag{1b}$$

$$\mathbf{x}_{k+1} = f\left(\mathbf{x}_k, \mathbf{u}_k, \mathbf{a}_k\right), \qquad k \in \mathbb{I}_0^{N-1}, \tag{1c}$$

$$h\left(\mathbf{x}_k, \mathbf{u}_k, \mathbf{a}_k\right) \leq 0, \qquad k \in \mathbb{I}_0^{N-1}, \tag{1d}$$

where, $\ell : \mathbb{R}^{n_x} \times \mathbb{R}^{n_u} \times \mathbb{R}^{n_a} \to \mathbb{R}$ is the stage cost function; $\ell_{\mathrm{T}} : \mathbb{R}^{n_x} \to \mathbb{R}$ is the terminal cost function. The initial condition (1b) is expressed by setting $\mathbf{x}_0$ equal to the state estimate $\hat{\mathbf{x}}_0$ received from the state estimator. The vector of model parameters $\mathbf{a}_k$ is not optimized but it is computed externally by the reference generator and provided to the optimization problem formulation. The nonlinear system dynamics are introduced by the equality constraints (1c). Finally, the path constraints are included with (1d) which, for example, can be bounds on the decision variables. The NLP (1) is defined for a *prediction horizon* T that is divided into N discrete time *control intervals* of lengths $T_{\mathrm{s}} = \frac{T}{N}$. Hereafter, we will refer to T_{s} as the *sampling time*.

A. COST

In our NMPC formulation we use a cost function of the form:

$$\ell\left(\mathbf{x}_k, \mathbf{u}_k, \mathbf{a}_k\right) = \ell_{\mathrm{t}} + \ell_{\mathrm{m}} + \ell_{\mathrm{r}}, \tag{2a}$$

$$\ell_{\mathrm{t}} = \| \mathbf{x}_k - \mathbf{x}_k^{\mathrm{ref}} \|_{\mathbf{Q}}^2 + \| \mathbf{u}_k - \mathbf{u}_k^{\mathrm{ref}} \|_{\mathbf{R}}^2, \tag{2b}$$

$$\ell_{\mathrm{m}} = \| {}_{\mathcal{C}}\mathbf{p}_{\mathrm{hf}_k} - {}_{\mathcal{C}}\mathbf{p}_{\mathrm{hf}_k}^{\mathrm{ref}} \|_{\mathbf{M}}^2, \tag{2c}$$

$$\ell_{\mathrm{r}} = \rho \| {}_{\mathcal{K}}\mathbf{u}_k \|_{\mathbf{P}}^2 \tag{2d}$$

- The tracking cost (2b) is associated to state and control input and the references trajectories $\mathbf{x}_k^{\mathrm{ref}}, \mathbf{u}_k^{\mathrm{ref}}$ are provided by the reference generator for each sampling instance k (refer Section V).
- The mobility cost (2c) is one of the contributions of this work that accounts for improving the leg mobility by penalizing the difference between the hip-to-foot distance ${}_{\mathcal{C}}\mathbf{p}_{\mathrm{hf}}$ and the reference value ${}_{\mathcal{C}}\mathbf{p}_{\mathrm{hf}}^{\mathrm{ref}}$ of maximum mobility. This cost allows the NMPC to optimize the robot base orientation (e.g. align it to the terrain shape) in order to increase the leg mobility which has as a desirable consequence to stay far from kinematic limits during locomotion. The derivation of ${}_{\mathcal{C}}\mathbf{p}_{\mathrm{hf}}^{\mathrm{ref}}$ is detailed separately in Section IV-A.
- In some locomotion scenarios [4], to cope with uncertainties in the contact normal estimation and increase robustness to external disturbances, it is desirable to have the GRFs $\mathbf{f}_i$ as close as possible to the center of the friction cone. This can be achieved with by penalizing X-Y components of $\mathbf{u}$ in a frame $\mathcal{K}$ (see Fig. 3) that is aligned to the normal of the contact and it is included in our cost function by a control input regularization term (2d), refer Section IV-C for the details.

The positive definite weight matrices $\mathbf{Q} \in \mathbb{S}_+^{n_x}, \mathbf{R} \in \mathbb{S}_+^{n_u}$, $\mathbf{M} \in \mathbb{S}_+^{12}, \mathbf{P} \in \mathbb{S}_+^{n_u}$ act as important tuning parameters in

the NMPC formulation. The regularization factor ρ decides the trade-off between force robustness cost (2d) and both the tracking (2b) and mobility (2c) cost. Finally, we define the terminal cost $\ell_{\mathrm{T}} = \| \mathbf{x}_N - \mathbf{x}_N^{\mathrm{ref}} \|_{\mathbf{Q}_N}$ and use the weight matrix $\mathbf{Q}_N = \mathbf{Q}$ for this cost.

B. ROBOT MODEL

The inertial frame $\mathcal{W}$ and the CoM frame $\mathcal{C}$ are shown in Fig. 3. The CoM frame is aligned with the base of the robot and its origin is located at the CoM. A variable with left subscript denotes its frame of reference. For example ${}_{\mathcal{C}}\boldsymbol{\omega}$ represents the angular velocity of the robot base expressed in the CoM frame $\mathcal{C}$. Note that, unless explicitly specified, all the relevant quantities in this paper are defined in the inertial frame $\mathcal{W}$. Throughout this paper we define $(a, \dots, b)$ as the column vector stacking any generic column vectors $a, \dots, b$.
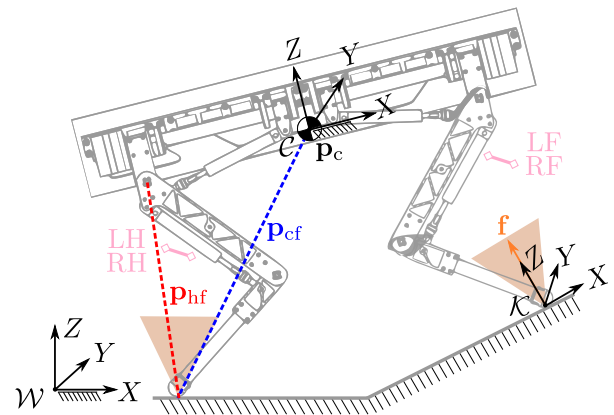

FIGURE 3. HyQ schematic showing the inertial frame ($\mathcal{W}$), the CoM frame ($\mathcal{C}$) attached to the CoM of the robot, and the contact frame ($\mathcal{K}$). The robot legs are shown in the *default* configuration.

We use a simplified reduced-order SRBD model [16] defined in a 6D space that describes the translational and angular dynamics of the robot while neglecting the dynamics of its swinging legs. This is a valid approximation for the HyQ robot because most of its mass is concentrated in the base, as mentioned in [32] (the mass of the base is $61\,\mathrm{kg}$ and the mass of each leg is $6.5\,\mathrm{kg}$). The robot is approximated as a rigid body with the inertia computed considering the robot in a default leg configuration as shown in Fig. 3. We choose to define the SRBD model in the CoM frame (specifically the angular dynamics) because this choice yields a constant inertia tensor. Thus, the angular dynamic equations are much simpler i.e., less non-linear because the inertia tensor is not time varying. In the SRBD model, GRFs are applied as inputs to control the position and orientation of the robot base. The SRBD model is:

$$m\dot{\mathbf{v}}_{\mathrm{c}} = m\mathbf{g} + \sum_{i=1}^{4} \delta_i \mathbf{f}_i \tag{3a}$$

$$_{\mathcal{C}}\mathbf{I}_{\mathrm{c}}\,{}_{\mathcal{C}}\dot{\boldsymbol{\omega}} + {}_{\mathcal{C}}\boldsymbol{\omega} \times {}_{\mathcal{C}}\mathbf{I}_{\mathrm{c}}\,{}_{\mathcal{C}}\boldsymbol{\omega} = \sum_{i=1}^{4} \delta_i\,{}_{\mathcal{C}}\mathbf{p}_{\mathrm{cf},i} \times {}_{\mathcal{C}}\mathbf{f}_i \tag{3b}$$

where m is the robot mass, $\dot{\mathbf{v}}_c \in \mathbb{R}^3$ is the CoM acceleration, $\mathbf{g}$ is the gravitational acceleration, $\mathbf{f}_i \in \mathbb{R}^3$ is the ground reaction force at foot i, $_c\mathbf{I}_c \in \mathbb{R}^{3 \times 3}$ is the inertia tensor computed at the CoM frame origin, $_c\dot{\boldsymbol{\omega}} \in \mathbb{R}^3$ is the angular acceleration of the robot's base, $\mathbf{p}_{cf,i} \in \mathbb{R}^3$ is the distance between the CoM position $\mathbf{p}_c \in \mathbb{R}^3$ and the position $\mathbf{p}_{f,i} \in \mathbb{R}^3$ of foot i. We introduce binary parameters $\delta_i = \{0, 1\}$ to define whether foot i is in contact with the ground and can therefore generate contact forces or not.

The robot dynamics governed by (3) can be expressed as the continuous-time state-space model:

$$\begin{bmatrix} \dot{\mathbf{p}}_c \\ \dot{\mathbf{v}}_c \\ \dot{\boldsymbol{\Phi}} \\ _c\dot{\boldsymbol{\omega}} \end{bmatrix} = \begin{bmatrix} \mathbf{v}_c \\ 1/m \sum_{i=1}^{4} \delta_i \mathbf{f}_i + \mathbf{g} \\ \mathbf{E}'^{-1}(\boldsymbol{\Phi})_c\boldsymbol{\omega} \\ -_c\mathbf{I}_c^{-1}(_c\boldsymbol{\omega} \times _c\mathbf{I}_c)_c\boldsymbol{\omega} + \sum_{i=1}^{4} \delta_i {}_c\mathbf{I}_c^{-1} {}_c\mathbf{p}_{cf,i} \times _c\mathbf{f}_i \end{bmatrix}$$
$$(4)$$

where $\mathbf{v}_c$ is the CoM velocity of the robot. The robot base orientation is represented by the sequence of Z-Y-X Euler angles [1] [39] $\boldsymbol{\Phi} = (\phi, \theta, \psi)$ i.e., roll (ϕ), pitch (θ) and yaw (ψ), respectively. The relation between the Euler Angles rates $\dot{\boldsymbol{\Phi}}$ and angular velocity $_c\boldsymbol{\omega}$ is well-known and discussed in Appendix A for the sake of completeness. We define the state and control vectors as $\mathbf{x} = (\mathbf{p}_c, \mathbf{v}_c, \boldsymbol{\Phi}, {}_c\boldsymbol{\omega})$, and $\mathbf{u} = (\mathbf{f}_1, \ldots, \mathbf{f}_4)$. Equation (4) can be concisely written as:

$$\dot{\mathbf{x}}(t) = g(\mathbf{x}(t), \mathbf{u}(t), \mathbf{a}(t)), \quad (5)$$

where $\mathbf{a} = (\mathbf{p}_f, \boldsymbol{\delta})$ is a vector of parameters that includes the feet positions $\mathbf{p}_f$ and the contact status $\boldsymbol{\delta} \in \mathbb{R}^4$.

The rigid-body dynamics (5) are discretized using numerical integration [40, 41, 42, 43] to obtain the discrete-time model:

$$\mathbf{x}_{k+1} = f(\mathbf{x}_k, \mathbf{u}_k, \mathbf{a}_k), \quad (6)$$

which defines equality constraints (1c) imposed at every stage k in MPC to ensure that the state trajectory satisfies the system dynamics for the given control inputs.

One specific feature of legged robots is the need to ensure that the values of the GRFs equal to zero for a swinging leg. This is typically done by introducing complementarity constraints [31, 44]. These constraints, however, pose several difficulties in the solution of the optimization problem, since the vast majority of the NLP algorithms cannot handle them and tailored solvers are required. Ultimately, this results in a significant increase in computation time. An alternative to complementarity constraints consists in providing the sequence of contact status $\boldsymbol{\delta}$ as input parameters in the state space model (5). In this manner, a contact mode δ_i is multiplied with the terms involving force $\mathbf{f}_i$ in (4) and the contribution of that force is nullified during the swing phase of the corresponding leg i. Hence, there is no more need to

---

[1]Note that Euler angles can suffer from singularities that occur in certain configurations [45]. Because in this work we do not consider motions that involve such configurations, using Euler angles does not pose any issue. A singularity-free implementation is out of the scope of this work and is left for future research.

include complementarity constraints separately in (1) which results in fewer constraints and, consequently, in a relatively smaller NMPC formulation.

C. FRICTION CONE AND UNILATERAL CONSTRAINTS

Friction cone constraints are encoded with their square pyramid approximation:

$$-\mu_i \mathbf{f}_{z,i} \leq \mathbf{f}_{x,i} \leq \mu_i \mathbf{f}_{z,i} \quad (7a)$$
$$-\mu_i \mathbf{f}_{z,i} \leq \mathbf{f}_{y,i} \leq \mu_i \mathbf{f}_{z,i} \quad (7b)$$
$$\underline{\mathbf{f}}_z \leq \mathbf{f}_{z,i} \leq \overline{\mathbf{f}}_z \quad (7c)$$

where, $\underline{\mathbf{f}}_z$ and $\overline{\mathbf{f}}_z$ are upper and lower bounds on GRFs Z component, respectively, and μ_i is the friction coefficient of the contact surface. Choosing $\underline{\mathbf{f}}_z$ greater than or equal to zero enforces unilateral constraints on the normal forces $\mathbf{f}_z$. The friction cone and unilateral constraint are represented by $h(\mathbf{x}_k, \mathbf{u}_k, \mathbf{a}_k) \leq 0$ in the NMPC formulation.

IV. LOCOMOTION-ENHANCING FEATURES

In this section we discuss the main distinctive features of our approach, which we found relevant to improve locomotion ability of our quadruped robot. These features are *mobility*, *force robustness* and *Zero Moment Point (ZMP) margin*.

A. MOBILITY AND MOBILITY FACTOR

Terrain adaptability is vital when it comes to locomotion of the legged robots. Adjusting the posture of the robot depending on the environment is important for safe locomotion. A way to enable our NMPC to choose robot orientation adaptively to any terrain is to employ the concept of *mobility* [46]. In order to rigorously discuss mobility in mathematical terms, we first define it in words as the attitude of a manipulator (leg) to arbitrarily change end-effector position/orientation [47].

In order to penalize low leg mobility in the cost function (2c) we need to compute the reference value of hip-to-foot distance $_c\mathbf{p}_{hf_k}^{\text{ref}}$. Our goal in this section is to define a convenient metric to represent mobility and compute $_c\mathbf{p}_{hf_k}^{\text{ref}}$ corresponding to the maximum value of such a metric. Among several ways to compute mobility [46], the velocity transformation ratio [48] allows one to evaluate mobility in a particular direction. However, the velocity transformation ratio cannot be used in our setting because it requires prior knowledge of the evolution of the relative foot position with respect to CoM. In our case it is not available in advance because it is an output the NMPC.

As an alternative approach, we consider the volume of the *manipulability ellipsoid* $\left(\mathbf{v}(\mathbf{J}\mathbf{J}^\top)^{-1}\mathbf{v} = 1\right)$ [47] as a metric to evaluate mobility. A change in the volume of the manipulability ellipsoid with different leg configurations is visualized in Fig. 4 (left). Inspecting Fig. 4 (top right), it can be seen that the maximum volume V is in the vicinity of the most extended leg configuration because the mobility becomes very big in the X and Y direction, even if it is still very limited in the Z direction. However, because it is desirable to achieve a
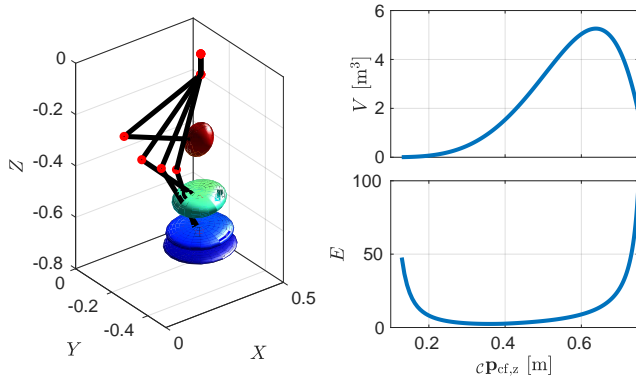
FIGURE 4. Manipulability ellipsoid changing with leg configuration (left) of the right front leg. Volume of the ellipsoid (top right) and Eccentricity of the ellipsoid (bottom right).
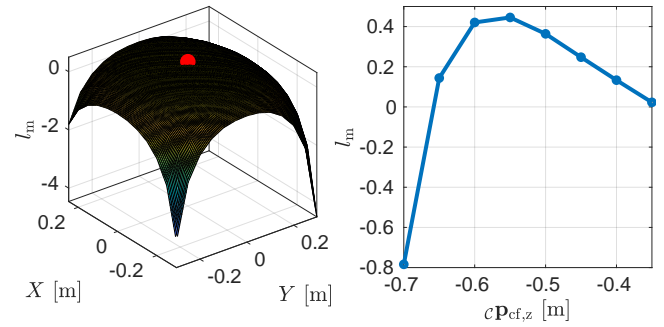

FIGURE 5. Slices of the mobility factor function for the RF leg: the left figure plots it against the X-Y components keeping Z constant. The red dot in the left plot represents the maxima. The right figure plots it against the Z component for a constant X-Y foot position.

good mobility in all the directions of the leg configuration, a better metric to do so is the one that accounts for the *isotropy* of the manipulability ellipsoid. A measure of the isotropy of an ellipsoid can be expressed as the inverse of its *eccentricity* E. Hence, a new manipulability index that we call *mobility factor* (8) can be defined in terms of both the eccentricity and the volume of manipulability ellipsoid. Again from Fig. 4, it can be visualized that to keep a good mobility (left plot) in all directions, the volume (top right plot) should be maximized while the eccentricity (bottom right plot) as small as possible. Defining a foot Jacobian $\mathbf{J}(\mathbf{q}) \in \mathbb{R}^{3\times 3}$ computed at a particular joint configuration $\mathbf{q}$, the volume V of a manipulability ellipsoid is evaluated as a product of the eigenvalues of $\left(\mathbf{J}(\mathbf{q})\mathbf{J}(\mathbf{q})^\top\right)^{-1}$ while the eccentricity is the ratio between its maximum and minimum eigenvalue [46]. First, the volume and eccentricity of manipulability ellipsoid are normalized by their ranges $\bar{V}$ and $\bar{E}$. Then we define the mobility factor as:

$$m_{\mathrm{f}} = \beta \frac{V}{\bar{V}} - \gamma \frac{E}{\bar{E}} \tag{8}$$

The minus sign in (8) represents conflicting contributions of the V and E in the definition of the mobility factor (i.e. the goal is to achieve high volume and low eccentricity). Parameters β and γ are introduced to find a best trade-off between volume and eccentricity while deciding a mobility factor.

The mobility factor is a convex nonlinear function m_{f} : $\mathbb{R}^3 \rightarrow \mathbb{R}$ that can be numerically evaluated inside the workspace of each leg. By selecting $\beta = 1$ and $\gamma = 4$, and after conducting a numerical analysis for all the feet positions in the workspace of a leg of the HyQ robot we found that hip-to-foot distance $_C\mathbf{p}_{\mathrm{hf}} = (0, 0, -0.55)\,\mathrm{m}$ maximizes m_{f}. In Fig. 5 (left) we show a slice of the scalar function m_{f} in the X-Y plane for $_C\mathbf{p}_{\mathrm{hf,z}} = -0.55\,\mathrm{m}$ obtained for the Right-Front (RF) leg. Instead, in Fig. 5 (right) we plot m_{f} against the change of foot position in the Z direction considering the hip under the foot ($X = 0$, $Y = 0$) which clearly highlights $_C\mathbf{p}_{\mathrm{hf,z}} = -0.55\,\mathrm{m}$ corresponding to the maximum value of

the mobility factor m_{f} (i.e., around 0.41). We use the output of this analysis as a reference for the hip-to-foot distance $_C\mathbf{p}_{\mathrm{hf}}^{\mathrm{ref}}$ in the mobility cost (2c) for all the legs.

In the mobility cost, we multiply δ_i to the term corresponding to the ith leg. Thus, the mobility cost solely accounts for stance legs because the robot can only use them to control its base orientation. Since, including the mobility cost enables NMPC to provide the optimal base orientation for a particular locomotion that retains mobility, there is no need to separately specify tracking cost for roll and pitch in the NMPC. This relieves a user from the burden of implementing a customized heuristic (e.g., to align the robot base to the terrain), as was necessary in, e.g., [3, 44, 49]. The relative tracking task for the CoM Z position is no longer required either, because maximizing the mobility in the Z direction automatically takes care of keeping an average distance of hips from the terrain to $_C\mathbf{p}_{\mathrm{hf,z}}$, consequently keeping the robot base at a certain height.

Moreover, the yaw motion results by penalizing the mobility cost along the X-Y directions. This has the effect of driving the hips of the robot base over the feet, naturally aligning the base to the feet, similar to what was done in [50]. However, a tracking cost on yaw was still necessary in the NMPC to track the heading velocity $\omega_{\mathrm{z}}^{\mathrm{usr}}$ commanded by the user and to avoid oscillations.

Remark: The concept of mobility is model independent hence, it can also be used with other models such as full body dynamics in the MPC setting.

B. ZMP MARGIN

In legged locomotion, the robot is often operated close to unstable configurations which require a controller to continuously compensate for model inaccuracies and external disturbances while maintaining locomotion stability. However, a configuration in which the ZMP [51] is close to the boundary of the support polygon [52] could cause instability even with small perturbations due to the loss of control authority.

In our case, the reference generator computes references for the GRFs by dividing the robot mass with the number of legs as explained in Section V. Penalizing GRFs Z com-

ponent heavily in the tracking cost (2b) ensures that they stay close to the reference, consequently maintaining a higher loading on the diagonally opposite leg to the swinging one, and therefore maintaining some margin for the locomotion stability. To evaluate the locomotion stability, we define the *ZMP margin* which is computed as the minimum of the distance of the ZMP from each support polygon edge, i.e.,

$$m_c = \min(\mathbf{d}) \tag{9}$$

where $\mathbf{d}$ is a vector of the distances of ZMP projection (on a horizontal plane) from the support polygon edges.

C. FORCE ROBUSTNESS

Similar to the considerations on mobility, in order to effectively compensate for disturbances acting on the system, robustness in the GRFs is required. The closer the GRF is to the friction cone boundary, the less lateral force is available to compensate for perturbations. An approach penalizing GRFs that are in the vicinity of the cone boundaries has been proposed in [4, 5] inside the WBC. These WBC based approaches instantaneously generate GRFs that are as close as possible to the normals of the cones while yielding the prescribed resultant wrench on the robot base. However, WBC does not account for the future state of the robot and hence, it leaves some room for the NMPC to compensate for the contact normal estimation error and recover from external disturbances. Introducing these margins on GRFs from the cone boundaries is especially important in some scenarios, such as the one reported in simulation in Section VIII-B2.

In this paper, we adopt a similar idea to [4, 5] and introduce the additional cost term (2d) in the NMPC, which penalizes the tangential components of GRFs in the contact frame $\mathcal{K}$ (see Fig. 3) to obtain the resultant GRFs as close as possible to the contact normals. The weight matrix $\mathbf{P}$ used in this cost is defined in Table 2 (Section VIII). Note that it is required to penalize the X-Y components higher than Z component of GRFs in the contact frame to achieve this behaviour.

V. REFERENCE GENERATOR

In our approach, the NMPC requires a reference trajectory of the state and control input along with the model parameters i.e., foot positions and contact status. For the very first run of the NMPC, this reference trajectory also serves as an initial guess. The references are generated for the length of control intervals N, since the reference generator is called before every iteration of the NMPC in order to obtain prompt adaptation to terrain changes and user set-point. Our reference generator is based on heuristics and it takes as inputs:

- the user commanded longitudinal and lateral CoM velocity $_\mathcal{H}\mathbf{v}_c^{usr} \in \mathbb{R}^2$ in the horizontal frame $\mathcal{H}$,[1]
- user commanded heading velocity $\omega_z^{usr} \in \mathbb{R}$,
- current pose of the robot $(\mathbf{p}_c, \mathbf{\Phi})$,
- current feet positions $\mathbf{p}_f \in \mathbb{R}^{12}$,

[1]The horizontal frame is placed like the CoM frame but with the Z-axis aligned with the gravity

- heightmap of the terrain

The reference generator outputs:

- the references for the NMPC cost: states $\mathbf{x}^{ref} \in \mathbb{R}^{n_x \times (N+1)}$, control $\mathbf{u}^{ref} \in \mathbb{R}^{n_u \times N}$,
- parameters $\mathbf{a}$ of the model: sequence of the contact status $\boldsymbol{\delta}$ ($\in \mathbb{R}^{4 \times N}$) and sequence of the foot locations $\mathbf{p}_f$ ($\in \mathbb{R}^{12 \times N}$),
- normals of the terrain at the foothold locations, which are provided as inputs to the NMPC for the cone constraints.

First we compute the X-Y components of the total velocity $\mathbf{v}_c^{ref} \in \mathbb{R}^3$, which depend on both $\mathbf{v}_c^{usr}$ and the X-Y components of the tangential velocity due to the heading velocity $\boldsymbol{\omega}^{usr} = (0, 0, \omega_z^{usr})$.

$$\mathbf{v}_{c,(x,y)}^{ref} = \mathbf{v}_c^{usr} + (\boldsymbol{\omega}^{usr} \times \mathbf{p}_c^{ref})_{(x,y)} \tag{10}$$

The X-Y CoM position $\mathbf{p}_{c,(x,y)}^{ref}$ is obtained by integrating the $\mathbf{v}_{c,(x,y)}^{ref}$ with the explicit Euler scheme. The references for CoM Z, roll and pitch are set to 0 because we do not track them in the NMPC cost (2b). Instead, the reference for the yaw ψ is obtained by integrating the user defined yaw rate $\dot{\psi}^{usr}$ with $\dot{\mathbf{\Phi}}^{usr} = \mathbf{E}^{-1}(\mathbf{\Phi}^{ref})\boldsymbol{\omega}^{usr}$ (see Appendix A for the transformation between angular velocity and Euler rates). The reference for angular velocity, instead, coincides with $\boldsymbol{\omega}^{usr}$.

The references for GRFs $\mathbf{u}^{ref}$ are calculated by simply dividing the total mass of the robot by the number of legs in stance. Dividing the forces equally onto the legs is correct only if the robot is static, but, in case of dynamic conditions, it is a better approximation than passing no references.

The sequence of contact status $\boldsymbol{\delta}$ and of footholds are computed by the *gait scheduler* and *robocentric stepping* strategy, respectively. It is important to mention that the reference generator does not compute the swing trajectories and they are obtained from the WBC interface discussed in Section VI-A.

1) Gait scheduler

The gait scheduler is logically decoupled from the reference trajectory generation and determines if a leg is either in swing or in stance (δ_i) at each time instance for the entire gait cycle as shown in Fig. 6 (left).

The leg duty factor D_i and offsets o_i can be used to encode different gaits such as crawl, trot and pace. The gait scheduler implements a time parametrization $s \in [0, 1]$ (*stride phase*) which is normalized about the cycle time duration T_c such that the leg duty factor D_i and offsets o_i are independent from the cycle time. Each trigger l_i^{tr} (red flag in Fig. 6 (left)) corresponds to a new lift-off event. We can express the value of δ for leg i as:

$$\delta_i = \begin{cases} 1, & s < o_i \lor s > ((o_i + (1 - D_i)) \bmod 1) \\ 0, & \text{otherwise} \end{cases} \tag{11}$$

Every time the reference generator is called, it extracts N points from the gait schedule starting from an index
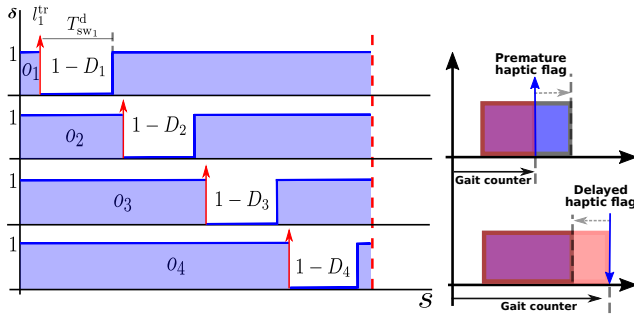
FIGURE 6. Gait schedule for a walk. Offsets $\mathbf{o} = [0.05, 0.3, 0.55, 0.8]$, duty-factors $\mathbf{D} = [0.85, 0.85, 0.85, 0.85]$. The red arrows represent the trigger l_i^{tr} for a swing leg i. Right part shows the fast-forwarding (top) or re-winding (bottom) of the gait counter to recover synchronization between actual (haptic) and planned touchdown.
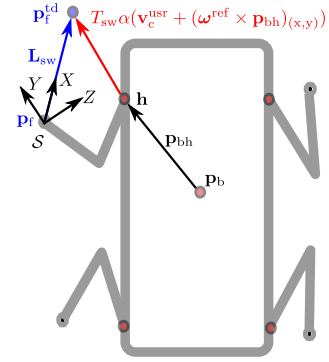

FIGURE 7. Representation of the robocentric stepping strategy and of the Swing Frame, located at the lift-off point. The red arrow shows the distance between the touchdown point $\mathbf{p}_{\text{f}}^{\text{td}}$ and the hip $\mathbf{h}$. The blue vector $\mathbf{L}_{\text{sw}}$ connects lift-off and touchdown point.

called *gait counter*. It keeps memory of the index of the gait schedule achieved by the previous call of the reference generator. The synchronization between the first point of a contact sequence δ_{i_k} computed by the reference generator and the actual contact state of the robot avoids the reference generator to compute a zero reference force while the leg is in stance and vice-versa. In the case of premature or delayed touchdown events, the synchronization is lost and the gait counter is shifted backwards or forward to re-conciliate the planned touchdown with the actual touchdown as shown in Fig. 6 (right). This is a crucial feature when dealing with rough terrains.

2) Robocentric stepping

The choice of the foothold is a key element in locomotion, since it deals with the kinematic limits of the robot. Inspired by [1], we use an approach that continuously computes footholds consistent with the current position of the robot. To compute a foothold for a swinging leg i, we consider its hip position $\mathbf{h}_i$ instead of using the foot position at the moment of lift-off. In this way a disturbance acting on the robot or a tracking error occurred during a swing can be recovered in the following swing. For a leg i, dropping the index to simplify the notation and defining the lift-off trigger as $l_k^{\text{tr}} = \delta_k \wedge \bar{\delta}_{k+1}$, the foot position is computed as:

$$\mathbf{p}_{\text{f}_{k+1}} = \begin{cases} \mathbf{p}_{\text{f}_k}^{\text{td}} & l_k^{\text{tr}} = 1 \\ \mathbf{p}_{\text{f}_k} & l_k^{\text{tr}} = 0 \end{cases} \quad (12)$$

Notice that at the lift-off condition $l^{\text{tr}} = 1$ at instance k, $\mathbf{p}_{\text{f}}$ is set equal to the touchdown point $\mathbf{p}_{\text{f}}^{\text{td}}$ and it is kept constant until the next lift-off event occurs. The X-Y component of the touchdown point is given by:

$$\mathbf{p}_{\text{f}_k,(\text{x,y})}^{\text{td}} = \mathbf{h}_k + \alpha T_{\text{sw}}^{\text{d}}(\mathbf{v}_{\text{c}}^{\text{usr}} + (\boldsymbol{\omega}^{\text{usr}} \times \mathbf{p}_{\text{bh}})_{(\text{x,y})}) \quad (13)$$

The second term in (13) represents the step length (red arrow in Fig. 7) which is computed with respect to the hip instead of the previous foot location. Parameter α is an empirically chosen scaling factor. Parameter T_{sw}^{d} is the default swing duration computed starting from user-defined offsets $\mathbf{o}$ and

duty-factors $\mathbf{D}$. The distance between hip and center of the base is denoted by $\mathbf{p}_{\text{bh}} \in \mathbb{R}^3$. A 2.5D heightmap of the terrain is evaluated in correspondence of the touchdown point $\mathbf{p}_{\text{f}_k,(\text{x,y})}^{\text{td}}$ to obtain $\mathbf{p}_{\text{f}_k,\text{z}}^{\text{td}}$ that does not penetrate the terrain. If $\mathbf{p}_{\text{f}}^{\text{td}}$ is located near to an edge or leads to collisions (e.g., of the foot or the shin) during the step cycle, this can be harmful for the robot's balance. To prevent this from happening, the robot acquires a local heightmap in the vicinity of the touchdown point $\mathbf{p}_{\text{f}}^{\text{td}}$ and adjusts the foot landing location using the Vision-based Foothold Adaptation (VFA) module presented in [53].

VI. WHOLE-BODY CONTROLLER

In this section, we describe the WBC that tracks planned trajectories $\mathbf{x}^{\text{p}}$ and $\mathbf{u}^{\text{p}}$ provided by the NMPC. The WBC first computes feed-forward $\mathbf{W}_{\text{ff}}^{\text{d}}$ and feedback $\mathbf{W}_{\text{fb}}^{\text{d}}$ wrenches from the planned trajectories and then the sum of these wrenches are mapped into GRFs through the QP optimization (17). The WBC also maps the GRFs into the joint torques $\boldsymbol{\tau}^*$. This joint torque along with low-impedance feedback torque $\boldsymbol{\tau}_{\text{fb}}$ results in the total torque $\boldsymbol{\tau}_{\text{d}}$ required by the low-level joint torque control block. Refer to Fig. 2 for the block representation of WBC inside our locomotion framework.

A. WBC INTERFACE

In our planning framework, the NMPC runs at re-planning frequency of 25 Hz whereas the WBC requires state and control inputs at 250 Hz (we will call this the *WBC frequency*). Hence, we introduce a WBC interface block that re-samples state and control inputs at the WBC frequency. In particular, in order to obtain the desired $\mathbf{u}^{\text{d}}$ we use a zero-order hold filter of $\mathbf{u}^{\text{p}}$. The planned states $\mathbf{x}^{\text{p}}$ from the NMPC, instead, are re-sampled with a linear interpolation to obtain $\mathbf{x}^{\text{d}}$. [1]

---

[1]The rigorous approach is to use the model (3) to predict the evolution of the system in the T_{s} time interval, considering the $\mathbf{u}^{\text{p}}$ coming from the NMPC, but for the motions considered in this paper the result is very similar, so a linear interpolation is a fair approximation.

Finally, the feed-forward wrench $\mathbf{W}_{ff} \in \mathbb{R}^6$ is computed from the desired GRFs $\mathbf{u}^d$ as:

$$\mathbf{W}_{ff}^d = \left[\sum_{i=1}^4 \mathbf{u}_i^d \quad \sum_{i=1}^4 \mathbf{p}_{cf,i}^d \times \mathbf{u}_i^d \right]^\top \quad (14)$$

B. FEEDBACK WRENCH

We use the approach of [4] to define desired feedback wrench obtained from a Cartesian impedance and briefly recall it next for completeness:

$$\mathbf{W}_{fb}^d = \mathbf{K} \begin{bmatrix} \mathbf{p}_c^d - \mathbf{p}_c \\ \mathbf{e}(_w\mathbf{R}_b^\top {}_w\mathbf{R}_d) \end{bmatrix} + \mathbf{D} \begin{bmatrix} \mathbf{v}_c^d - \mathbf{v}_c \\ \boldsymbol{\omega}_b^d - \boldsymbol{\omega}_b \end{bmatrix} \quad (15)$$

where $_w\mathbf{R}_b$ and $_w\mathbf{R}_d \in \mathbb{R}^{3\times3}$ are the rotation matrices representing actual and desired orientation of the base with respect to the inertial frame, respectively, $\mathbf{e}(\cdot): \mathbb{R}^{3\times3} \to \mathbb{R}^3$ is a mapping from a rotation matrix to the associated rotation vector. Matrices $\mathbf{K}$ and $\mathbf{D}$ are diagonal matrices containing the proportional and derivative gains and they can be interpreted as impedances.

Remark: At each re-planning instance of NMPC, the state reference is computed from the current state of the robot $\hat{\mathbf{x}}_0$. Thus, at each re-planning instance the feedback term is nullified.

C. PROJECTION OF THE GRFs

While the feedforward wrenches $\mathbf{W}_{ff}^d$ provided by MPC satisfy the friction cone and unilateral constraints by construction, this guarantee is lost with the addition of the feedback term $\mathbf{W}_{fb}^d$ to the wrenches. Therefore, one needs to project the total wrenches $\mathbf{W}_{ff}^d + \mathbf{W}_{fb}^d$ onto the set of wrenches that satisfy the constraints. The matrix representation

$$\underbrace{\begin{bmatrix} \delta_1 \mathbf{I} & \dots & \delta_4 \mathbf{I} \\ \delta_1 [\mathbf{p}_{cf,1} \times] & \dots & \delta_4 [\mathbf{p}_{cf,4} \times] \end{bmatrix}}_{\mathbf{A}} \underbrace{\begin{bmatrix} \mathbf{f}_1 \\ \vdots \\ \mathbf{f}_4 \end{bmatrix}}_{\mathbf{f}} = \underbrace{\mathbf{W}_{ff}^d + \mathbf{W}_{fb}^d}_{\mathbf{b}}$$

$$(16)$$

is derived from a simplified SRBD model [4] and allows us to map the desired wrenches into GRFs. To compute the desired GRFs $\mathbf{f}^d$ we solve the following QP:

$$\mathbf{f}^d = \underset{\mathbf{f}}{\arg\min} \ \| \mathbf{A}\mathbf{f} - \mathbf{b} \|_{\mathbf{S}}^2 + \| \mathbf{f} - \mathbf{u}^d \|_{\mathbf{T}}^2 \quad (17a)$$

$$\text{s.t.} \quad \underline{\mathbf{d}} \leq \mathbf{C}\mathbf{f} \leq \overline{\mathbf{d}} \quad (17b)$$

The term $\| \mathbf{f} - \mathbf{u}^d \|_{\mathbf{T}}^2$ in the cost (17) allows the tracking of the desired forces $\mathbf{u}^d$ received from the NMPC. Matrices $\mathbf{S} \in \mathbb{S}_+^6$ and $\mathbf{T} \in \mathbb{S}_+^{12}$ are positive-definite weight matrices. Inequality (17b) encodes the friction cone and unilateral constraints similar to (7) for which further details can be found in [4]. It is important to note that gravity compensation is already incorporated in the NMPC formulation through the SRBD model.

D. MAPPING GRFs TO JOINT TORQUES

The GRFs $\mathbf{f}^d$ must be mapped into joint torques $\boldsymbol{\tau}^*$. We do so by exploiting the joint dynamics:

$$\boldsymbol{\tau}^* = -\mathbf{J}(\mathbf{q})^\top \mathbf{f}^d + \mathbf{h}(\mathbf{q}, \dot{\mathbf{q}}) \quad (18)$$

where $\mathbf{J}(\mathbf{q}) \in \mathbb{R}^{n_u \times n_x}$ is the contact Jacobian and $\mathbf{h}(\mathbf{q}, \dot{\mathbf{q}})$ the vector of gravity/Coriolis terms in the leg joint dynamics. We neglect the joint acceleration contribution, because it is very small with respect to the other terms.

E. JOINT-SPACE PD

A $1\,\mathrm{kHz}$ Joint-Space PD is put in cascade with the WBC before sending torques to the low-level controller. In this way, we track the desired trajectories of the swinging legs and we increase the robustness in case a foot loses contact with the ground. The WBC interface provides the joint trajectories $\mathbf{q}^d$ and $\dot{\mathbf{q}}^d$ required by the Joint-Space PD. To compute the joint trajectories, inverse kinematics is required which in turn needs the swing trajectory $\mathbf{p}_f^{sw}$. We define the swing frame $\mathcal{S}$ [50] (Fig. 7), whose X-axis is aligned with the vector that links lift-off and touchdown point ($\mathbf{L}_{sw}$), Y-axis is perpendicular to the X-axis of the swing frame and to the Z-axis of the world frame. Finally the Z-axis is such that $\mathcal{S}$ is a counter-clockwise coordinate system. The origin of the swing frame $\mathcal{S}$ coincides with the lift-off point. In this way the swing trajectory lies on the X-Z plane and we shape it as a semi-ellipse with L_{sw} and H_{sw} as lengths of the axes:

$$_\mathcal{S}\mathbf{p}_f^{sw} = \begin{bmatrix} \frac{L_{sw}}{2}(1 - \cos(\pi f_{sw} t_{sw})) \\ 0.0 \\ H_{sw}\sin(\pi f_{sw} t_{sw}) \end{bmatrix} \quad (19)$$

where t_{sw} is the time elapsed from the beginning of a swing and $f_s = 1/T_{sw}^d$ is the swing frequency. We map $_\mathcal{S}\mathbf{p}_f^{sw}$ and its derivative in the inertial frame $\mathcal{W}$ to obtain $\mathbf{p}_f^{sw}$ and $\dot{\mathbf{p}}_f^{sw}$, respectively. Finally, after evaluating the relative foot position $_c\mathbf{p}_{cf}$ and velocity $_c\dot{\mathbf{p}}_{cf}$ we can obtain $\mathbf{q}^d$ and $\dot{\mathbf{q}}^d$ via inverse kinematics.

VII. REAL-TIME ITERATION FOR NMPC

One of the main drawbacks of NMPC is its computational burden, thus efficient tailored algorithms are necessary in order to achieve fast sampling rates for complex systems with fast dynamics. While many approaches have been developed for optimal control, a complete discussion about all possible approaches is beyond the scope of this paper. We focus on direct multiple shooting methods derived from Sequential Quadratic Programming (SQP) that have been specifically developed for real-time NMPC [34].

In multiple shooting methods both state $\mathbf{x}$ and control input $\mathbf{u}$ are decision variables unlike in single shooting where the decision vector only includes the control input. We ought to stress that this does not increase the computational complexity with respect to single shooting (where computations are moved from linear algebra to the evaluation of derivatives). Furthermore, multiple-shooting allows one to provide an initial guess also for the state trajectory, which is typically beneficial for unstable systems in an NMPC context [34].

SQP is a popular algorithm which solves an NLP by iteratively solving local quadratic approximations (QPs) of the problem [35]. At each SQP iteration, the solution from the

previous step is recycled to define an initial guess $(\mathbf{x}_k^L, \mathbf{u}_k^L)$, which is then used to construct a QP approximation of the NLP (1), given by

$$\min_{\Delta\mathbf{x}, \Delta\mathbf{u}} \sum_{k=0}^{N-1} \frac{1}{2} \begin{bmatrix} \Delta\mathbf{x}_k \\ \Delta\mathbf{u}_k \end{bmatrix}^\top \mathbf{H}_k \begin{bmatrix} \Delta\mathbf{x}_k \\ \Delta\mathbf{u}_k \end{bmatrix} + \mathbf{J}_k^\top \begin{bmatrix} \Delta\mathbf{x}_k \\ \Delta\mathbf{u}_k \end{bmatrix} \tag{20a}$$

$$\text{s.t.} \quad \Delta\mathbf{x}_0 = \hat{\mathbf{x}}_0 - \mathbf{x}_0^L, \tag{20b}$$

$$\Delta\mathbf{x}_{k+1} = \mathbf{A}_k\Delta\mathbf{x}_k + \mathbf{B}_k\Delta\mathbf{u}_k + \mathbf{r}_k, \tag{20c}$$

$$\mathbf{C}_k\Delta\mathbf{x}_k + \mathbf{D}_k\Delta\mathbf{u}_k + \mathbf{h}_k \geq 0, \tag{20d}$$

where, $\Delta\mathbf{x}_k = \mathbf{x}_k - \mathbf{x}_k^L, \Delta\mathbf{u}_k = \mathbf{u}_k - \mathbf{u}_k^L, \hat{\mathbf{x}}_0$ is the current system state, and

$$\mathbf{A}_k = \left.\frac{\partial f(\mathbf{x}, \mathbf{u}, \mathbf{a}_k)}{\partial \mathbf{x}}\right|_{\mathbf{x}_k^L, \mathbf{u}_k^L}, \qquad \mathbf{B}_k = \left.\frac{\partial f(\mathbf{x}, \mathbf{u}, \mathbf{a}_k)}{\partial \mathbf{u}}\right|_{\mathbf{x}_k^L, \mathbf{u}_k^L},$$

$$\mathbf{C}_k = \left.\frac{\partial h(\mathbf{x}, \mathbf{u}, \mathbf{a}_k)}{\partial \mathbf{x}}\right|_{\mathbf{x}_k^L, \mathbf{u}_k^L}, \qquad \mathbf{D}_k = \left.\frac{\partial h(\mathbf{x}, \mathbf{u}, \mathbf{a}_k)}{\partial \mathbf{u}}\right|_{\mathbf{x}_k^L, \mathbf{u}_k^L},$$

$$\mathbf{r}_k = g\left(\mathbf{x}_k^L, \mathbf{u}_k^L, \mathbf{a}_k\right) - \mathbf{x}_{k+1}^L, \quad \mathbf{h}_k = h\left(\mathbf{x}_k^L, \mathbf{u}_k^L, \mathbf{a}_k\right)$$

$$\mathbf{J}_k = \mathbf{W}_k \begin{bmatrix} \mathbf{x}_k^L - \mathbf{x}_k^{\text{ref}} \\ \mathbf{u}_k^L - \mathbf{u}_k^{\text{ref}} \end{bmatrix} \tag{21}$$

Matrix $\mathbf{H}_k$ is the diagonal blocks of a suitable approximation of the Lagrangian Hessian. Since our problem relies on a least-squares cost, we adopt the popular Gauss-Newton Hessian approximation [35] that gives $\mathbf{H}_k = \mathbf{W}_k$.

While in SQP one solves several QPs until convergence is reached, the RTI scheme consists in solving a single QP per sampling time. This is motivated by the observation that in NMPC two subsequent problems have very similar solutions. Therefore, by reusing the solution of the previous NMPC problem, one obtains a very good initial guess for the next problem, which essentially only needs to correct for external perturbations and model mismatch. For all details on the RTI scheme, we refer to [26, 35] and references therein. We limit ourselves to observe that, since $(\mathbf{x}_k^L, \mathbf{u}_k^L)$ is known *before* the next state measurement is available, one can already evaluate the functions and their derivatives (21) before the initial state $\hat{\mathbf{x}}_0$ is available. Consequently, the QP can be constructed and prepared beforehand; note that this also includes the first factorization of the QP Hessian. Once $\hat{\mathbf{x}}_0$ is available, one only has to finish solving the QP. Therefore, while the overall sampling time must still be long enough to prepare the next QP, the latency between the time at which $\hat{\mathbf{x}}_0$ is available and the time at which the control input can be applied to the system is very small.

Note that in the RTI scheme proposed above, the functions and their derivatives (21) are evaluated along a guess obtained from the previous solution, rather than along the reference trajectory. Another important aspect to highlight is the fact that there exist several approaches to compute (21). One choice consists of first linearizing the continuous-time system dynamics and then using the matrix exponential to obtain a discrete-time linear system. This approach presents some advantages, but can be computationally demanding.

For the time-varying and infeasible references, however, it is preferred to *first discretize and then linearize* [35]. In this work we deal with time-varying and infeasible references, hence we opt for first discretize and then linearize approach. An advantage of this approach is that after numerically approximating the discrete-time dynamics, the linearization can be obtained at a desired accuracy.

A very popular way to obtain discret-time dynamics is with the explicit Euler integrator, which is computationally inexpensive, but can be inaccurate and unstable. Therefore, it is usually more efficient to resort to higher-order integration schemes, such as, e.g., the popular Runge-Kutta methods. Finally, we should further stress that there also exist implicit integration schemes, which require more computations per step, but they are typically much more stable and accurate than explicit schemes for some classes of systems. Unfortunately, the selection of the least computationally demanding integrator which delivers sufficient accuracy depends on the problem setting and typically requires some trial-and error approach, which can be educated using some guidelines based on the theoretical properties of each integrator [40, 41, 42, 43, 54].

In this work, we relied on the RTI implementation provided by `acados` [27], which consists of tailored efficient implementations of QP solvers, numerical integration schemes, and all other components of the RTI scheme.

VIII. RESULTS

In this section we discuss the implementation details and results obtained from the simulations and experiments with the NMPC scheme proposed in Section III.

A. IMPLEMENTATION DETAILS

To check the efficacy of our RTI based NMPC algorithm with the proposed features mentioned in Section IV, we performed several simulations and experiments in challenging scenarios. The simulation and experiments were performed on the HyQ robot of mass $m = 87\,\text{kg}$. The CoM is computed considering the mass of the individual link of the robot and the actual position of the link's CoM. The position of the link's CoM in their local frame is obtained from their CAD models. The feedback gains used in the WBC are $\mathbf{K} = \text{diag}(1500, 1500, 1500, 100, 100, 100)$ and $\mathbf{D} = \text{diag}(1000, 1000, 1000, 50, 50, 50)$. We chose the weights $\mathbf{S} = \text{diag}(5, 5, 10, 10, 10, 10)$ and $\mathbf{T} = \text{diag}(1000, \cdots, 1000)$ for the QP (17). The parameters and weights used by the NMPC are reported in Table 1 and 2, respectively. In all of our simulations and experiments, we do not set any weights on the CoM position $(\mathbf{p}_c)$, roll (ϕ) and pitch (θ) tasks because we wanted the NMPC to sort out these CoM trajectories autonomously.

1) Discretization
For the discretization of the dynamic constraints, we mainly investigated two integration schemes, i.e, a single step of the explicit Euler of order 1 and implicit midpoint method

TABLE 1. NMPC parameters

Parameter	Symbol	Value	Unit
Number of state	n_x	12	-
Number of control inputs	n_u	12	-
Number of model parameters	n_a	16	-
Prediction horizon	T	2	s
Sampling time	T_{s}	0.04	s
Number of control intervals	N	50	-
Friction coefficient	μ	0.7	-
GRFs lower bound	$\underline{\mathbf{f}}_{\mathrm{z}}$	0	N
GRFs upper bound	$\overline{\mathbf{f}}_{\mathrm{z}}$	500	N
Hip-to-foot distance reference	$_{\mathcal{C}}\mathbf{P}_{\mathrm{hf},i}^{\mathrm{ref}}$	$(0.0, 0.0, -0.55)$	m
Regularization parameter	ρ	3×10^{-5}	-

TABLE 2. Weights used in the NMPC

Cost	Weight	Value
State	$\mathbf{Q}_{\mathbf{P}_c}$	$\mathrm{diag}(0, 0, 0)$
	$\mathbf{Q}_{\mathbf{v}}$	$\mathrm{diag}(100, 100, 100)$
	$\mathbf{Q}_{\mathbf{\Phi}}$	$\mathrm{diag}(0, 0, 100)$
	$\mathbf{Q}_{\mathbf{\omega}}$	$\mathrm{diag}(100, 100, 1000)$
Force	$\mathbf{R}_{\mathrm{x}}$	1×10^{-3}
	$\mathbf{R}_{\mathrm{y}}$	1×10^{-3}
	$\mathbf{R}_{\mathrm{z}}$	8×10^{-4}
Mobility	$\mathbf{M}_{\mathrm{x}}$	1×10^{-4}
	$\mathbf{M}_{\mathrm{y}}$	2×10^{-3}
	$\mathbf{M}_{\mathrm{z}}$	1000
Force robustness	$\mathbf{P}_{\mathrm{x}}$	100
	$\mathbf{P}_{\mathrm{y}}$	100
	$\mathbf{P}_{\mathrm{z}}$	1

of order 2 due to their low computational complexity that favors our real-time implementation needs. We chose the implicit midpoint method of order 2 because of its stability and accuracy properties. The sampling time $T_{\mathrm{s}} = 40\,\mathrm{ms}$ was chosen and it was sufficient to conduct NMPC computation *online* along with the other necessary computations for the re-planning.

2) NMPC software
We use the `acados` software package [27] to implement the RTI scheme described in Section VII. Since `acados` comes with a Python interface allowing rapid prototyping, we first tuned the algorithm in simulation and then used the generated `C`-code to perform real experiments. We employ the QP solver High-Performance Interior Point Method (HPIPM) [55], which exploits the sparsity structure of the MPC QP sub-problem (20), and supports inequality constraints.

The computation time required by the NMPC was in the range of 5-7 ms with the prediction horizon of 2 s and control intervals N equal to 50 on the on-board computer (a Quad Core Intel Pentium PC104 @ 1 GHz) of HyQ for all the experiments. This computation time corresponds to the feedback phase of the RTI scheme where the QP (20) is solved after receiving the current state of the robot. The preparation phase of the RTI takes about 2-3 ms which is a fraction of the sampling time we chose. Refer to Section VII for more details on these phases of the RTI scheme. Even though the computation time of NMPC is mostly consistent, we observed some outliers. Hence we opted for a conservative approach to run the NMPC at 25 Hz to guarantee that the computation time stays always less than 40 ms. Besides the computation time of the NMPC, we also account for the time required by other blocks such as reference generator so that the total computation time does not exceed 40 ms.

3) Integration with the locomotion framework
The NMPC is integrated in a ROS node that publishes $\mathbf{x}^{\mathrm{P}}$ and $\mathbf{u}^{\mathrm{P}}$ at a frequency of 25 Hz. The on-board computer along with our locomotion framework (WBC Interface, WBC, etc., illustrated in Fig. 2) runs a real-time node that subscribes to the topic of the NMPC ROS node. ROS is not a real-time operating system, so it can introduce quite a significant and unpredictable communication delay if the NMPC is run on an external (e.g. more powerful) computer. These delays are difficult to compensate for and they can cause a loss of synchronization between the NMPC ROS node and the WBC interface. Therefore, we decided to launch the NMPC node natively on the on-board computer to avoid communication delays between two different computers. Even though we chose not to use a more powerful dedicated off-board computer for the NMPC ROS node, we obtained a better performance in the overall implementation by avoiding the communication delays of ROS.

B. SIMULATIONS
We show our NMPC planner in action on challenging terrain starting with simulations. These simulations are pallet crossing, walking over unstructured rough terrain and walk into a V-shaped Chimney.

1) Pallet crossing
In this simulation, HyQ traverses pallets of different heights, placed at varying distances form each other. This simulation highlights the importance of including mobility in the NMPC formulation (2c). In particular, the simulation scenario includes a set of pallets, each one of 1 m length, with variable heights between 0.13 and 0.17 m and placed at unequal gap lengths ranging from 0.2 and 0.7 m. We performed multiple trials commanding the robot to move forward at different velocities i.e., 0.05 m/s and 0.1 m/s to show the repeatability of our approach. To avoid stepping on undesired locations such as pallet edges and to prevent foot or shin collisions, the nominal footholds are adjusted by using the VFA (see Section V-2). Fig. 9 shows the results of five different trials for each of the commanded velocities. The middle plot shows the pitch angle θ of the robot as it traverses the scenario. Since the robot is only commanded to move along its X direction with a constant forward velocity and the foot locations are provided as known quantities to the NMPC, the adjustment in pitch is the result of minimizing the deviation

from the hip-to-foot distance configuration corresponding to high mobility for all four legs. Without this feature, the robot would maintain a constant horizontal orientation (refer Fig. 8) eventually reaching low mobility in some legs as shown in the attached video[1] for a single pallet simulation. We have
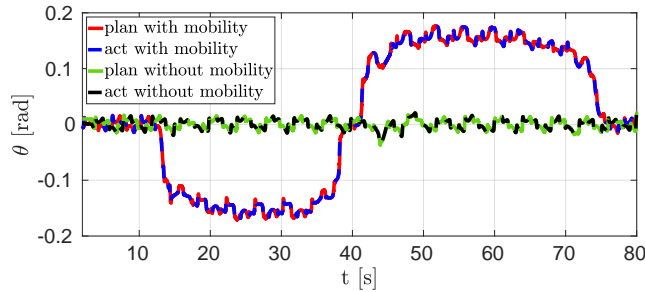

FIGURE 8. Comparison of the robot pitch θ in simulation for with and without mobility cost in the NMPC. The red and green lines represent the planned pitch values delivered by the NMPC. The blue and black dashed lines are actual pitch of the robot. Without mobility cost, the robot maintains a horizontal orientation, whereas the robot pitches to improve leg mobility when it is included.

also showed in the accompanying video the simulation of a walk on randomly generated rough terrain (using terrain generation tool by [56]) with the forward velocity of $0.3\,\mathrm{m/s}$ further stressing the advantages of mobility cost mentioned earlier.
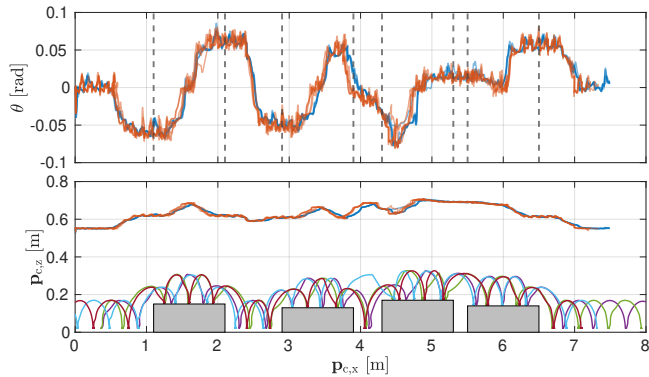
FIGURE 9. Simulation of pallet crossing scenario for five different trials commanding the robot to cross at 0.1 m/s (blue) and 0.05 m/s (red). The top graph shows the pitch angle and the dashed vertical lines indicate the edges of the pallets for that specific location in the plot. The bottom graph shows the CoM Z position for all the trials and the feet trajectories for one of the trials performed at 0.1 m/s. The color of swings are related to the different legs.

2) Walk into a V-shaped chimney

In this simulation, we show HyQ walking at $0.03\,\mathrm{m/s}$ commanded velocity in the X direction into a V-shaped chimney with friction coefficient $\mu = 0.7$ and walls inclined at $35°$ to the ground. This simulation exploits the cone constraints and force robustness cost defined inside the NMPC formulation that is vital for the success of this task. The robot receives online an update of the map of the environment through an

[1]https://www.dropbox.com/sh/le7sunqq6sebbou/AADi0S1Lbg3iiS63H HBstyoja?dl=0

on-board camera to get the information about the normals at the location of the contact. These normals are used to formulate the force robustness cost (2d) in the contact frame $\mathcal{K}$. With this cost, the NMPC provides optimal GRFs to stay close to the normals of the friction cones at the contacts.

As shown in the accompanying video, without the cone constraints the robot slips while climbing the chimney and ultimately falls. When the force robustness cost is enabled, the forces are regularized to stay in the middle of the cones, thanks to the robustness feature described in Section IV-C. In this case the robot walks successfully into the chimney. In Fig. 10, it can be seen that the longitudinal and lateral components of the GRF at the Left-Front (LF) foot stay within the bound μf_z (in red) imposed by the cone constraints. Moreover, Fig. 11 plots the normal versus the tangential force of the GRF together with the cone bound μf_z (red line). The picture shows that the GRF stays well within the bound without any violation. Therefore, including the force regularization term enables the NMPC to account for the estimation error in the orientation of contact normals and increase robustness to the external disturbances.
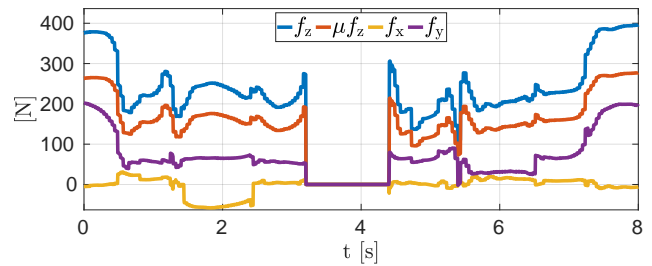

FIGURE 10. Walk into a V-shaped chimney simulation: GRFs of LF leg for a single gait cycle with cone constraints and regularization cost. Both the longitudinal f_x and lateral f_y lie conservatively within the bound μf_z imposed by cone constraints.
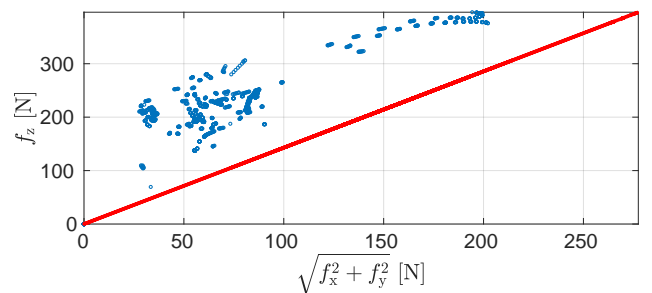

FIGURE 11. Walk into a V-shaped chimney simulation: Normal force f_z versus tangential force $\sqrt{f_x^2 + f_y^2}$ of the LF leg for a single gait cycle expressed in the contact frame. The red line is the cone bound μf_z.

Apart from the simulation mentioned above, we also have added in the attached video, the simulations regarding the ZMP margin (see Section IV-B) and the importance of the re-planning at a higher rate. For the ZMP margin simulation, the robot is pushed with $200\,\mathrm{N}$ of lateral force for $1\,\mathrm{s}$ both in case of sufficient (higher weight on GRFs Z) and no (lower weight on GRFs Z) ZMP margin. The ZMP margin plots for
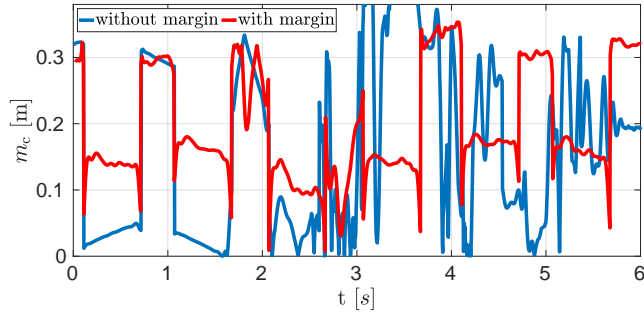
13

FIGURE 12. Plot of the ZMP margin m_c used to measure locomotion stability. The robot is pushed immediately after $2\,\mathrm{s}$ with lateral force of $200\,\mathrm{N}$ while walking on a flat terrain at $0.1\,\mathrm{m/s}$ CoM X velocity. The discontinuities are due to the switching between 3/4 stance legs in a crawl gait.

this simulation can be seen in Fig. 12 where the ZMP margin is improved in case of the red line compared to the blue one because the GRFs Z components are penalized relatively more (100 times) for the red line. Because of the improved margin the robot walks stably, whereas it falls while walking when there is no margin.

In the second simulation, the robot is commanded with a constant CoM X velocity and heading velocity simultaneously. In case of re-planning at lower rate of $0.8\,\mathrm{Hz}$, the robot becomes unstable and falls due to increase in the model uncertainties and tracking errors. We would like to stress that when the re-planning is done at a lower frequency than $25\,\mathrm{Hz}$, the robot is in open-loop for the time interval between two consecutive re-planning instances, hence, it is no more NMPC but an online open-loop trajectory optimization. On the other hand, at a higher re-planning frequency of $25\,\mathrm{Hz}$ the robot walks successfully because the NMPC compensates for the model uncertainties and tracking errors.

C. EXPERIMENTS

We performed three different experiments to demonstrate the real-time implementation of our NMPC running on the on-board computer of the robot as follows.

1) Omni-directional walk

With this experiment, we show the omni-directional walk performed by HyQ with the NMPC on a flat terrain. This experiment validates that the NMPC computes feasible trajectories after receiving different velocity commands from the user while walking. In this experiment, the robot is commanded with a longitudinal velocity $_{\mathcal{H}}\mathbf{v}_{c,x}^{usr}$ by the user to walk forward/backward and then a lateral velocity $_{\mathcal{H}}\mathbf{v}_{c,y}^{usr}$. Finally, a heading velocity ω_z^{usr} is commanded to turn in the left/right direction. Fig. 13 shows the CoM X-Y position and yaw angle of the robot base and it can be noticed that the actual values track very closely the planned trajectories provided by the NMPC. Fig. 14 depicts the deviation of the actual velocities from the reference values while following the planned trajectories form NMPC. It can be seen in Fig. 15 that the GRFs generated by the WBC are compliant with

the planned values $\mathbf{u}^p$ and again the actual values of GRFs track closely the planned values. From these plots, it can be observed that the continuous re-planning with NMPC plays an important role to achieve good tracking of the planned trajectory.
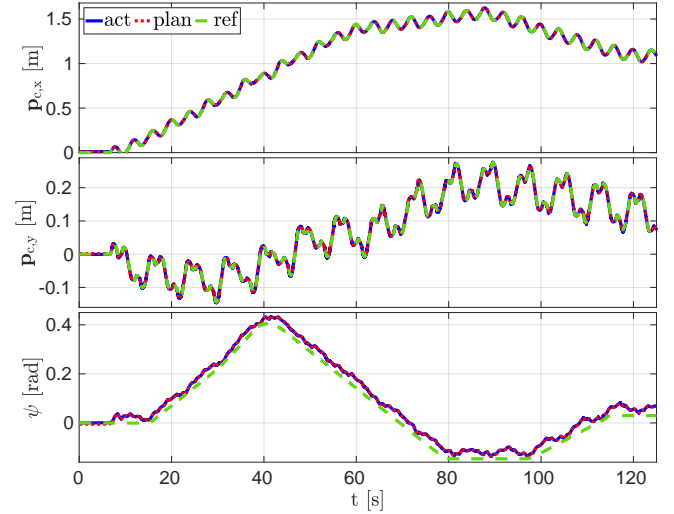

FIGURE 13. CoM X-Y position and yaw ψ in omni-directional walk experiment. The blue, dotted red and dashed green line represent the actual, planned and reference values, respectively.

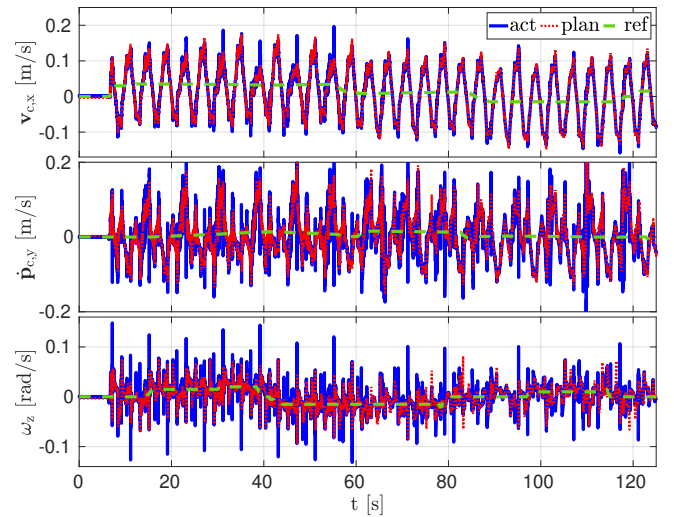**FIGURE 14.** The longitudinal $\dot{\mathbf{p}}_{c,x}$, lateral $\dot{\mathbf{p}}_{c,y}$ and angular $\boldsymbol{\omega}_z$ velocity of the robot in omni-directional walk experiment. The blue, dotted red and the green line represent the actual, planned and reference values, respectively.

2) Traversing a static pallet

The purpose of this experiment is to demonstrate that the mobility cost (2c) incorporated in the NMPC formulation provides the necessary body pitch for the robot to traverse over a static pallet while maintaining good leg mobility. The pallet used in this experiment is $0.13\,\mathrm{m}$ in height and $0.8\,\mathrm{m}$ in length. Fig. 16 shows that the robot pitches up while climbing up the pallet and pitches down consequently while
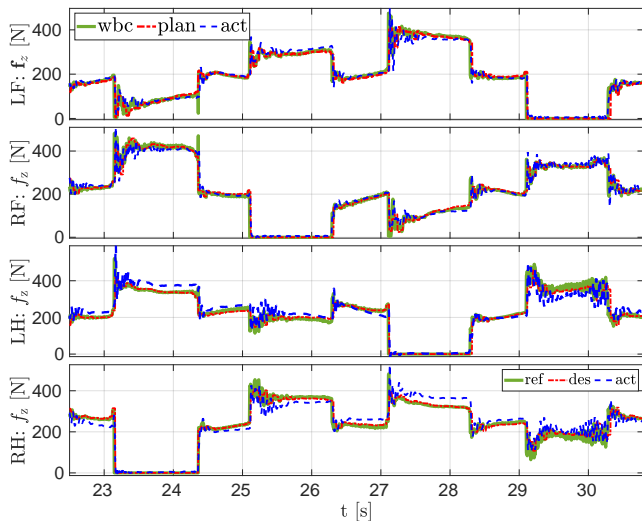
FIGURE 15. GRFs from one gait cycle in the omni-directional walk experiment (We show only one cycle for better visibility of the data). The green, dotted red and dashed blue line represent the output from WBC i.e., $f_{z,i}^{d}$, planned and reference values, respectively.

climbing down from the pallet. As shown in the attached video in simulation, when the mobility cost is deactivated, the NMPC maintains the horizontal base orientation. This causes a reduced hip-to-foot distance while stepping up/down on the pallet ultimately resulting in low leg mobility. When mobility cost is activated, it directs the NMPC solution to achieve the necessary pitch that allows to maintain the hip-to-foot distance at the reference value and hence the leg mobility is improved. Moreover, the VFA provides the corrected foot position (i.e., to avoid shin or feet collisions with the edges of the pallet) to the NMPC and this further enhances the overall locomotion.
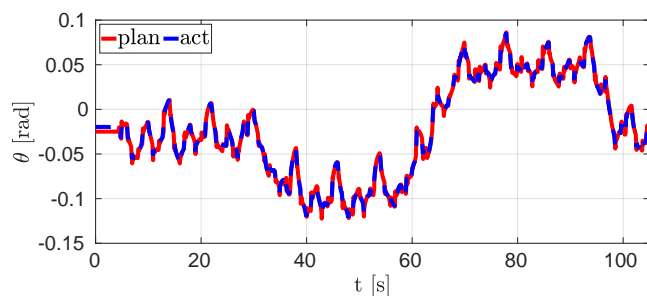

FIGURE 16. Planned (red) and Actual (dashed blue) pitch of the robot base while traversing a static pallet in the experiment at a commanded CoM X velocity of $0.03\,\mathrm{m/s}$.

3) Traversing a repositioned pallet

In this experiment we test our NMPC to plan the robot motion in real-time by adapting the changes in the environment with the help of VFA. As it can be seen from the attached video, when the pallet ($0.13\,\mathrm{m}$ in height and $0.8\,\mathrm{m}$ in length) is pushed in front of HyQ while walking, the heightmap detects the pallet and the VFA provides updated foot locations to

the NMPC. The NMPC after receiving these updated foot locations delivers a solution by pitching up the robot base in order to adapt to the change in the environment while maintaining the mobility. Even though the mobility cost is defined for the stance legs, it is interesting to notice that the NMPC decides to adjust the base pitch while swinging the RF leg onto the pallet (see Fig. 17) by forecasting the change in hip-to-foot distance at the touchdown. This experiment highlights the advantage of the predictive control over the traditional control apporaches for its ability to incorporate the knowledge of the future states. It also validates the effectiveness of our mobility cost in the NMPC coupled with the VFA to adapt to the changes in the locomotion environment.
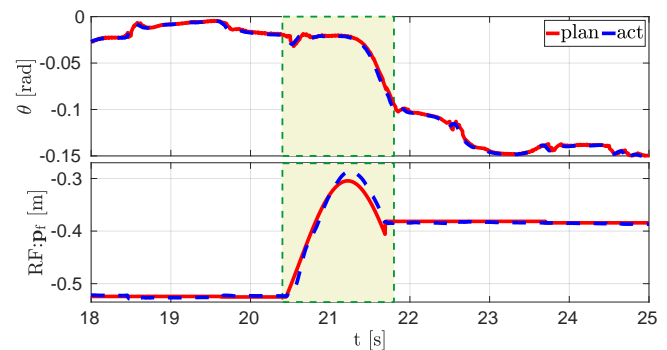

FIGURE 17. Robot base pitch achieved during the swing of RF leg while traversing a repositioned pallet in the experiment a commanded CoM X velocity of $0.05\,\mathrm{m/s}$. The red and dashed blue line are planned and actual values.

IX. CONCLUSION

In this work, we have demonstrated in experiments a real-time NMPC which leverages optimization of leg mobility to achieve terrain adaptation. The contact sequence parameters embedded inside the SRBD model allows us to encode the complementarity constraints directly, and without a need to enforce these constraints separately in the NMPC. We exploited the RTI scheme for our NMPC that enable us to close the loop at $25\,\mathrm{Hz}$ on the NMPC with a prediction horizon of $2\,\mathrm{s}$. Closing the loop on NMPC at $25\,\mathrm{Hz}$ allows us to compensate for the state drifts due to model uncertainties and tracking errors, and also adapt to the changes in the environment while following user velocity commands both in the simulations and experiments.

In our NMPC, the mobility cost penalizes the hip-to-foot distance from a reference value corresponding to a high mobility factor and hence it directs the NMPC to compute essential robot orientation to maintain a high mobility while respecting the kinematic limits. This is evident from the pallet experiments where we also included VFA to correct undesired foot positions defined by the heuristics and avoid possible foot and shin collision. Accounting for the ZMP margin in our NMPC improved the locomotion stability of the robot in all of our experiments and simulation by keeping a sufficiently large ZMP margin from support polygon boundaries. Incorporating a force robustness term in

the NMPC ensures that the GRFs stay close to the contact normals and hence, it enables the robot to cope with the estimation error of the orientation of the contact normals.

With our NMPC, we have performed successful dynamic locomotion in simulation as well as in the experiments on different rough terrains. In our future work we would like to extend our NMPC to optimize the step timing and foot locations. Additionally, the reference generator does not provide references by rejecting the external disturbances acting on the robot state, hence the robot complies transparently with these disturbances. Therefore, in the future we plan to empower the reference generator to reject disturbances to bring back the robot from a perturbed state to a state coherent with the user commands.

.

APPENDIX A ANGULAR VELOCITY
We employ the Z-Y-X convention [39] for the Euler angles sequence $\boldsymbol{\Phi} = (\phi, \theta, \psi)^{\top}$ to represent the orientation of the robot base where, ϕ, θ and ψ are the roll, pitch and yaw, respectively. The angular velocity in inertial and CoM frame is related to the Euler angle rates with the following relations:

$$\boldsymbol{\omega} = \mathbf{E}(\boldsymbol{\Phi})\,\dot{\boldsymbol{\Phi}} \tag{22}$$

$$_c\boldsymbol{\omega} = \mathbf{E}'(\boldsymbol{\Phi})\,\dot{\boldsymbol{\Phi}} \tag{23}$$

$\mathbf{E}(\boldsymbol{\Phi})$ and $\mathbf{E}'(\boldsymbol{\Phi})$ are the *Euler angle rates matrix* and *conjugate Euler angle rates matrix* respectively given by,

$$\mathbf{E}(\boldsymbol{\Phi}) = \begin{bmatrix} \cos(\theta)\cos(\psi) & -\sin(\psi) & 0 \\ \cos(\theta)\sin(\psi) & \cos(\psi) & 0 \\ -\sin(\theta) & 0 & 1 \end{bmatrix} \tag{24}$$

$$\mathbf{E}'(\boldsymbol{\Phi}) = \begin{bmatrix} 1 & 0 & -\sin(\theta) \\ 0 & \cos(\phi) & \cos(\theta)\sin(\phi) \\ 0 & -\sin(\phi) & \cos(\theta)\cos(\phi) \end{bmatrix} \tag{25}$$

Remark: $\mathbf{E}$ depends on pitch and yaw, whereas $\mathbf{E}'$ on roll and pitch. Thus, the Euler angle rates $\dot{\boldsymbol{\Phi}}$ is

$$\dot{\boldsymbol{\Phi}} = \mathbf{E}^{-1}(\boldsymbol{\Phi})\,\boldsymbol{\omega} \tag{26}$$

$$\dot{\boldsymbol{\Phi}} = \mathbf{E}'^{-1}(\boldsymbol{\Phi})\,_c\boldsymbol{\omega} \tag{27}$$

REFERENCES
[1] M. H. Raibert, *Legged robots that balance*. MIT press, 1986.
[2] J. Pratt, C.-M. Chew, A. Torres, P. Dilworth, and G. Pratt, "Virtual model control: An intuitive approach for bipedal locomotion," *The International Journal of Robotics Research*, vol. 20, no. 2, pp. 129–143, 2001.
[3] 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.

[4] M. Focchi, A. del Prete, I. Havoutis, R. Featherstone, D. G. Caldwell, and C. Semini, "High-slope terrain locomotion for torque-controlled quadruped robots," *Auton. Robots*, pp. 1–14, 2016.
[5] S. Fahmi, M. Focchi, A. Radulescu, G. Fink, V. Barasuol, and C. Semini, "STANCE: Locomotion adaptation over soft terrain," *IEEE Trans. Robot. (T-RO)*, vol. 36, no. 2, pp. 443–457, Apr. 2020.
[6] C. D. Bellicoso, F. Jenelten, P. Fankhauser, C. Gehring, J. Hwangbo, and M. Hutter, "Dynamic locomotion and whole-body control for quadrupedal robots," in *2017 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2017, pp. 3359–3365.
[7] S. Kuindersma, F. Permenter, and R. Tedrake, "An efficiently solvable quadratic program for stabilizing dynamic locomotion," in *2014 IEEE International Conference on Robotics and Automation (ICRA)*, 2014, pp. 2589–2594.
[8] M. Neunert, F. Farshidian, A. W. Winkler, and J. Buchli, "Trajectory Optimization Through Contacts and Automatic Gait Discovery for Quadrupeds," *IEEE Robot. Autom. Lett.*, vol. 2, no. 3, pp. 1502–1509, 2017.
[9] M. Posa, S. Kuindersma, and R. Tedrake, "Optimization and stabilization of trajectories for constrained dynamical systems," in *2016 IEEE International Conference on Robotics and Automation (ICRA)*, 2016, pp. 1366–1373.
[10] B. Aceituno-Cabezas, C. Mastalli, D. Hongkai, M. Focchi, A. Radulescu, D. G. Caldwell, J. Cappelletto, J. C. Grieco, G. Fernando-Lopez, and C. Semini, "Simultaneous Contact, Gait and Motion Planning for Robust Multi-Legged Locomotion via Mixed-Integer Convex Optimization," in *IEEE Robot. Autom. Lett.*, 2018.
[11] O. Melon, M. Geisert, D. Surovik, I. Havoutis, and M. Fallon, "Reliable trajectories for dynamic quadrupeds using analytical costs and learned initializations," in *2020 IEEE International Conference on Robotics and Automation (ICRA)*, 2020, pp. 1410–1416.
[12] M. Neunert, M. Stauble, M. Giftthaler, C. D. Bellicoso, J. Carius, C. Gehring, M. Hutter, and J. Buchli, "Whole-Body Nonlinear Model Predictive Control Through Contacts for Quadrupeds," *IEEE Robot. Autom. Lett.*, vol. 3, no. 3, pp. 1458–1465, 2018.
[13] J. Koenemann, A. D. Prete, Y. Tassa, E. Todorov, O. Stasse, M. Bennewitz, and N. Mansard, "Whole-body model-predictive control applied to the HRP-2 humanoid," in *IEEE/RSJ Int. Conf. Intell. Robot. Syst.*, 2015, pp. 3346–3351.
[14] H. Dai, A. Valenzuela, and R. Tedrake, "Whole-body Motion Planning with Simple Dynamics and Full Kinematics," *IEEE-RAS Int. Conf. Humanoid Robot.*, 2014.
[15] D. E. Orin, A. Goswami, and S.-H. Lee, "Centroidal dynamics of a humanoid robot," *Auton. Robots*, vol. 35, no. 2-3, pp. 161–176, jun 2013.
[16] A. W. Winkler, C. D. Bellicoso, M. Hutter, and J. Buchli, "Gait and Trajectory Optimization for Legged Systems Through Phase-Based End-Effector Parameterization," *IEEE Robot. Autom. Lett.*, vol. 3, no. 3, pp. 1560–1567, 2018.
[17] G. Bledt, P. M. Wensing, and S. Kim, "Policy-regularized model predictive control to stabilize diverse quadrupedal gaits for the mit cheetah," in *2017 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2017, pp. 4102–4109.
[18] J. Di Carlo, P. M. Wensing, B. Katz, G. Bledt, and S. Kim, "Dynamic locomotion in the mit cheetah 3 through convex model-predictive control," in *2018 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2018, pp. 1–9.
[19] T. Horvat, K. Melo, and A. J. Ijspeert, "Model predictive control based framework for com control of a quadruped robot,"

in *2017 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2017, pp. 3372–3378.

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

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

[22] J. Hauser and A. Saccon, "A barrier function method for the optimization of trajectory functionals with constraints," in *Proceedings of the 45th IEEE Conference on Decision and Control*, 2006, pp. 864–869.

[23] G. Lantoine and R. P. Russell, "A hybrid differential dynamic programming algorithm for constrained optimal control problems. part 1: Theory," *Journal of Optimization Theory and Applications*, vol. 154, no. 2, pp. 382–417, Aug 2012.

[24] H. Li and P. M. Wensing, "Hybrid systems differential dynamic programming for whole-body motion planning of legged robots," *IEEE Robotics and Automation Letters*, vol. 5, no. 4, pp. 5448–5455, 2020.

[25] H. Bock and K. Plitt, "A multiple shooting algorithm for direct solution of optimal control problems*," *IFAC Proceedings Volumes*, vol. 17, no. 2, pp. 1603–1608, 1984, 9th IFAC World Congress Budapest, Hungary, 2-6 July 1984.

[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] R. Verschueren, G. Frison, D. Kouzoupis, N. van Duijkeren, A. Zanelli, B. Novoselnik, J. Frey, T. Albin, R. Quirynen, and M. Diehl, "Acados - A modular open-source framework for fast embedded optimal control," *arXiv*, 2019.

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

[29] G. Bledt and S. Kim, "Implementing regularized predictive control for simultaneous real-time footstep and ground reaction force optimization," in *2019 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2019, pp. 6316–6323.

[30] P. Fankhauser, M. Bjelonic, C. Dario Bellicoso, T. Miki, and M. Hutter, "Robust rough-terrain locomotion with a quadrupedal robot," in *2018 IEEE International Conference on Robotics and Automation (ICRA)*, 2018, pp. 5761–5768.

[31] O. Cebe, C. Tiseo, G. Xin, H.-c. Lin, J. Smith, and M. Mistry, "Online Dynamic Trajectory Optimization and Control for a Quadruped Robot," *arXiv*, 2020.

[32] C. Semini, N. G. Tsagarakis, E. Guglielmino, M. Focchi, F. Cannella, and D. G. Caldwell, "Design of hyq - a hydraulically and electrically actuated quadruped robot," *IMechE Part I: Journal of Systems and Control Engineering*, vol. 225, no. 6, pp. 831–849, 2011.

[33] C. Mastalli, I. Havoutis, M. Focchi, D. G. Caldwell, and C. Semini, "Motion planning for quadrupedal locomotion: Coupled planning, terrain mapping and whole-body control," *IEEE Trans. Robot. (T-RO)*, pp. 1–14, 2020.

[34] M. Diehl, H. G. Bock, H. Diedam, and P.-B. Wieber, "Fast Direct Multiple Shooting Algorithms for Optimal Robot Control," in *Fast Motions in Biomechanics and Robotics*, Heidelberg, Germany, 2005.

[35] S. Gros, M. Zanon, R. Quirynen, A. Bemporad, and M. Diehl, "From linear to nonlinear MPC: bridging the gap via the real-time iteration," *Int. J. Control*, vol. 93, no. 1, pp. 62–80, 2020.

[36] T. Boaventura, J. Buchli, C. Semini, and D. Caldwell, "Model-based hydraulic impedance control for dynamic robots," *Robotics, IEEE Transactions on*, vol. 31, no. 6, pp. 1324–1336, Dec 2015.

[37] S. Nobili, M. Camurri, V. Barasuol, M. Focchi, D. G. Caldwell, C. Semini, and M. Fallon, "Heterogeneous sensor fusion for accurate state estimation of dynamic legged robots," in *Proceedings of Robotics: Science and Systems*, Boston, USA, July 2017.

[38] P. Fankhauser and M. Hutter, "A Universal Grid Map Library: Implementation and Use Case for Rough Terrain Navigation," in *Robot Operating System (ROS) âĂŞ The Complete Reference (Volume 1)*, A. Koubaa, Ed. Springer, 2016, ch. 5.

[39] J. Diebel, "Representing Attitude: Euler Angles, Unit Quaternions, and Rotation Vectors," 2006.

[40] R. Quirynen, "Numerical Simulation Methods for Embedded Optimization," *Thesis*, no. January, p. 327, 2017.

[41] J. C. Butcher, *Numerical Methods for Ordinary Differential Equations*. J. Wiley, 2003.

[42] E. Hairer, S. Nørsett, and G. Wanner, *Solving Ordinary Differential Equations I*, 2nd ed., ser. Springer Series in Computational Mathematics. Berlin: Springer, 1993.

[43] E. Hairer, S. Norsett, and G. Wanner, *Solving Ordinary Differential Equations II – Stiff and Differential-Algebraic Problems*, 2nd ed., ser. Springer Series in Computational Mathematics. Berlin: Springer, 1996.

[44] O. Villarreal, V. Barasuol, P. M. Wensing, D. G. Caldwell, and C. Semini, "MPC-based controller with terrain insight for dynamic legged locomotion," in *2020 IEEE International Conference on Robotics and Automation (ICRA)*, 2020, pp. 2436–2442.

[45] A. B. Younes, J. D. Turner, D. Mortari, and J. L. Junkins, "A Survey of attitude error representations," 2012.

[46] M. Focchi, R. Featherstone, R. Orsolino, D. G. Caldwell, and C. Semini, "Viscosity-based height reflex for workspace augmentation for quadrupedal locomotion on rough terrain," in *2017 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2017, pp. 5353–5360.

[47] L. Sciavicco, B. Siciliano, and B. Sciavicco, *Modelling and Control of Robot Manipulators*, 2nd ed. Berlin, Heidelberg: Springer-Verlag, 2000.

[48] T. Yoshikawa, "Analysis and Control of Robot Manipulators with Redundancy," pp. 735–747, 1984.

[49] C. Gehring, C. D. Bellicoso, S. Coros, M. Bloesch, P. Fankhauser, M. Hutter, and R. Siegwart, "Dynamic trotting on slopes for quadrupedal robots," in *2015 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2015, pp. 5129–5135.

[50] G. Raiola, E. Mingo Hoffman, M. Focchi, N. Tsagarakis, and C. Semini, "A simple yet effective whole-body locomotion framework for quadruped robots," *Frontiers in Robotics and AI*, vol. 7, p. 159, 2020.

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

[52] T. Bretl and S. Lall, "Testing static equilibrium for legged robots," *IEEE Transactions on Robotics*, vol. 24, no. 4, pp. 794–807, 2008.

[53] O. Villarreal, V. Barasuol, M. Camurri, L. Franceschi, M. Focchi, M. Pontil, D. G. Caldwell, and C. Semini, "Fast and continuous foothold adaptation for dynamic locomotion through cnns," *IEEE Robotics and Automation Letters*, vol. 4, no. 2, pp. 2140–2147, 2019.

[54] R. Quirynen, M. Vukov, M. Zanon, and M. Diehl, "Autogenerating Microsecond Solvers for Nonlinear MPC: a Tutorial Using ACADO Integrators," *Optimal Control Applications*

and Methods, vol. 36, pp. 685–704, 2014.

[55] G. Frison and M. Diehl, "HPIPM: a high-performance quadratic programming framework for model predictive control," *arXiv*, 2020.

[56] B. Abbyasov, R. Lavrenov, A. Zakiev, K. Yakovlev, M. Svinin, and E. Magid, "Automatic tool for gazebo world construction: from a grayscale image to a 3d solid model," in *2020 IEEE International Conference on Robotics and Automation (ICRA)*, 2020, pp. 7226–7232.

NIRAJ RATHOD received the Bachelor's degree in Instrumentation and Control from College of Engineering Pune, India, in 2011. Immediately after bachelor's he worked as a System Engineer in EEEC, Pune, India, until September 2014. Thereafter, he studied the Master's in Automation and Control Engineering from Politecnico di Milano, Italy, and graduated in 2017. For his Master's thesis he collaborated with RSE S.p.A., Milan, Italy. He is currently a PhD student at IMT School for Advanced Studies Lucca in Computer Science and System Engineering. During his PhD, he was an Affiliated Researcher at Dynamic Legged System (DLS), IIT, Genova, Italy. His research interests include MPC, Optimal Control, Control of Legged Robots and Hybrid Systems.

ANGELO BRATTA received his B.Sc. degree in control Engineering from the Politecnico di Bari, in 2017, and the M.Sc. degree in Mechatronics Engineering from the Politecnico di Torino, in 2019. In March 2019, he joined the Dynamic Legged Systems (DLS) lab at Istituto Italiano di Tecnologia (IIT) for his master thesis. Since November 2019 he is a Ph.D. student of the DLS. His research interests include robotics, controls, and online trajectory optimization.

MICHELE FOCCHI is currently a Researcher at the DLS team in IIT. He received both the Bsc. and the Msc. in Control System Engineering from Politecnico di Milano. After gaining some R&D experience in the industry, in 2009 he joined IIT where he developed a micro-turbine for which he obtained an international patent and a prize. In 2013, he got a PhD in robotics, getting involved in the Hydraulically Actuated Quadruped Robot (HyQ) project. He initially was developing torque controllers for locomotion purposes, subsequently he moved to higher level (whole-body) controllers and model identification. He was also investigating locomotion strategies that are robust to uncertainties and work reliably on the real platform. Currently his research interests are focused on pushing the performances of quadruped robots in traversing unstructured environments, by using optimization-based planning strategies to perform dynamic motion planning. He published more than 42 papers in international journals and conferences and supervised several master and PhD thesis.

MARIO ZANON received the Master's degree in Mechatronics from the University of Trento, and the Diplôme d'Ingénieur from the Ecole Centrale Paris, in 2010. After research stays at the KU Leuven, University of Bayreuth, Chalmers University, and the University of Freiburg he received the Ph.D. degree in Electrical Engineering from the KU Leuven in November 2015. He held a Post-Doc researcher position at Chalmers University until the end of 2017 and is now Assistant Professor at the IMT School for Advanced Studies Lucca. His research interests include numerical methods for optimization, economic MPC, optimal control and estimation of nonlinear dynamic systems, in particular for aerospace and automotive applications.

OCTAVIO VILLARREAL is currently a postdoctoral researcher at the Dynamic Legged System (DLS) lab of the Istituto Italiano di Tecnologia (IIT). He receive his MSc degree in Mechanical Engineering track Control Engineering from TUDelft in the Netherlands in 2016. He received his PhD from IIT and the University of Genoa on 2019 working at the DLS for his work on vision-based foothold adaptation and locomotion strategies for legged robots. His current research interests are related to control of legged robots, vision-based locomotion and foothold adaptation strategies, model predictive control for legged robots and trajectory optimization.
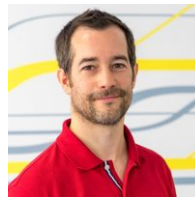
CLAUDIO SEMINI (MSc 2005, PhD 2010) is the head of the Dynamic Legged Systems (DLS) lab at Istituto Italiano di Tecnologia (IIT) that developed a number of high-performance hydraulic robots, including HyQ, HyQ2Max, and HyQReal. He holds an MSc degree from ETH Zurich in electrical engineering and information technology. He spent 2 years in Tokyo for his research: MSc thesis at the Hirose Lab at Tokyo Tech and staff engineer at the Toshiba R&D center in Kawasaki working on mobile service robotics. During his PhD and subsequent PostDoc at IIT, he developed the quadruped robot HyQ and worked on its control. Since 2012 he leads the DLS lab. Claudio Semini is the author and co-author of more than 100 peer-reviewed publications in international journals and conferences and he received several awards for them. He is also a co-founder of the Technical Committee on Mechanisms and Design of the IEEE-RAS Society. He is/was the coordinator/partner of several EU-, National and Industrial projects (including HyQ-REAL, INAIL Teleop, Moog@IIT joint lab, ESA-ANT, etc). His research interests include the construction and control of highly dynamic and versatile legged robots for field application in real-world operations, locomotion, hydraulic drives, and others.

ALBERTO BEMPORAD received his Master's degree in Electrical Engineering in 1993 and his Ph.D. in Control Engineering in 1997 from the University of Florence, Italy. In 1996/97 he was with the Center for Robotics and Automation, Department of Systems Science & Mathematics, Washington University, St. Louis. In 1997-1999 he held a postdoctoral position at the Automatic Control Laboratory, ETH Zurich, Switzerland, where he collaborated as a senior researcher until 2002. In 1999-2009 he was with the Department of Information Engineering of the University of Siena, Italy, becoming an Associate Professor in 2005. In 2010-2011 he was with the Department of Mechanical and Structural Engineering of the University of Trento, Italy. Since 2011 he is Full Professor at the IMT School for Advanced Studies Lucca, Italy, where he served as the Director of the institute in 2012-2015. He spent visiting periods at Stanford University, University of Michigan, and Zhejiang University. In 2011 he co-founded ODYS S.r.l., a company specialized in developing model predictive control systems for industrial production. He has published more than 400 papers in the areas of model predictive control, hybrid systems, optimization, automotive control, and is the co-inventor of 16 patents. He is author or coauthor of various software packages for model predictive control design and implementation, including the Model Predictive Control Toolbox (The Mathworks, Inc.) and the Hybrid Toolbox for MATLAB. He was an Associate Editor of the IEEE Transactions on Automatic Control during 2001-2004 and Chair of the Technical Committee on Hybrid Systems of the IEEE Control Systems Society in 2002-2010. He received the IFAC High-Impact Paper Award for the 2011-14 triennial and the IEEE CSS Transition to Practice Award in 2019. He is an IEEE Fellow since 2010.

● ● ●