Dynamic Complementarity Conditions and Whole-Body Trajectory Optimization for Humanoid Robot Locomotion

Stefano Dafarra, *Member, IEEE*, Giulio Romualdi, *Graduate Student Member, IEEE*, and Daniele Pucci, *Associate Member, IEEE*

***Abstract*—This article presents a planner to generate walking trajectories by using the centroidal dynamics and the full kinematics of a humanoid robot. The interaction between the robot and the walking surface is modeled explicitly via new conditions, the *dynamic complementarity conditions*. The approach does not require a predefined contact sequence and generates the footsteps automatically. We characterize the robot control objective via a set of tasks, and we address it by solving an optimal control problem. We show that it is possible to achieve walking motions automatically by specifying a minimal set of references, such as a constant desired center of mass velocity and a reference point on the ground. Furthermore, we analyze how the contact modeling choices affect the computational time. We validate the approach by generating and testing walking trajectories for the humanoid robot iCub.**

***Index Terms*—Complementarity, contact modeling, humanoids, nonlinear optimization, whole-body trajectory generation.**

I. Introduction

THE general problem of planning whole-body locomotion trajectories of humanoid robots remains an interesting and partially open challenge for the robotics community [1]. Whole-body *humanoid robot motion planning*, in fact, is often associated with optimization problems having high-dimensional search space, which further complexifies the process of finding either local or global solutions. This process requires sophisticated models that characterize the robot motions and interactions with its surrounding environment. When one of the robot bodies makes contact with a *rigid* environment, the models that may describe the robot motions and contact forces are known as *complementarity conditions*, which add a degree of complexity when solving the associated whole-body trajectory optimization problem. A common approach to circumvent this barrier is to consider simplified, and often linear, robot models that seldom consider the contacts between the humanoid robot and its surroundings. This article presents a nonlinear trajectory optimization approach that employs novel definitions of the complementarity conditions tailored for *whole-body humanoid robot motion planning*, here called *dynamic complementarity conditions* (DCCs).

After the DARPA Robotics Challenge [2], it became popular to tackle humanoid robot locomotion using a hierarchical control architecture [3]. Each loop of the architecture receives inputs from the robot and the environment, and provides references to the loop next. The inner the layer, the shorter the time horizon that is used to evaluate the outputs. Also, inner loops usually employ more complex robot models to evaluate their outputs, but the shorter time horizon often results in faster computations for obtaining these outputs. More precisely, from outer to inner, the hierarchical control architecture is usually composed of the following loops: the *trajectory optimization*; the *simplified model control*; and the *whole-body quadratic programming (QP) control* loop [4], [5].

The *trajectory optimization* loop is often in charge of generating foothold trajectories starting from high-level commands, such as those coming from the user via a joypad. The output of this layer is given to the *simplified model control* loop—also called *receding horizon* [6]. Its aim is to generate desired and feasible *centroidal* quantities associated with *stable* walking instances [7]. The output of this loop is fed into the *whole-body QP control* layer, which is in charge of stabilizing the planned trajectories exploiting the full robot model with a suitable *QP* formulation. The *trajectory optimization* and the *simplified model control* loops aim at generating robot trajectories, so can be viewed as the *planning* layers of the architecture. The present article proposes novel methods for combining these two layers. What follows is a short recap of the state-of-the-art concerning the planning algorithms employed in the *trajectory optimization* and the *simplified model control* loops.

The *trajectory optimization* layer mainly deals with defining the contact locations of the robot locomotion pattern.

Manuscript received 30 September 2021; revised 28 February 2022; accepted 6 May 2022. Date of publication 2 August 2022; date of current version 6 December 2022. This work was supported by the Italian National Institute for Insurance against Accidents at Work (INAIL) ergoCub Project. This article was recommended for publication by Associate Editor O. Stasse and Editor E. Yoshida upon evaluation of the reviewers' comments. *(Corresponding author: Stefano Dafarra.)*

Stefano Dafarra is with the Artificial and Mechanical Intelligence, Italian Institute of Technology, 16163 Genoa, Italy (e-mail: stefano.dafarra@iit.it).

Giulio Romualdi is with the Artificial and Mechanical Intelligence, Italian Institute of Technology, 16163 Genoa, Italy, and also with the DIBRIS, University of Genoa, 16145 Genoa, Italy (e-mail: giulio.romualdi@iit.it).

Daniele Pucci is with the Artificial and Mechanical Intelligence, Italian Institute of Technology, 16163 Genoa, Italy, and also with the Machine Learning and Optimisation, University of Manchester, M13 9PL Manchester, U.K. (e-mail: daniele.pucci@iit.it).

This article has supplementary material provided by the authors and color versions of one or more figures available at https://doi.org/10.1109/TRO.2022.3183785.

Digital Object Identifier 10.1109/TRO.2022.3183785

Several strategies exist to address this problem, and they can be categorized depending on the amount of hand-crafted contact information an external user needs to specify to run the planning algorithms. *Fixed contact sequence, timing, and location* planning assumes that contact positions and timings are either specified by an external user or by an outer control loop. Planning then focuses on the center of mass (CoM) state [8]. In this case, the main complexity is given by the nonlinearity of the angular momentum dynamics. This problem can be faced by considering only the CoM linear acceleration [9], or by using a convex upper bound of the angular momentum to be minimized [10]. A convex optimization problem is obtained also by forcing the CoM trajectory to a polynomial with only one free variable [11]. Body postures can be planned together with the centroidal quantities, thus considering the robot kinematic structure in the same formulation [12]–[16]. These methods, however, require external contact planners. Analogously, *differential dynamic programming*-based methods can generate whole-body motions [17]–[20], and track them online [21]. *Predefined contact sequence* planning assumes to know in advance the contact sequence, although the contact locations and timings are not fixed [22]–[24]. Different sets of equations for each contact phase usually model the hybrid nature arising from the pattern of making and breaking contacts. To solve the associated problem, one may use *mixed integer programming* that uses integer variables in the definition of the optimization framework to determine where and at which time a contact should be made [25]–[29]. Exploiting integer variables, however, strongly affects the computational performances, especially when the robot may make several contacts with the environment. *Complementarity-free* planning approaches model multibody systems subject to contacts without directly enforcing the complementarity conditions [30]. Equivalently accurate results are obtained by maximizing the rate of energy dissipation [31]. These methods consider contacts explicitly in the planner formulation, so contact location, timing, and sequence are obtained as solutions to the optimization problem, thus generating complex robot motions [32]–[34].

The other planning layer, the *simplified model control* layer, is in charge of finding feasible trajectories for the robot CoM along the walking path. The computational burden to find feasibility regions, however, usually calls for simplified models to characterize the robot dynamics. Indeed, when restricting the CoM on a plane at a fixed height and assuming constant angular momentum, it is possible to derive simple and effective control laws based on the linear inverted pendulum (LIP) model [35], [36], the capture point [37], and the *divergent component of motion* (DCM) [38]. These simplified linear models have allowed online model predictive control [39]–[43], also providing references for the footstep locations.

An emerging approach is that of combining the *simplified model control* into the *trajectory optimization* layer so as to obtain both foothold trajectories and feasible centroidal quantities with a single optimization problem, which often employs complete or reduced robot models. When complete robot models are combined with impact dynamics, the control problem increases its complexity considerably.

A. Related Works

Neunert *et al.* [44] presented one of the first real-time implementation of an MPC controller adopting the full robot model with contacts. This has been tested on a quadruped robot adopting a tailored soft contact model. Nevertheless, the use of soft contact models presents some limitations when applied to robots with finite size feet [45].

Instead of considering the entire robot model in a single optimal control problem, another possibility consists in planning trajectories for the centroidal and joint quantities separately, iterating until a consensus is reached between the two sets of trajectories [12]. The footstep locations and timings can also be determined while planning for the robot momentum [46].

Dai *et al.* [47] exploit the full robot kinematics and its centroidal dynamics in a single optimal control problem. They follow the approach proposed by Posa *et al.* [48] on the consecutive relaxations of the complementarity conditions, so they avoid to consider soft contact models. Contact sequences can be predefined or defined by the optimizer.

B. Contribution

This article proposes new methods to consider the *simplified model control* objectives as part of the *trajectory optimization* layer. So, feasible footsteps and whole-body robot trajectories are obtained as the solution to a single nonlinear optimization problem. With our approach, neither contact sequences nor contact locations and timings need to be known beforehand, and no hand-crafted *a priori* knowledge is injected in the system to generate walking motions.

The approach we propose follows similar arguments and methods presented in Dai *et al.* [47]. In particular, we take the same modeling approach that consists in considering the full robot kinematics and centroidal momentum. On the other hand, we introduce a new and alternative formulation of the classical complementarity conditions tailored for trajectory optimization problems aimed at finding robot locomotion patterns. These are called *dynamic complementarity conditions (DCCs)* and represent the first contribution of the article.

As a second contribution, we show how the DCCs can be embedded into a whole-body nonlinear optimization framework to find feasible footsteps and whole-body robot trajectories with no *a priori* knowledge about the robot locomotion. This represents another difference compared to the work by Dai *et al.* [47], where a postural term in the cost function determines the robot motion, and the contacts are predefined in the majority of the cases.

We compute the robot trajectories using the *receding horizon* principle. We validate the whole approach using the real iCub humanoid robot. Finally, we perform a comparison between the DCCs and other classical complementarity formulations to show that DCCs may lead to improved computational time and contact condition accuracy. This represents the third contribution of the article.

This article extends and encompasses the authors previous work [49]. The main differences between the two articles are as follows. 1) The previously introduced contact parametrization [49] is presented as one of the DCCs and we compare its performances with other methods. The comparison is performed in two contexts and with a crescendo of complexity. First, we analyze the performances of the complementarity methods when applied to a simple toy problem. Then, we use the whole-body nonlinear optimization framework presented in the article as a testing ground. The comparisons run both on a personal laptop, and by using `Github Actions`, thus exploiting a reproducible setup. 2) We validate the generated trajectories on the real iCub Humanoid robot. 3) We update the formulation of the whole-body nonlinear optimization framework adding more tasks that lead to a better generation of walking trajectories.

The rest of this article is organized as follows. Section II presents some basic humanoid robot modeling tools and introduces to the complementarity conditions of rigid contact models. Section III presents the DCCs. These are then adopted in a nonlinear trajectory optimization framework, described in Section IV. The output of the planner is validated and tested on the real humanoid robot iCub in Section V, while we draw comparisons between the DCCs and state-of-the-art methods in Section VI. Finally, Section VII concludes this article.

II. Background

A. Notation

Throughout the article, we use the following notation.

1) Vector and matrices are expressed in bold symbols.
2) The ith component of a vector $\boldsymbol{x}$ is denoted as x_i.
3) The transpose operator is denoted by $(\cdot)^\top$.
4) The superscript $(\cdot)^*$ indicates desired values.
5) Given a function of time $f(t)$, the dot notation denotes its time derivative, i.e., $\dot{f} := \frac{\mathrm{d}f}{\mathrm{d}t}$. Higher order derivatives are denoted with a corresponding amount of dots.
6) $\mathcal{I}$ is a fixed inertial frame with respect to (w.r.t.) which the robot's absolute pose is measured. Its z axis is supposed to point against gravity, while the x direction defines the forward direction.
7) $\mathbb{1}_n \in \mathbb{R}^{n\times n}$ denotes the identity matrix of dimension n.
8) $\mathbf{0}_{n\times n} \in \mathbb{R}^{n\times n}$ denotes a zero matrix, while $\mathbf{0}_n = \mathbf{0}_{n\times 1}$ is a zero column vector of size n.
9) $\boldsymbol{e}_i$ is the canonical base in $\mathbb{R}^n$. It corresponds to $\boldsymbol{e}_i = [0, 0, \ldots, 1, 0, \ldots, 0]^\top \in \mathbb{R}^n$, where the only unitary element is in the ith position.
10) The operator $\wedge$ defines the skew-symmetric operation associated with the cross-product in $\mathbb{R}^3$. For example, $(a^\wedge)b$, with $a, b \in \mathbb{R}^3$, is equivalent to $a \times b$. Its inverse is the operator vee $\vee$, i.e., $(a^\wedge)^\vee = a$.
11) Given $\boldsymbol{x} \in \mathbb{R}^n$, and $\alpha \in \mathbb{R}$, $\boldsymbol{x} \succcurlyeq \alpha$ indicates that each component of $\boldsymbol{x}$ is greater or equal than α, i.e., $x_i \geq \alpha \,\forall i \in [1, n]$.
12) The weighted L2-norm of a vector $\boldsymbol{v} \in \mathbb{R}^n$ is denoted by $\|\boldsymbol{v}\|_{\boldsymbol{W}}$, $\boldsymbol{W} \in \mathbb{R}^{n\times n}$ is a positive definite weight matrix.
13) ${}^A\boldsymbol{R}_B \in SO(3)$ and ${}^A\boldsymbol{H}_B \in SE(3)$ denote the rotation and transformation matrices which transform a vector expressed in the B frame, ${}^B\boldsymbol{x}$, into a vector expressed in the A frame, ${}^A\boldsymbol{x}$.
14) ${}^D\boldsymbol{V}_{A,D} \in \mathbb{R}^6$ is the relative velocity between frame A and D, and its coordinates are expressed in frame D. The first three coordinates refer to the linear part of the velocity, while the bottom three to the angular velocity.
15) $\boldsymbol{x} \in \mathbb{R}^3$ is the position of the center of mass w.r.t. $\mathcal{I}$.
16) $\mathbf{n}(\cdot) : \mathbb{R}^3 \to \mathbb{R}^3$ is a function returning the direction that is perpendicular to the walking plane given the x and y coordinates of the input point.
17) $\mathbf{t}(\cdot) : \mathbb{R}^3 \to \mathbb{R}^{3\times 2}$ is a function returning two perpendicular directions normal to $\mathbf{n}(\cdot)$. The composition of $\mathbf{t}(\cdot)$ and $\mathbf{n}(\cdot)$, i.e., $[\mathbf{t}(\cdot)\, \mathbf{n}(\cdot)]$, defines the rotation matrix ${}^\mathcal{I}\boldsymbol{R}_{plane}$.
18) $h(\boldsymbol{p}) : \mathbb{R}^3 \to \mathbb{R}$ defines the distance between $\boldsymbol{p}$ and the walking surface.
19) $\mathrm{diag}(\cdot) : \mathbb{R}^n \to \mathbb{R}^{n\times n}$ is a function casting the argument into the corresponding diagonal matrix.

B. Humanoid Robot Modeling

The robot configuration space is characterized by the *position* and the *orientation* of the base frame B, and the joint configurations. Thus, it corresponds to the group $\mathbb{Q} = \mathbb{R}^3 \times SO(3) \times \mathbb{R}^n$ and an element $\boldsymbol{q} \in \mathbb{Q}$ can be defined as the following triplet: $\boldsymbol{q} = ({}^\mathcal{I}\boldsymbol{p}_B, {}^\mathcal{I}\boldsymbol{R}_B, \boldsymbol{s})$.

The *velocity* of the multibody system can be characterized by the *algebra* $\mathbb{V}$ of $\mathbb{Q}$ defined by $\mathbb{V} = \mathbb{R}^3 \times \mathbb{R}^3 \times \mathbb{R}^n$. An element of $\mathbb{V}$ corresponds to $\boldsymbol{\nu}$.

We also assume that the robot is interacting with the environment exchanging n_c distinct wrenches.[1] By employing the Euler–Poincaré formalism [50, Ch. 13.5], we obtain

$$\boldsymbol{M}(\boldsymbol{q})\dot{\boldsymbol{\nu}} + \boldsymbol{C}(\boldsymbol{q}, \boldsymbol{\nu})\boldsymbol{\nu} + \boldsymbol{G}(\boldsymbol{q}) = \begin{bmatrix} \mathbf{0}_{6\times n} \\ \mathbb{1}_n \end{bmatrix} \boldsymbol{\tau}_s + \sum_{k=1}^{n_c} \boldsymbol{J}^\top_{\mathcal{C}_k}\, {}_k\mathbf{f}, \tag{1}$$

where $\boldsymbol{M} \in \mathbb{R}^{(n+6)\times(n+6)}$ is the mass matrix, $\boldsymbol{C} \in \mathbb{R}^{(n+6)\times(n+6)}$ accounts for Coriolis and centrifugal effects, $\boldsymbol{G} \in \mathbb{R}^{n+6}$ is the gravity term. The vector $\boldsymbol{\tau}_s \in \mathbb{R}^n$ represents the actuation torques, while ${}_k\mathbf{f} \in \mathbb{R}^6$ denotes the kth external wrench applied by the environment on the robot, namely

$${}_k\mathbf{f} = \begin{bmatrix} {}_k\boldsymbol{f} \\ {}_k\boldsymbol{\tau} \end{bmatrix} \tag{2}$$

with ${}_k\boldsymbol{f}$, ${}_k\boldsymbol{\tau} \in \mathbb{R}^3$ being the contact force and torque, respectively. We assume the wrenches to be measured in a frame located on the contact link origin and oriented as $\mathcal{I}$.

As described in [51, Section 5], it is possible to apply a coordinate transformation in the state-space (q, ν) that transforms the system dynamics (1) into a new form where the mass matrix is block diagonal, thus decoupling joint and base frame accelerations. Also, in this new set of coordinates, the first six rows of (1) correspond to the *centroidal dynamics*. In the specialized literature, this term is used to indicate the rate of change of the robot's momentum expressed at the CoM, which then equals the summation of all external wrenches acting on the multibody system [52], [53].

[1] As an abuse of notation, we define as *wrench* a quantity that is not the dual of a *twist*, but a 6-D force/moment vector.

C. Centroidal Dynamics

By definition, the CoM $\boldsymbol{x} \in \mathbb{R}^3$ corresponds to the weighted average of all the links' CoM position

$$\boldsymbol{x} = {}^{\mathcal{I}}\boldsymbol{H}_B \sum_i {}^B\boldsymbol{H}_i\, {}^i\mathbf{p}_{\text{CoM}} \frac{m_i}{m} \tag{3}$$

where ${}^i\mathbf{p}_{\text{CoM}} \in \mathbb{R}^3$ is the (constant) CoM position of the ith link expressed w.r.t. the ith coordinate systems. The scalars $m, m_i \in \mathbb{R}^+$ are the robot total mass and the ith link mass, respectively. In order to introduce the *centroidal dynamics*, it is convenient to define a frame, called $\bar{G}$, whose origin is located on the CoM, while the orientation is parallel to the inertial frame $\mathcal{I}$. We introduce ${}_{\bar{G}}\boldsymbol{h} \in \mathbb{R}^6$ to denote the robot total momentum expressed w.r.t. $\bar{G}$, namely

$${}_{\bar{G}}\boldsymbol{h} = \begin{bmatrix} {}_{\bar{G}}\boldsymbol{h}^p \\ {}_{\bar{G}}\boldsymbol{h}^\omega \end{bmatrix} \tag{4}$$

where ${}_{\bar{G}}\boldsymbol{h}^p \in \mathbb{R}^3$ and ${}_{\bar{G}}\boldsymbol{h}^\omega \in \mathbb{R}^3$ are the linear and angular momentum, respectively. In addition, the following holds:

$$\dot{\boldsymbol{x}}_{\text{CoM}} = \frac{1}{m}{}_{\bar{G}}\boldsymbol{h}^p. \tag{5}$$

The robot total momentum corresponds to the summation of all the links momenta ${}_B\boldsymbol{h}^i$, measured in the base frame and projected on $\bar{G}$

$${}_{\bar{G}}\boldsymbol{h} = \sum_i {}_{\bar{G}}\boldsymbol{X}^B\, {}_B\boldsymbol{h}^i. \tag{6}$$

The *adjoint matrix* ${}_{\bar{G}}\boldsymbol{X}^B$ transforms a wrench expressed in B into one expressed in $\bar{G}$. Then, ${}_{\bar{G}}\boldsymbol{h}$ writes

$${}_{\bar{G}}\boldsymbol{h} = {}_{\bar{G}}\boldsymbol{X}^B \sum_i {}_B\boldsymbol{X}^i \boldsymbol{I}_i\, {}^i\boldsymbol{V}_{A,i} \tag{7}$$

with $\boldsymbol{I}_i \in \mathbb{R}^{6\times 6}$ being the (constant) link inertia expressed in link frame. Hence, one obtains

$${}_{\bar{G}}\boldsymbol{h} = \boldsymbol{J}_{\text{CMM}}\boldsymbol{\nu} \tag{8}$$

where $\boldsymbol{J}_{\text{CMM}} \in \mathbb{R}^{6\times n}$ is the *centroidal momentum matrix* (CMM) [7]. The rate of change of the centroidal momentum balances the external wrenches applied to the robot, which yields

$$\begin{aligned} {}_{\bar{G}}\dot{\boldsymbol{h}} &= \sum_{k=1}^{n_c} {}_{\bar{G}}\boldsymbol{X}^k\, {}_k\mathbf{f} + m\bar{\boldsymbol{g}} \\ &= \sum_{k=1}^{n_c} \begin{bmatrix} {}^{\mathcal{I}}\boldsymbol{R}_k & \mathbf{0}_{3\times 3} \\ ({}^{\mathcal{I}}\boldsymbol{o}_k - \boldsymbol{x})^\wedge\, {}^{\mathcal{I}}\boldsymbol{R}_k & {}^{\mathcal{I}}\boldsymbol{R}_k \end{bmatrix} {}_k\mathbf{f} + m\bar{\boldsymbol{g}}. \end{aligned} \tag{9}$$

The adjoint matrix ${}_{\bar{G}}\boldsymbol{X}^k \in \mathbb{R}^{6\times 6}$ transforms the contact wrench from the application frame (located in ${}^{\mathcal{I}}\boldsymbol{o}_k$ with orientation ${}^{\mathcal{I}}\boldsymbol{R}_k$) to $\bar{G}$. Finally, $\bar{\boldsymbol{g}} = [0\ 0\ -g\ 0\ 0\ 0]^\top$ is the 6-D gravity acceleration vector.

Alternatively, the centroidal momentum dynamics can be obtained by differentiating (8), which writes

$${}_{\bar{G}}\dot{\boldsymbol{h}} = \boldsymbol{J}_{\text{CMM}}\dot{\boldsymbol{\nu}} + \dot{\boldsymbol{J}}_{\text{CMM}}\boldsymbol{\nu} \tag{10}$$

thus highlighting the dependency of ${}_{\bar{G}}\dot{\boldsymbol{h}}$ on $\dot{\boldsymbol{\nu}}$.

D. Recall on Complementarity Conditions

Humanoid robots are often equipped with flat feet, which simplify the walking and balancing task with respect to line and point foot. It is important, then, to model how the foot can land on the walking surface, also limiting the set of admissible contact wrenches. As an example, in case of line contacts, no torque can be exerted along the contact line.

A common approach is to consider the foot as composed by a set of points, for example (but not limited to), four points located at the corners of the foot [23], [47], [54]. A pure force is supposed to be applied on each of the contact points.

Define ${}_i\boldsymbol{p} \in \mathbb{R}^3$ as the ith contact point location in an inertial frame $\mathcal{I}$, and ${}_i\boldsymbol{f} \in \mathbb{R}^3$ as the force exerted on that point. Such force is expressed on a frame located in ${}_i\boldsymbol{p}$ and with orientation parallel to $\mathcal{I}$.

In this article, we exploit a *rigid contact model* to characterize the interactions between the robot feet and the environment floor. In the literature, other contact models exist, e.g., *compliant models* that allow a degree of compenetration [44], [55], [56]. In a rigid setting, instead, the contact points are not supposed to penetrate the walking ground, i.e.,

$$h({}_i\boldsymbol{p}) \geq 0.$$

The force ${}_i\boldsymbol{f}$ results from the interaction between the contact point and the ground, and hence, it is bounded. Being a reaction force, its normal component with respect to the walking ground is supposed to be non-negative. In particular,

$$\mathbf{n}({}_i\boldsymbol{p})^\top {}_i\boldsymbol{f} \geq 0. \tag{11}$$

Additionally, we focus on the case where the contact force is not enough to overcome the static friction

$$\|\mathbf{t}({}_i\boldsymbol{p})^\top {}_i\boldsymbol{f}\| \leq \mu_s\, \mathbf{n}({}_i\boldsymbol{p})^\top {}_i\boldsymbol{f} \tag{12}$$

where $\mu_s \in \mathbb{R}^+$ is the static friction coefficient. It can be shown that, for what concern the validity of the contact wrench, it is sufficient to guarantee that the forces at the vertices satisfy (11) and (12) [57].

Being a reaction force, the value of ${}_i\boldsymbol{f}$ is different from zero only if the associated contact point is in contact with the walking surface. This condition can be compactly written as

$$h({}_i\boldsymbol{p})\, \mathbf{n}({}_i\boldsymbol{p})^\top {}_i\boldsymbol{f} = 0. \tag{13}$$

Such a constraint may lead to a number of numerical issues when solving an optimization problem that makes use of it. This is due to the fact that the feasible set is only defined by two lines, namely $h({}_i\boldsymbol{p}) = 0$ and $\mathbf{n}({}_i\boldsymbol{p})^\top {}_i\boldsymbol{f} = 0$, that intersect the origin. In particular, at this point, the so-called *constraint Jacobian* is singular, thus violating the *linear independence constraint qualification*, on which most off-the-shelf solvers rely [58].

The conditions given by (13) can be *relaxed* considering that both $h({}_i\boldsymbol{p})$ and $\mathbf{n}({}_i\boldsymbol{p})^\top {}_i\boldsymbol{f}$ are positive quantities. Hence, instead of using an equality condition, it is possible to upper bound their product with a small positive constant $\epsilon \in \mathbb{R}^+$

$$h({}_i\boldsymbol{p})\, \mathbf{n}({}_i\boldsymbol{p})^\top {}_i\boldsymbol{f} \leq \epsilon. \tag{14}$$

Inequality (14) is referred to as *relaxed complementarity*. Thanks to this relaxation, the feasibility region increases in its dimension. This common approach also corresponds to using "bounded" slack variables [48], [58]–[60].

Once a contact is made, the relaxed complementarity condition cannot prevent the contact point to move on the walking plane: Even if the friction constraints defined by (12) are satisfied, the contact points are still free to move on the walking surface. This problem is often faced by imposing that the product between the planar velocities and tangential forces is equal to zero [47], [48], the relaxed version of which writes

$$-\boldsymbol{\epsilon}_p \leq \operatorname{diag}\left(\mathbf{t}\left({}_i\boldsymbol{p}\right)^\top {}_i\dot{\boldsymbol{p}}\right) \mathbf{t}\left({}_i\boldsymbol{p}\right)^\top {}_i\boldsymbol{f} \leq \boldsymbol{\epsilon}_p \tag{15}$$

with $\boldsymbol{\epsilon}_p \in \mathbb{R}^2, \boldsymbol{\epsilon}_p \succcurlyeq 0$ the relaxation parameters. The relations (15) are referred to as *relaxed planar complementarity*.

III. Dynamic Complementarity Conditions

The *relaxed complementarity* (14) trades off the numerical tractability of the problem with the accuracy of the model. This is particularly important when (14) is integrated in optimal control problems for *small* humanoid robots, whose feet little rise up from ground during the swing phases of walking motions. The normal force allowed by the *relaxed complementarity* (14) may have a non-negligible effect on the overall robot *stability* of these phases.

This section proposes two novel approaches to describe the complementarity conditions of a rigid contact model, namely

1) dynamically enforced complementarity; and
2) hyperbolic secant in velocity bounds.

We call them dynamic complementarity conditions (DCCs).

A. Dynamically Enforced Complementarity

The complementarity constraints can be enforced using a Baumgarte stabilization method [61]. Thus, we enforce conditions (13) so as to obtain convergence to zero. This is achieved by setting the following *dynamical* constraint:

$$\frac{\mathrm{d}}{\mathrm{d}t}\left(h({}_i\boldsymbol{p})\,\mathbf{n}({}_i\boldsymbol{p})^\top {}_i\boldsymbol{f}\right) = -K_{\text{bs}}\left(h({}_i\boldsymbol{p})\,\mathbf{n}({}_i\boldsymbol{p})^\top {}_i\boldsymbol{f}\right) \tag{16}$$

where $K_{\text{bs}} \in \mathbb{R}^+$ is a positive gain. Hence, the product $h({}_i\boldsymbol{p})\,\mathbf{n}({}_i\boldsymbol{p})^\top {}_i\boldsymbol{f}$ exponentially decreases to zero at a rate dependent on K_{bs}. Direct calculations then lead to

$$\frac{\mathrm{d}}{\mathrm{d}t}(\cdot) = \frac{\mathrm{d}}{\mathrm{d}t}\left(h({}_i\boldsymbol{p})\right)\mathbf{n}({}_i\boldsymbol{p})^\top {}_i\boldsymbol{f} + \\ + h({}_i\boldsymbol{p}) {}_i\boldsymbol{f}^\top \frac{\mathrm{d}}{\mathrm{d}t}\left(\mathbf{n}({}_i\boldsymbol{p})\right) + h({}_i\boldsymbol{p})\,\mathbf{n}({}_i\boldsymbol{p})^\top {}_i\dot{\boldsymbol{f}} \tag{17}$$

where we use the fact that the left-hand side of (13) is scalar, which eases the computations above. We can substitute the time derivative of the $h(\cdot)$ and $\mathbf{n}(\cdot)$ functions with the relations

$$\frac{\mathrm{d}}{\mathrm{d}t}\left(h({}_i\boldsymbol{p})\right) = \frac{\partial}{\partial_i\boldsymbol{p}}\left(h({}_i\boldsymbol{p})\right){}_i\dot{\boldsymbol{p}} \tag{18a}$$

$$\frac{\mathrm{d}}{\mathrm{d}t}\left(\mathbf{n}({}_i\boldsymbol{p})\right) = \frac{\partial}{\partial_i\boldsymbol{p}}\left(\mathbf{n}({}_i\boldsymbol{p})\right){}_i\dot{\boldsymbol{p}}. \tag{18b}$$

For simplicity, let us define ζ as follows:

$$\zeta := \frac{\partial}{\partial_i\boldsymbol{p}}\left(h({}_i\boldsymbol{p})\right){}_i\dot{\boldsymbol{p}}\,\mathbf{n}({}_i\boldsymbol{p})^\top {}_i\boldsymbol{f} \\ + h({}_i\boldsymbol{p}) {}_i\boldsymbol{f}^\top \frac{\partial}{\partial_i\boldsymbol{p}}\left(\mathbf{n}({}_i\boldsymbol{p})\right){}_i\dot{\boldsymbol{p}} + h({}_i\boldsymbol{p})\,\mathbf{n}({}_i\boldsymbol{p})^\top {}_i\dot{\boldsymbol{f}}.$$

Finally, we obtain the condition

$$\zeta = -K_{\text{bs}}\left(h({}_i\boldsymbol{p})\,\mathbf{n}({}_i\boldsymbol{p})^\top {}_i\boldsymbol{f}\right). \tag{19}$$

In case of planar ground, we have the following relations:

$$h({}_i\boldsymbol{p}) = \boldsymbol{e}_3^\top {}_i\boldsymbol{p},\ \frac{\partial}{\partial_i\boldsymbol{p}}\left(h({}_i\boldsymbol{p})\right) = \boldsymbol{e}_3^\top \tag{20a}$$

$$\mathbf{n}({}_i\boldsymbol{p}) = \boldsymbol{e}_3,\ \frac{\partial}{\partial_i\boldsymbol{p}}\left(\mathbf{n}({}_i\boldsymbol{p})\right) = \mathbf{0}_{3\times3}. \tag{20b}$$

In other words, the normal vector to the walking plane corresponds to $\boldsymbol{e}_3$, while the height $h({}_i\boldsymbol{p})$ of the point coincides with the point z-coordinate. Hence, in the planar case, ζ writes

$$\zeta_{\text{planar}} = {}_i\dot{\boldsymbol{p}}_z \cdot {}_i\boldsymbol{f}_z + {}_i\boldsymbol{p}_z \cdot {}_i\dot{\boldsymbol{f}}_z. \tag{21}$$

We can relax (19) by exploiting again the fact that the product $h({}_i\boldsymbol{p})\,\mathbf{n}({}_i\boldsymbol{p})^\top {}_i\boldsymbol{f}$ is positive by construction. Henceforth, if we impose the following:

$$\zeta \leq -K_{\text{bs}}\left(h({}_i\boldsymbol{p})\,\mathbf{n}({}_i\boldsymbol{p})^\top {}_i\boldsymbol{f}\right) \tag{22}$$

we still have exponential convergence, at a rate which is higher or equal to the one specified by K_{bs}. Finally, similarly to (14), we add a further relaxation through a positive number $\varepsilon \in \mathbb{R}^+$ to increase the feasibility region

$$\zeta \leq -K_{\text{bs}}\left(h({}_i\boldsymbol{p})\,\mathbf{n}({}_i\boldsymbol{p})^\top {}_i\boldsymbol{f}\right) + \varepsilon \tag{23}$$

obtaining the final version of the complementarity condition.

B. Hyperbolic Secant in Velocity Bounds

We can impose (13) dynamically by enforcing the following set of constraints on the force derivative:

$$-\boldsymbol{M}_f \leq {}_i\dot{\boldsymbol{f}} \leq \boldsymbol{M}_f \quad \text{if } h({}_i\boldsymbol{p}) = 0 \tag{24a}$$

$${}_i\dot{\boldsymbol{f}} = -\boldsymbol{K}_f {}_i\boldsymbol{f} \quad \text{if } h({}_i\boldsymbol{p}) \neq 0 \tag{24b}$$

meaning that when the point is in contact, ${}_i\dot{\boldsymbol{f}}$ is free to take any value in $[-\boldsymbol{M}_f, \boldsymbol{M}_f]$ with $\boldsymbol{M}_f \in \mathbb{R}^3$ a (non-negative) control bound. On the other hand, if the contact point is not on the walking surface, the control input makes the contact force decreasing exponentially, see (24b), at a rate depending on the positive definite control gain $\boldsymbol{K}_f \in \mathbb{R}^{3\times3}$. Defining $\delta^*({}_i\boldsymbol{p})$ as a binary function such that

$$\delta^*({}_i\boldsymbol{p}) = \begin{cases} 1 & \text{if } h({}_i\boldsymbol{p}) = 0 \\ 0 & h({}_i\boldsymbol{p}) \neq 0 \end{cases} \tag{25}$$

it is possible to write (24) as a set of two inequalities

$$-\boldsymbol{K}_f\left(1 - \delta^*({}_i\boldsymbol{p})\right){}_i\boldsymbol{f} - \delta^*({}_i\boldsymbol{p})\boldsymbol{M}_f \leq {}_i\dot{\boldsymbol{f}} \tag{26a}$$

$$-\boldsymbol{K}_f\left(1 - \delta^*({}_i\boldsymbol{p})\right){}_i\boldsymbol{f} + \delta^*({}_i\boldsymbol{p})\boldsymbol{M}_f \geq {}_i\dot{\boldsymbol{f}}. \tag{26b}$$

Even if $\delta^*({}_i\boldsymbol{p})$ would require the adoption of integer variables, it is possible to use a continuous approximation, $\delta({}_i\boldsymbol{p})$, namely the hyperbolic secant

$$\delta({}_i\boldsymbol{p}) = \operatorname{sech}\left(k_h\, h({}_i\boldsymbol{p})\right) \tag{27}$$

where k_h is a user-defined scaling factor. Notice that, when $\delta^*({}_i\boldsymbol{p}) = 0$, the bounds coincide and are equal to $-\boldsymbol{K}_{f\,i}\boldsymbol{f}$.

As discussed in Section III-A, we can simplify the lower bound defined by (26a) allowing the force to decrease at a higher rate than that given by (26b). Hence, we can rewrite (26) as

$$-\boldsymbol{M}_f \leq {}_i\dot{\boldsymbol{f}} \leq -\boldsymbol{K}_f\left(1 - \delta({}_i\boldsymbol{p})\right){}_i\boldsymbol{f} + \delta({}_i\boldsymbol{p})\boldsymbol{M}_f. \tag{28}$$

Given (12), it is enough to apply any of these equations only to the force component normal to the ground: If it decreases to zero, also planar components have to vanish to satisfy friction constraints. Hence, we can reduce (28) to

$$-\boldsymbol{e}_3^\top \boldsymbol{M}_f \leq \boldsymbol{e}_3^\top {}_i\dot{\boldsymbol{f}} \tag{29a}$$

$$-K_{f,z}\left(1 - \delta({}_i\boldsymbol{p})\right)\mathbf{n}({}_i\boldsymbol{p})^\top {}_i\boldsymbol{f} + \delta({}_i\boldsymbol{p})\boldsymbol{e}_3^\top \boldsymbol{M}_f \geq \boldsymbol{e}_3^\top {}_i\dot{\boldsymbol{f}} \tag{29b}$$

with $K_{f,z}$ the corresponding element of $\boldsymbol{K}_f$.

C. Summary on Complementarity Conditions

We have presented different methods for enforcing the complementarity constraints given by (13), namely:

1) $h({}_i\boldsymbol{p})\,\mathbf{n}({}_i\boldsymbol{p})^\top {}_i\boldsymbol{f} \leq \epsilon$;
2) $\zeta \leq -K_{\text{bs}}(h({}_i\boldsymbol{p})\,\mathbf{n}({}_i\boldsymbol{p})^\top {}_i\boldsymbol{f}) + \varepsilon$;
3) $-\boldsymbol{e}_3^\top \boldsymbol{M}_f \leq \boldsymbol{e}_3^\top {}_i\dot{\boldsymbol{f}} \leq -K_{f,z}(1 - \delta({}_i\boldsymbol{p}))\mathbf{n}({}_i\boldsymbol{p})^\top {}_i\boldsymbol{f} + \delta({}_i\boldsymbol{p})\boldsymbol{e}_3^\top \boldsymbol{M}_f$.

It is also pragmatic to assume the force derivative to be bounded,[2] i.e., $-\boldsymbol{M}_f \leq {}_i\dot{\boldsymbol{f}} \leq \boldsymbol{M}_f$. Observe that all the conditions presented in the above bullet points do not depend on the type of ground, which is here considered rigid. The parameters involved do not have a direct physical meaning (like in compliant contact models), but rather determine the "accuracy" of the simulated behavior.

IV. DCCs-Based Nonlinear Trajectory Planning

We detail below how the *DCCs* presented in Section III can be used in a nonlinear trajectory planning framework aimed at generating walking trajectories.

A. Kinematic Control

First of all, we assume to have full control over the derivative of the contact point's positions and forces

$${}_i\dot{\boldsymbol{p}} = \boldsymbol{u}_{ip} \tag{30a}$$

$${}_i\dot{\boldsymbol{f}} = \boldsymbol{u}_{if} \tag{30b}$$

where $\boldsymbol{u}_{ip},\ \boldsymbol{u}_{if} \in \mathbb{R}^3$ are the control inputs for the ith contact point. While each contact point is supposed to be independent from the control point of view, they all need to remain on the same surface and maintain a constant relative distance: The contact points belong to the same rigid body. At the same time, we want them to be within the workspace reachable by the robot legs. We can achieve both objectives with the following algebraic condition acting on each contact point:

$${}_i\boldsymbol{p} = {}^{\mathcal{I}}\boldsymbol{H}_{\text{foot}}\, {}^{\text{foot}}{}_i\boldsymbol{p} \tag{31}$$

where ${}^{\text{foot}}{}_i\boldsymbol{p}$ is the (fixed) position of the contact point within the foot surface, expressed in foot coordinates. Here, the transformation matrix ${}^{\mathcal{I}}\boldsymbol{H}_{\text{foot}}$ would depend on $\boldsymbol{q}$, including the joint configuration $\boldsymbol{s}$. As a consequence, the full kinematics of the robot is taken into account.

Concerning the robot controls, we assume that the joint values can be assigned at will. In the language of automatic control and thanks to the so-called backstepping hypothesis [62, p. 589], full joint controllability is essentially equivalent to assuming the joint velocities as a control input, namely

$$\dot{\boldsymbol{s}} = \boldsymbol{u}_s. \tag{32}$$

This assumption requires an additional control loop: Joint velocities, together with contact vertex positions and forces, are considered as references to an inner whole-body controller.

Finally, the base rotation included in $\boldsymbol{q}$ is vectorized using the quaternion parametrization. The corresponding unitary quaternion is called ${}^{\mathcal{I}}\boldsymbol{\rho}_B \in \mathbb{H}$. The base position is indicated with the symbol ${}^{\mathcal{I}}\boldsymbol{p}_B \in \mathbb{R}^3$. The equations governing the dynamical evolution of the base are as follows:

$${}^{\mathcal{I}}\dot{\boldsymbol{p}}_B = {}^{\mathcal{I}}\boldsymbol{R}_B\, {}^B\boldsymbol{v}_{\mathcal{I},B} \tag{33a}$$

$${}^{\mathcal{I}}\dot{\boldsymbol{\rho}}_B = \boldsymbol{u}_\rho. \tag{33b}$$

${}^B\boldsymbol{v}_{\mathcal{I},B} \in \mathbb{R}^3$, $\boldsymbol{u}_\rho \in \mathbb{R}^4$, and $\boldsymbol{u}_s \in \mathbb{R}^n$ are control inputs, defining the base linear velocity, the quaternion derivative, and the joints velocity, respectively. More specifically, ${}^B\boldsymbol{v}_{\mathcal{I},B}$ is the linear part of ${}^B\boldsymbol{V}_{\mathcal{I},B} \in \mathbb{R}^6$, the *left-trivialized* (i.e., measured in body coordinates) base velocity. The control input $\boldsymbol{u}_\rho$ is such that the base quaternion remains unitary over time.

B. Planar DCC

As mentioned in Section II-D, we need to prevent the contact points to move on the walking plane when they are in contact with the ground. Instead of applying the *relaxed planar complementarity* (15), we achieve the same result by limiting the effect of the control input $\boldsymbol{u}_{ip}$ along the planar components

$$\mathbf{t}({}_i\boldsymbol{p})^\top {}_i\dot{\boldsymbol{p}} = \tanh\left(k_t\, h({}_i\boldsymbol{p})\right)\left[\boldsymbol{e}_1\, \boldsymbol{e}_2\right]^\top \boldsymbol{u}_{ip} \tag{34}$$

where $k_t \in \mathbb{R}$ is a user-defined scaling factor. Equation (34) projects the control input $\boldsymbol{u}_{ip}$ along the planar directions, and forces these projections to zero when $h({}_i\boldsymbol{p})$ is null. At the same time, (34) reduces the velocity when the contact point is approaching the ground. So, (34) can then be used to substitute (30a) as follows:

$${}_i\dot{\boldsymbol{p}} = \boldsymbol{\tau}({}_i\boldsymbol{p})\boldsymbol{u}_{ip} \tag{35}$$

[2] Note that this is not necessary if one uses (26b) instead of (29b) since the bounds are already included in the constraint.

where $\boldsymbol{\tau}({}_i\boldsymbol{p})$ is defined as

$$\boldsymbol{\tau}({}_i\boldsymbol{p}) = {}^{\mathcal{I}}\boldsymbol{R}_{plane}\, diag\left(\begin{bmatrix}\tanh\left(k_t\, h({}_i\boldsymbol{p})\right)\\ \tanh\left(k_t\, h({}_i\boldsymbol{p})\right)\\ 1\end{bmatrix}\right). \tag{36}$$

Note that, from now on, $\boldsymbol{u}_{ip}$ is assumed to be expressed in the *plane* coordinates. Thus, the normal component of the velocity is directly regulated by $\boldsymbol{e}_3^\top \boldsymbol{u}_{ip}$. Also, it is necessary to bound this control input, $\boldsymbol{u}_{ip} \in [-\boldsymbol{M}_V, \boldsymbol{M}_V]$, $\boldsymbol{M}_V \in \mathbb{R}^3$, to properly exploit the effect of the hyperbolic tangent.

Compared to the *relaxed planar complementarity* (15), the advantages provided by (35) are twofold. 1) The proposed DCC planar complementarity (35) naturally limits the point velocity to zero when approaching the contact surface. 2) Through the parameter k_t, it is possible to control the height of the contact points when swinging. The smaller the k_t, the higher is the resulting swing height.

C. Momentum Control

In Section II-D, we assumed that the contact points can exert any force on the environment. Let us now describe the effect of these forces on the robot motion through the centroidal dynamics introduced in Section II-C, which writes

$${}_{\bar{G}}\dot{\boldsymbol{h}} = m\bar{\boldsymbol{g}} + \sum_i \begin{bmatrix}\mathbb{1}_3\\ ({}_i\boldsymbol{p} - \boldsymbol{x})^\wedge\end{bmatrix} {}_i\boldsymbol{f}. \tag{37}$$

Compared to (9), (37) reduces the matrix ${}_{\bar{G}}\boldsymbol{X}^i$ since no torque is applied at the contact points. We also need to make sure that the CoM position obtained by integrating (5) corresponds to the one obtained via the joints variables. This is done through the following algebraic equation:

$$\boldsymbol{x} = \text{CoM}({}^{\mathcal{I}}\boldsymbol{p}_B, \mathcal{I}\boldsymbol{\rho}_B, s) \tag{38}$$

where $\text{CoM}({}^{\mathcal{I}}\boldsymbol{p}_B, \mathcal{I}\boldsymbol{\rho}_B, s)$ is the function mapping base pose and joints position to the CoM position, i.e., the right-hand side of (3). Equation (38) links the linear momentum and the joint variables, so we also need to link the angular momentum and the joints evolution. To do so, we use the CMM $\boldsymbol{J}_{\text{CMM}}$, thus obtaining

$${}_{\bar{G}}\boldsymbol{h}^\omega = [\boldsymbol{0}_{3\times3}\ \mathbb{1}_3]\, \boldsymbol{J}_{\text{CMM}}\boldsymbol{\nu} \in \mathbb{R}^3. \tag{39}$$

The system velocity $\boldsymbol{\nu}$ contains the base angular velocity ${}^B\boldsymbol{\omega}_{\mathcal{I},B}$. It can be substituted with the quaternion derivative through the map $\boldsymbol{\mathcal{G}}$. It is defined in [63, Section 1.5.4] as

$$\boldsymbol{\mathcal{G}}(\mathcal{I}\boldsymbol{\rho}_B) = \begin{bmatrix} -{}^{\mathcal{I}}\rho_{B,1} & {}^{\mathcal{I}}\rho_{B,0} & {}^{\mathcal{I}}\rho_{B,3} & -{}^{\mathcal{I}}\rho_{B,2}\\ -{}^{\mathcal{I}}\rho_{B,2} & -{}^{\mathcal{I}}\rho_{B,3} & {}^{\mathcal{I}}\rho_{B,0} & {}^{\mathcal{I}}\rho_{B,1}\\ -{}^{\mathcal{I}}\rho_{B,3} & {}^{\mathcal{I}}\rho_{B,2} & -{}^{\mathcal{I}}\rho_{B,1} & {}^{\mathcal{I}}\rho_{B,0}\end{bmatrix} \tag{40}$$

such that

$${}^B\boldsymbol{\omega}_{\mathcal{I},B} = 2\boldsymbol{\mathcal{G}}({}^{\mathcal{I}}\rho_B)\boldsymbol{u}_\rho. \tag{41}$$

Hence, it depends on the same control input of (33b).

D. Complete Differential-Algebraic System of Equations

By summarizing all the ODEs and algebraic conditions, we obtain the following inequality constrained differential-algebraic system of equations (DAE).

- Dynamical constraints

$${}_i\dot{\boldsymbol{f}} = \boldsymbol{u}_{if}, \qquad \forall \text{ contact point} \tag{42a}$$
$${}_i\dot{\boldsymbol{p}} = \boldsymbol{\tau}({}_i\boldsymbol{p})\boldsymbol{u}_{ip}, \qquad \forall \text{ contact point} \tag{42b}$$
$${}_{\bar{G}}\dot{\boldsymbol{h}} = m\bar{\boldsymbol{g}} + \sum_i \begin{bmatrix}\mathbb{1}_3\\ ({}_i\boldsymbol{p} - \boldsymbol{x})^\wedge\end{bmatrix} {}_i\boldsymbol{f} \tag{42c}$$
$$\dot{\boldsymbol{x}} = \frac{1}{m}\left({}_{\bar{G}}\boldsymbol{h}^p\right) \tag{42d}$$
$${}^{\mathcal{I}}\dot{\boldsymbol{p}}_B = {}^{\mathcal{I}}\boldsymbol{R}_B\,{}^B\boldsymbol{v}_{\mathcal{I},B} \tag{42e}$$
$${}^{\mathcal{I}}\dot{\boldsymbol{\rho}}_B = \boldsymbol{u}_\rho \tag{42f}$$
$$\dot{\boldsymbol{s}} = \boldsymbol{u}_s. \tag{42g}$$

- Equality constraints

$${}_i\boldsymbol{p} = {}^A\boldsymbol{H}_{\text{foot}}\,{}^{\text{foot}}{}_i\boldsymbol{p}, \quad \forall \text{ contact point} \tag{43a}$$
$$\boldsymbol{x} = \text{CoM}({}^{\mathcal{I}}\boldsymbol{p}_B, {}^{\mathcal{I}}\boldsymbol{\rho}_B, \boldsymbol{s}) \tag{43b}$$
$${}_{\bar{G}}\boldsymbol{h}^\omega = [\boldsymbol{0}_{3\times3}\ \mathbb{1}_3]\, \boldsymbol{J}_{\text{CMM}} \begin{bmatrix}{}^B\boldsymbol{v}_{\mathcal{I},B}\\ 2\boldsymbol{\mathcal{G}}({}^{\mathcal{I}}\boldsymbol{\rho}_B)\boldsymbol{u}_\rho\\ \boldsymbol{u}_s\end{bmatrix} \tag{43c}$$
$$\|{}^{\mathcal{I}}\boldsymbol{\rho}_B\|^2 = 1. \tag{43d}$$

- Inequality constraints, applied for each contact point

$$\mathbf{n}({}_i\boldsymbol{p})^\top {}_i\boldsymbol{f} \geq 0 \tag{44a}$$
$$\|\mathbf{t}({}_i\boldsymbol{p})^\top {}_i\boldsymbol{f}\| \leq \mu_s\, \mathbf{n}({}_i\boldsymbol{p})^\top {}_i\boldsymbol{f} \tag{44b}$$
$$-\boldsymbol{M}_V \leq \boldsymbol{u}_{ip} \leq \boldsymbol{M}_V \tag{44c}$$
$$-\boldsymbol{M}_f \leq \boldsymbol{u}_{if} \leq \boldsymbol{M}_f \tag{44d}$$
$$h({}_i\boldsymbol{p}) \geq 0 \tag{44e}$$
$$\text{Complementarity, see Section III-C.} \tag{44f}$$

E. Walking Specific Constraints

While taking a step, the robot legs do not have to collide with each other. Self-collisions constraints are usually hard to consider and may slow down consistently the determination of a solution. A simpler solution to avoid self-collisions between legs consists of avoiding cross-steps. We assume the frame attached to the right foot to have the positive y-direction pointing toward left. In this case, it would be sufficient to impose the y-component of the ${}^r\boldsymbol{x}_l$ (i.e., the relative position of the left foot expressed in the right foot frame) to be greater than a given quantity, i.e.

$$\boldsymbol{e}_2^\top\,{}^r\boldsymbol{x}_l \geq d_{\min}. \tag{45}$$

Too wide motions of the swing leg may cause other self-collisions, especially between the robot arms and thighs. Hence, we set an upper bound on the difference between the height of the two feet. To simplify the definition of the constraint, we consider the mean position of every contact point

$$-M_{hf} \leq e_3^\top \left({}_{\#}\boldsymbol{p}_l - {}_{\#}\boldsymbol{p}_r\right) \leq M_{hf} \tag{46}$$

where ${}_{\#}\boldsymbol{p}_l$ and ${}_{\#}\boldsymbol{p}_r$ are the mean positions of all the contact points of the left and right feet, respectively, i.e., ${}_{\#}\boldsymbol{p}_\circ = \frac{1}{n_p}\sum_i^{n_p} {}_i\boldsymbol{p}_\circ$. The number of contact points in a single foot is n_p, and $M_{hf} \in \mathbb{R}^+$ is the constraint upper bound.

Some additional constraints can be considered

$$x_{z\ \min} \leq h(\boldsymbol{x}) \tag{47a}$$

$$-\boldsymbol{M}_{h_\omega} \leq {}_{\bar{G}}\boldsymbol{h}^\omega \leq \boldsymbol{M}_{h_\omega}. \tag{47b}$$

Equation (47a) avoids the emergence of solutions that set the CoM position too close or even below the ground. Equation (47b) sets a bound $\boldsymbol{M}_{h_\omega} \in \mathbb{R}^3$ on the angular momentum. When considered together, (47) tends to avoid the emergence of trajectories that cause excessive motions or let the robot fall.

F. Tasks in Cartesian Space

In order to make the robot move toward a desired position, it is necessary to specify tasks in Cartesian space.

1) Contact Point Centroid Position Task

We define as a task the L2 norm of the error between a point attached to the robot and its desired position in an absolute frame. Suppose we choose the CoM position as a target point. By moving its desired value forward in space, the robot could simply lean toward the goal without moving the feet. This undesired behavior may lead the robot to fall. It is possible to avoid the robot leaning forward by locating the target point on the feet instead of the CoM. In particular, we select the centroid of the feet contact points as target, thus avoiding specifying a desired placement for each foot

$$\Gamma_{\#p} = \frac{1}{2}\|{}_{\#}\boldsymbol{p} - {}_{\#}\boldsymbol{p}^*\|^2_{\boldsymbol{W}_\#} \tag{48}$$

where ${}_{\#}\boldsymbol{p} = \frac{1}{2}({}_{\#}\boldsymbol{p}_l + {}_{\#}\boldsymbol{p}_r)$ and ${}_{\#}\boldsymbol{p}^* \in \mathbb{R}^2$ is its desired value.

2) CoM Linear Velocity Task

While walking, we want the robot to keep (an approximately) constant forward motion. In fact, since foot positions are not scripted, it may be possible to plan two consecutive steps with the same foot. Requiring a constant forward velocity helps avoiding such phenomena. This task can be defined as

$$\Gamma_{\bar{G}h^p} = \frac{1}{2}\|{}_{\bar{G}}\boldsymbol{h}^p - m\dot{\boldsymbol{x}}^*\|^2_{\boldsymbol{W}_v} \tag{49}$$

with $\dot{\boldsymbol{x}}^*$ a desired CoM velocity. The weights $\boldsymbol{W}_v$ allows selecting and weighting the different directions separately.

3) Foot Yaw Task

The task on the centroid of the contact points defined in Section IV-F1 allows us to specify the direction the robot has to step toward. The *foot yaw task* specifies at which angle the foot should be oriented with respect to the z-axis, i.e., the foot yaw angle. Define $\gamma_\circ^*$ as the desired yaw angle for either the left or the right foot ($\circ$ is a placeholder). We construct a unitary vector $\boldsymbol{\ell}_\circ^* \in \mathbb{R}^2$ belonging to the xy-plane (of $\mathcal{I}$), oriented such that the angle with the x-axis of $\mathcal{I}$ corresponds to $\gamma_\circ^*$. Its components are

$$\boldsymbol{\ell}_\circ^* = \begin{bmatrix}\cos(\gamma_\circ^*)\\ \sin(\gamma_\circ^*)\end{bmatrix}. \tag{50}$$

Similarly, the vector $\boldsymbol{\ell}_\circ \in \mathbb{R}^2$ is fixed to the foot and parallel to the foot x-axis, but expressed in the $\mathcal{I}$ frame. This vector can be easily obtained as a linear combination of the contact points position. The goal of the *foot yaw task* is to align $\boldsymbol{\ell}_\circ$ to $\boldsymbol{\ell}_\circ^*$, which can be (locally) achieved by minimizing their cross-product, thus leading to

$$\Gamma'_{\text{yaw}} = \sum_{l,r}\frac{1}{2}\left\|\begin{bmatrix}-\sin(\gamma_\circ^*)\cos(\gamma_\circ^*)\end{bmatrix}\boldsymbol{\ell}_\circ\right\|^2. \tag{51}$$

Notice that (51) has a minimum also when $\boldsymbol{\ell}_\circ$ is null. In other words, Γ'_{yaw} can be minimized by shrinking the projection on the xy-plane of the vector attached to the robot foot. This is undesired because it would set the foot to be perpendicular to the ground. Hence, we consider also a second vector attached to the foot and perpendicular to $\boldsymbol{\ell}_\circ$, called $\boldsymbol{\ell}_\circ^\perp$. This vector is parallel, and has the same direction of the foot y-axis. Hence, the final task has the following form:

$$\Gamma_{\text{yaw}} = \sum_{l,r}\frac{1}{2}\left\|\begin{bmatrix}-\sin(\gamma_\circ^*) & \cos(\gamma_\circ^*)\end{bmatrix}\boldsymbol{\ell}_\circ\right\|^2 + \sum_{l,r}\frac{1}{2}\left\|\begin{bmatrix}\cos(\gamma_\circ^*) & \sin(\gamma_\circ^*)\end{bmatrix}\boldsymbol{\ell}_\circ^\perp\right\|^2. \tag{52}$$

Equation (52) does not prevent the foot to have roll and pitch motions during the swing phase.

G. Regularization Tasks

The dynamical system in Section IV-D depends on a high number of variables. Despite the additional constraints defined in Section IV-E and the Cartesian tasks presented in Section IV-F, a consistent part of the system dynamics is not taken into account nor constrained. For this reason, it is necessary to introduce regularization tasks that contribute in generating walking trajectories.

1) Frame Orientation Task

While moving, we want a robot frame to take a desired orientation ${}^{\mathcal{I}}\boldsymbol{R}^*_{\text{frame}}$. We weight the distance of the rotation matrix ${}^{\mathcal{I}}\tilde{\boldsymbol{R}}_{\text{frame}} = {}^{\mathcal{I}}\boldsymbol{R}^{*\top}_{\text{frame}}\,{}^{\mathcal{I}}\boldsymbol{R}_{\text{frame}}$ from the identity. To this end, we convert ${}^{\mathcal{I}}\tilde{\boldsymbol{R}}_{\text{frame}}$ into a quaternion (through a function `quat`, that implements the Rodrigues formula, while keeping the real part always non-negative), and weight its difference from the identity quaternion $\boldsymbol{I}_q$, thus exploiting the Euclidean distance of quaternions as a metric for rotation error [64]

$$\Gamma_{\text{frame}} = \frac{1}{2}\left\|\texttt{quat}\left({}^{A}\tilde{\boldsymbol{R}}_{\text{frame}}\right) - \boldsymbol{I}_q\right\|^2. \tag{53}$$

It can be applied on multiple bodies, like the torso and waist.

2) Base Quaternion Derivative Regularization Task

The base quaternion derivative task allows tracking a desired angular velocity for the base. During walking, this task helps in regularizing the robot motion. It is defined as follows:

$$\Gamma_{\text{reg}u_\rho} = \frac{1}{2}\left\|\boldsymbol{u}_\rho - \boldsymbol{u}^*_\rho\right\|^2 \tag{54}$$

with $\boldsymbol{u}^*_\rho \in \mathbb{R}^4$ a desired quaternion rate of change.

3) Force Regularization Task

While considering foot contact forces independent from each other, they still belong to a single rigid body. Thus, we prescribe the contact forces in a foot to be as similar as possible, refraining from using partial contacts if not strictly necessary. This can be obtained through the following:

$$\Gamma_{\text{reg}f} = \sum_{l,r}\sum_{i}^{n_p}\frac{1}{2}\left\|{}_i\boldsymbol{f} - \text{diag}({}_i\boldsymbol{\alpha}^*)\sum_{j}^{n_p}{}_j\boldsymbol{f}\right\|^2_{\boldsymbol{W}_f}. \tag{55}$$

Here ${}_i\boldsymbol{\alpha}^* \in \mathbb{R}^3$ determines the desired ratio for force i with respect to the total force. For example, if we want all the forces in a foot to be equal, it is sufficient to select all the components of ${}_i\boldsymbol{\alpha}^* \in \mathbb{R}^3$ equal to $\frac{1}{n_p}$. In this case, the corresponding CoP is the centroid ${}_\#\boldsymbol{p}$. In other cases, it may be helpful to move the CoP somewhere else in the foot. In this case, it is sufficient to compute the corresponding ${}_i\boldsymbol{\alpha}^*$.

4) Joint Regularization Task

In order to avoid solutions with huge joint variations, we can introduce a regularization task for the joint variables

$$\Gamma_{\text{reg}s} = \frac{1}{2}\left\|\dot{\boldsymbol{s}} + \boldsymbol{K}_s(\boldsymbol{s} - \boldsymbol{s}^*)\right\|^2_{\boldsymbol{W}_j} \tag{56}$$

with $\boldsymbol{s}^*$ a desired joints configuration. The minimum for this cost is achieved when $\dot{\boldsymbol{s}} = -\boldsymbol{K}_s(\boldsymbol{s} - \boldsymbol{s}^*)$, with $\boldsymbol{K}_s \in \mathbb{R}^{n\times n}$ a positive semi-definite matrix. When this equality holds, joint values converge exponentially to their desired values $\boldsymbol{s}^*$. In this way, both joint velocities and joint positions are regularized.

5) Swing Height Task

When performing a step, the swing foot height usually ensures some level of robustness with respect to ground asperity. In order to specify a desired swing height, we can define the following task:

$$\Gamma_{\text{swing}} = \sum_{l,r}\sum_{i}^{n_p}\frac{1}{2}\left(\boldsymbol{e}_3^\top{}_i\boldsymbol{p} - {}_sh^*\right)^2\left\|\left[\boldsymbol{e}_1\,\boldsymbol{e}_2\right]^\top\boldsymbol{u}_{ip}\right\|^2. \tag{57}$$

Equation (57) penalizes the distance between the z-component of each contact point position from a desired height ${}_sh^* \in \mathbb{R}$ when the corresponding planar velocity is different from zero. Trivially, this cost has two minima: When the planar velocity is zero (thus the point is not moving) or when the height of the points is equal to the desired one.

6) Contact Control Regularization Tasks

Even if we assume the contact point velocities and force derivatives to be control inputs, we want to avoid the planner to use them always at the limit. Hence, we add some basic regularization tasks

$$\Gamma_{\text{reg}u_p} = \sum_{l,r}\sum_{i}^{n_p}\frac{1}{2}\left\|\boldsymbol{u}_{ip} - \boldsymbol{u}^*_{ip}\right\|^2_{\boldsymbol{W}_{u_p}} \tag{58}$$

$$\Gamma_{\text{reg}u_f} = \sum_{l,r}\sum_{i}^{n_p}\frac{1}{2}\left\|\boldsymbol{u}_{if} - \boldsymbol{u}^*_{if}\right\|^2_{\boldsymbol{W}_{u_f}} \tag{59}$$

with $\boldsymbol{u}^*_{ip}, \boldsymbol{u}^*_{if} \in \mathbb{R}^3$ the corresponding desired quantities.

H. Complete Optimal Control Problem

Given the set of equations listed in Section IV-D and the tasks described in Sections IV-F and IV-G, it is possible to define an optimal control problem whose complete formulation is presented below. Here the vector $\mathbf{w}$ contains the set of weights defining the relative "importance" of each task.

$$\underset{\boldsymbol{\chi},\boldsymbol{\mathcal{U}}}{\text{minimize}} \quad \mathbf{w}^\top\begin{bmatrix}\Gamma_{\#p}\\ \Gamma_{\bar{G}h^p}\\ \Gamma_{\text{frame}}\\ \Gamma_{\text{reg}u_\rho}\\ \Gamma_{\text{reg}f}\\ \Gamma_{\text{reg}s}\\ \Gamma_{\text{swing}}\\ \Gamma_{\text{yaw}}\\ \Gamma_{\text{reg}u_p}\\ \Gamma_{\text{reg}u_f}\end{bmatrix} \tag{60a}$$

subject to:

$$\dot{\boldsymbol{\chi}} = \boldsymbol{f}(\boldsymbol{\chi},\boldsymbol{\mathcal{U}}), \text{see (42)} \tag{60b}$$

$$\mathbf{l} \le \boldsymbol{g}\left(\boldsymbol{\chi},\boldsymbol{\mathcal{U}}\right) \le \mathbf{u}, \text{see (43)–(44)}, \tag{60c}$$

$$\boldsymbol{e}_2^{\top r}\boldsymbol{x}_l \ge d_{\text{min}} \tag{60d}$$

$$x_{z\text{ min}} \le h(\boldsymbol{x}) \tag{60e}$$

$$-\boldsymbol{M}_{h_\omega} \le {}_{\bar{G}}\boldsymbol{h}^\omega \le \boldsymbol{M}_{h_\omega} \tag{60f}$$

$$-M_{hf} \le e_3^\top\left(\#\boldsymbol{p}_l - \#\boldsymbol{p}_r\right) \le M_{hf}. \tag{60g}$$

Here, the state variables $\boldsymbol{\mathcal{X}}$ are those derived in time, while $\boldsymbol{\mathcal{U}}$ contains all the control inputs. Thus,

$$\boldsymbol{\chi} = \begin{bmatrix}{}_i\boldsymbol{f}\\ {}_i\boldsymbol{p}\\ \vdots\\ {}_{\bar{G}}\boldsymbol{h}\\ \boldsymbol{x}\\ {}^{\mathcal{I}}\boldsymbol{p}_B\\ {}^{\mathcal{I}}\boldsymbol{\rho}_B\\ \boldsymbol{s}\end{bmatrix}, \quad \boldsymbol{\mathcal{U}} = \begin{bmatrix}\boldsymbol{u}_{if}\\ \boldsymbol{u}_{ip}\\ \vdots\\ {}^B\boldsymbol{v}_{\mathcal{I},B}\\ \boldsymbol{u}_\rho\\ \boldsymbol{u}_s\end{bmatrix} \tag{61}$$

where the symbol $\vdots$ represents the repetition of the corresponding variables for each contact point.

I. Considerations

The optimal control problem presented in Section IV-H is designed such that (almost) no constraint is task-specific. As a consequence, it is particularly important to define the cost function carefully since the solution is a trade-off between all the various tasks. On the other hand, the detailed model of the system allows us to achieve walking motions without specifying a desired CoM trajectory or by fixing the angular momentum to zero. Nevertheless, due to the limited time horizon, it is better to prevent the solver from finding solutions that would lead to unfeasible states in future planner runs. For this reason, (47a) and (47b) are added.

In order to plan for trajectories longer than the prediction horizon, we apply the receding horizon principle [65]. Hence, the planner is called iteratively, initialized with the results from the previous run. A possible effect resulting from the application of the receding horizon principle is the emergence of "procrastination" phenomena. Due to the moving horizon, the solver may continuously delay in actuating motions, since the task keeps being shifted in time. A simple fix to this phenomena is to increase the task weights with time, such that it is more convenient to reach a goal position earlier.

In addition, since the optimal control problem described in Section IV-H uses a quaternion as parametrization for the base rotation, it is necessary to preserve its unit norm. Since this problem is solved using iterative nonlinear optimization solvers, the enforcement of (43d) is not enough, as the solver might perform intermediate unfeasible iterations. Hence, we normalize the base quaternion every time we evaluate the costs and constraints.

Finally, given that the problem under consideration is nonconvex, the optimizer will find a local minimum. This may result in a suboptimal solution for the given tasks, but this fact does not limit the applicability of the results to the robot. In this context, a proper problem initialization plays a pivotal role. For the first run only, the planner is initialized by setting the force on the left foot to be null for the entire horizon. The position of the left foot is such that the centroid of the contact points is on the desired position. In this way, we hint the optimizer to use the left foot for the first step. In successive runs, the solver is warm-started with the solution computed in the previous run. Even if it affects only the first run, the initial guess can strongly influence the generated motion.

V. Validation

We present the results obtained when solving the optimal control problem presented in Section IV-H. In particular, we test its capabilities to generate whole-body walking trajectories for a flat ground using the model of the iCub humanoid robot [66]. All the tests presented in this section have been carried on a 7-generation Intel Core i7@2.8 GHz laptop.

The optimal control problem described in Section IV-H is solved using a direct multiple shooting method [58]. The system dynamics, defined in (42), is discretized adopting an implicit trapezoidal method with a fixed integration step. The corresponding optimization problem is solved, thanks to `Ipopt` [67], using the `MA57` linear solver [68]. The solver requires at least the first derivative of the cost and constraints with respect to the optimization variables. These are computed explicitly using the derivation presented in [69]. The pipeline from the problem definition to its solution is implemented using the

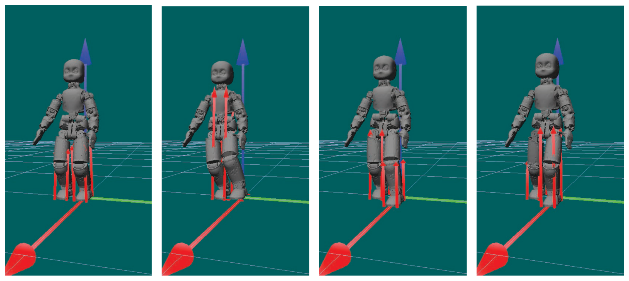


Fig. 1. Snapshots of the generated walking motion. The red arrows indicate the force required at each contact point scaled by a factor of 0.01. These images have been obtained using the complementarity constraints of Section III-A. (a) $t = 0.5$ s. (b) $t = 1.5$ s. (c) $t = 2.5$ s. (d) $t = 3.5$ s.

`iDynTree::optimalcontrol`[3] library, allowing for easy testing of other integrators or solvers. The code is implemented entirely in C++ and open-source.[4] The walking trajectories are generated with a fixed prediction window of 2 s. The horizon is large enough to predict at least one full step.

The planner is set up using an integration step of $100\,\mathrm{ms}$, while the time horizon is $2\,\mathrm{s}$. The choice of a large integration step serves for two reasons. First, it reduces the number of variables used by the optimization problem (fixing the time horizon). Second, it allows inserting another control loop at higher frequency. After each run of the planner, the first and the second states are added to the final trajectory. The latter is also used as a feedback state for the new planner run in a *receding horizon* fashion.

When planning, we control 23 of the robot joints. We consider four contact points for each foot, located at the vertices of the rectangle enclosing the robot foot. Concerning the references, the desired position for the centroid of the contact points is moved $10\,\mathrm{cm}$ along the walking direction every time the robot performs a step. A simple state machine where the reference is moved as soon as a step is completed, is enough to generate a continuous walking pattern. The speed is modulated by prescribing a desired CoM forward velocity equal to $5\,\mathrm{cm/s}$. The desired joint positions are fixed, and equal to the initial configuration of the robot. The terms $\boldsymbol{u}^*_{\rho}$, $\boldsymbol{u}^*_{ip}$, and $\boldsymbol{u}^*_{if}$ are set to zero. The value of k_t (see Section IV-B) is $10\,\mathrm{m}^{-1}$.

Fig. 1 shows some snapshots of the first generated step, while Fig. 2 shows the position of one of the right contact points. It is possible to recognize the different walking phases, though they are not planned *a priori*. Nevertheless, the controller does not specify explicitly when a phase begins and ends. It is also possible to notice that the trajectories obtained using the *hyperbolic secant* method, Section III-B, have been affected by some initial "procrastination," with a longer initial double support phase. Those produced with the other two methods are instead very similar.

Fig. 3 presents the planned CoM position. Here, it is possible to notice that the position along the x direction grows at a

[3][Online]. Available: https://github.com/robotology/idyntree/tree/devel/src/optimalcontrol

[4][Online]. Available: https://github.com/dic-iit/dynamical-planner

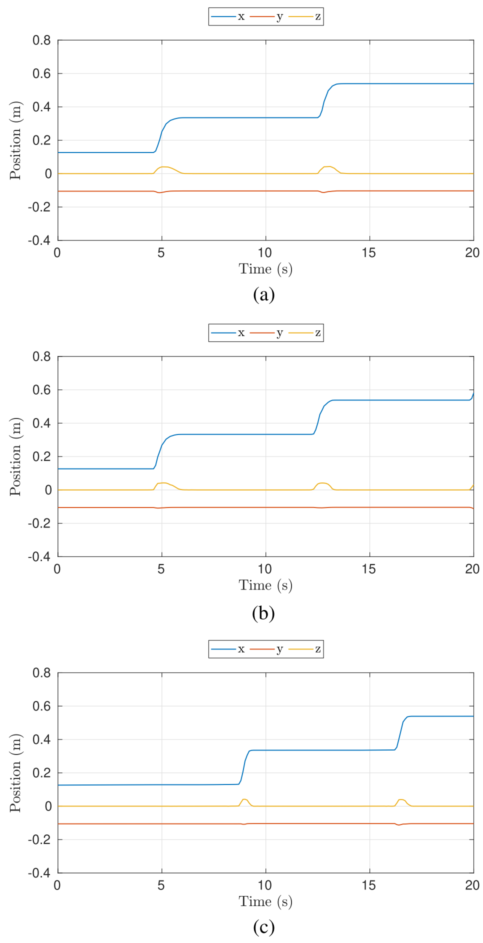


Fig. 2. Planned position of one of the right foot contact points using different complementarity constraints. The walking phases are recognizable, but they are not defined beforehand. The controller does not specify directly when a phase begins and ends. (a) Relaxed complementarity. (b) Dynamically enforced complementarity. (c) Hyperbolic secant in control bounds.

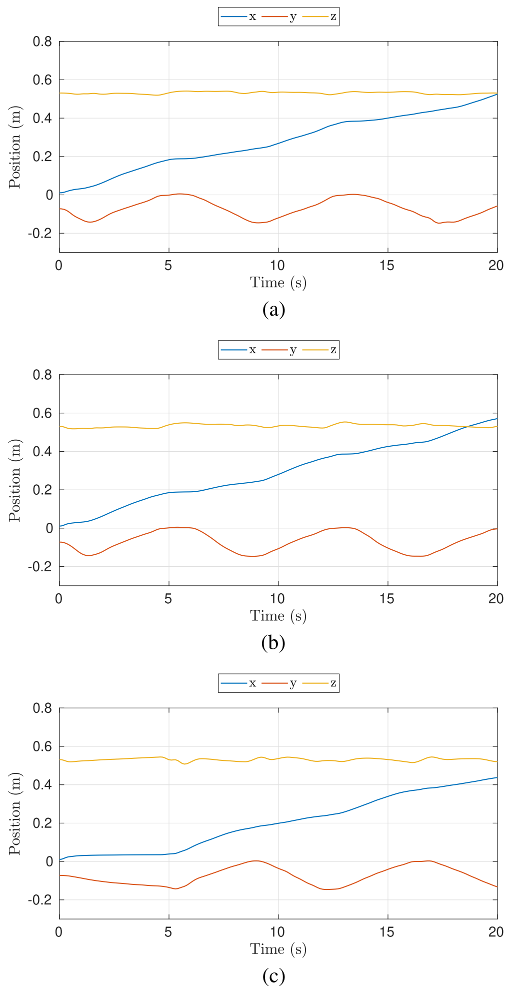


Fig. 3. Planned CoM position using different complementarity constraints. It is possible to notice a continuous velocity on the x direction. The plots appear a little irregular. This may be a consequence of the chosen time step. (a) Relaxed complementarity. (b) Dynamically enforced complementarity. (c) Hyperbolic secant in control bounds.

constant rate. This is a direct consequence of the task on the CoM velocity presented in Section IV-F2. Notice that the bound on the CoM height, $x_{z\ \mathrm{min}}$, is set to half of the initial robot height, but such constraint is never activated. These are the results of consecutive runs of the optimal control problem described in Section IV-H. From one run to another, the solver may find slightly different solutions because of the shift in the prediction horizon, causing the irregularities shown in the figures. In addition, there is no regularization on the CoM position, whose trajectory is fully determined by the solver.

Fig. 4 shows the planned angular momentum, that is not fixed to zero, but limited to $10\,\mathrm{kg\,m^2/s}$. This limit is never reached. Thanks to the regularization terms on the base and joint velocities, the angular momentum is kept close to zero.

It is worth stressing that none of the tasks described above define how and when to raise the foot. By prescribing a reference for the centroid of the contact points and by preventing the motion on the contact surface, swing motions are planned automatically. Nevertheless, this advantage comes with a cost. It is difficult to define a desired swing time and, more importantly, the relative importance of each task, i.e., the values of $\mathbf{w}$, must be chosen carefully. As anticipated above, in order to avoid "procrastination" phenomena, we adopted a time-varying weight for the centroid position of the contact points, described in Section IV-F1. In particular, we define

$$\tilde{w}_{\#p}(t) = \alpha_{ts}(t) w_{ts} + 1 \tag{62}$$

where $w_{ts} = 30$, while α_{ts} determines the current percentage of the step. It is computed as follows:

$$\alpha_{ts}(t) = \frac{t - t_{s,\mathrm{init}}}{t_s^*} \tag{63}$$

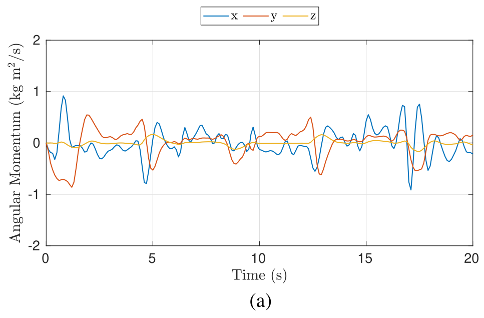
where t_s^* is the desired step duration, while $t_{s,\mathrm{init}}$ is the first moment in which a specific position reference is active.

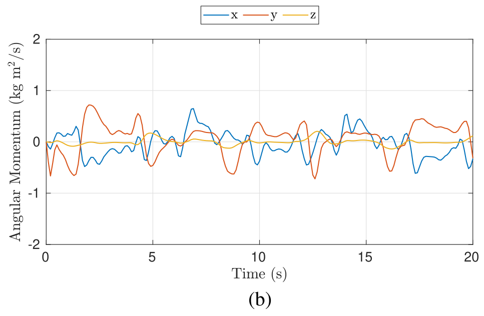
At the same time, since only the initial states of a given solution are considered in the final trajectory, it is necessary to increase the importance of some tasks at the beginning of the horizon. One specific example is the orientation of the torso. In order to avoid undesired motions for the robot upper body, we set a weight for the torso orientation task that decreases

TABLE I
COST FUNCTION WEIGHTS USED IN THE COMPARISON OF THE COMPLEMENTARITY METHODS

$w_{\#p}$	$w_{\bar{G}h^p}$	$w_{\text{frame}-\text{torso}}$	$w_{\text{frame}-\text{pelvis}}$	$w_{\text{reg}u_\rho}$	$w_{\text{reg}f}$	$w_{\text{reg}s}$	w_{swing}	w_{yaw}	$w_{\text{reg}u_p}$	$w_{\text{reg}u_f}$
$100.0 \cdot \tilde{w}_{\#p}(t)$	1.0	$90.0 \cdot \tilde{w}_{\text{frame}-\text{torso}}(t)$	50.0	0.001	0.1	0.1	1000	1000	5.0	0.0001

The terms $\tilde{w}_{\#p}(t)$ and $\tilde{w}_{\text{frame}-\text{torso}}(t)$ are defined in (62) and (64), respectively.





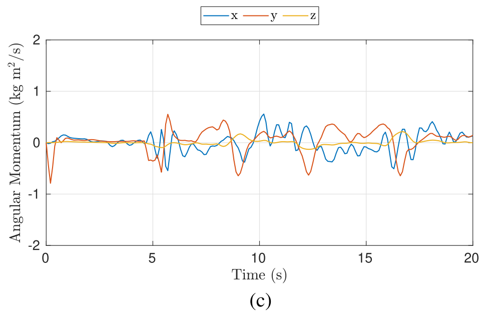


Fig. 4. Planned angular momentum obtained adopting different complementarity constraints. (a) Relaxed complementarity. (b) Dynamically enforced complementarity. (c) Hyperbolic secant in control bounds.

exponentially with time. In particular, we have

$$\tilde{w}_{\text{frame}-\text{torso}}(t) = 100e^{-10t} + 1 \tag{64}$$

where we assume, without loss of generality, that $t \in [0, 2]$.

The weights we adopted in the cost function are listed in Table I. During experiments, we adopted an incremental approach to determine them. We added the tasks one by one, starting from $\Gamma_{\#p}$ and then we gradually refine the walking motion by tuning one cost at a time.

A. Real Robot Validation

To further validate the output of the planner presented in this part, we tested the generated trajectories on the iCub humanoid robot. In particular, they are used as a reference for a joint position controller. Since their frequency is at 10 Hz, we interpolate them to have a reference point every 10 ms. Hence, the trajectories are replayed on the robot closing the loop only at joint level. The robot performed several steps, Fig. 5, demonstrating the feasibility of the generated trajectories.

Compared to the results shown in Section V, we reduce the forward speed to 3 cm/s, advancing the mean point reference of 6cm at every step. In this case, it has been also useful to move the desired CoP position, as anticipated in Section IV-G3. In particular, by moving it toward the inner foot edge, the robot walks more robustly. The cost function weights are listed in Table II, while $w_{ts} = 60$.

The trajectories are generated offline for a time span of 20 s. They are tested in open-loop, thus limiting the maximum velocity achievable by the robot. This specific test is aimed at validating the formulation of the optimal control problem described in Section IV-H, independently from the choice of the complementarity method (in this case, we selected the dynamically enforced complementarity method). In the next section, we analyze the difference between each method.

VI. DCCs COMPARISONS

In this section, we analyze the differences among the complementarity methods described in Section III-C. In order to asses the relative performances of the different methods, we adopt the DCCs in two contexts with a crescendo of complexity. First, in Section VI-A, we apply the DCCs in a toy problem involving a single contact. Then, in Section VI-B, we exploit the output of the nonlinear trajectory planning framework presented in Section IV.

As a measure of performance, we adopt the product between the normal force and the height of the contact point from the ground, i.e., ${}_ip_z \cdot {}_if_z$. In other words, we test the accuracy with which (13) is satisfied, simplifying the formulation, thanks to the planar ground assumption. As another performance measure, we use the average computational time needed to solve the corresponding optimization problem.

In order to have the results of Sections VI-A and VI-B comparable, we use the same integration method, time horizon, and integration step described in Section V in both cases. Similarly, the `Ipopt` configuration is identical.

Moreover, for the sake of reproducibility, the comparisons involving parameter variations run also on `Github Actions`.[5] Thus, we provide a comparison using a standard machine. In this latter case, `Ipopt` uses the openly available `MUMPS` [70] linear solver.

[5][Online]. Available: https://github.com/ami-iit/paper_dafarra_2022_tro_dcc-planner

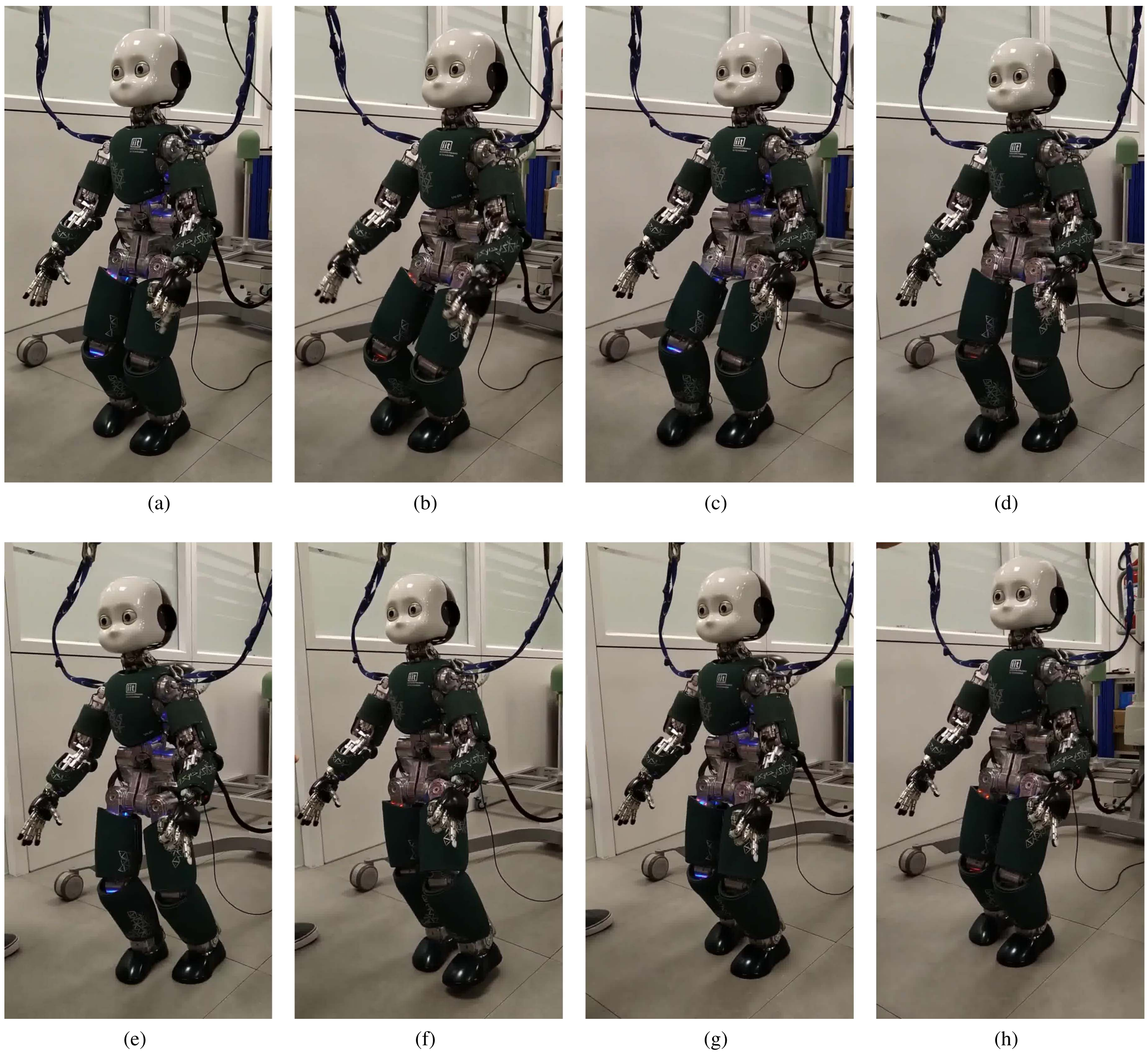


Fig. 5. Snapshots of the robot walking using the planned trajectories. The planner generates joint references which are interpolated and stabilized through a joint position controller. (a) $t = t_0$. (b) $t = t_0 + 1$ s. (c) $t = t_0 + 2$ s. (d) $t = t_0 + 3$ s. (e) $t = t_0 + 4$ s. (f) $t = t_0 + 5$ s. (g) $t = t_0 + 6$ s. (h) $t = t_0 + 7$ s.

TABLE II
COST FUNCTION WEIGHTS USED IN THE REAL ROBOT VALIDATION

$w_{\#p}$	$w_{\bar{G}h^p}$	$w_{\text{frame}-\text{torso}}$	$w_{\text{frame}-\text{pelvis}}$	$w_{\text{reg}u_\rho}$	$w_{\text{reg}f}$	$w_{\text{reg}s}$	w_{swing}	w_{yaw}	$w_{\text{reg}u_p}$	$w_{\text{reg}u_f}$
$100.0 \cdot \tilde{w}_{\#p}(t)$	1.0	90.0	20.0	0.0	0.1	0.1	1000	1000	5.0	0.0

The term $\tilde{w}_{\#p}(t)$ is defined in (62).

The various methods have been tuned to obtain the best complementarity accuracy, while remaining able to generate a walking pattern. Indeed, parameters that are too "restrictive" (e.g., an ϵ too small) may prevent the solver from finding a walking strategy because the points are not able to move. On the other hand, low accuracy may mean the solver requires a force of considerable magnitude when the point is still raised from the ground. The same parameters have been adopted also for the toy problem. The parameters are chosen as follows.

1) Relaxed complementarity: $\epsilon = 0.004[\mathrm{N} \cdot \mathrm{m}]$.
2) Dynamically enforced complementarity: $K_{\text{bs}} = 20[\frac{1}{\mathrm{s}}]$, $\varepsilon = 0.05[\frac{\mathrm{N} \cdot \mathrm{m}}{\mathrm{s}}]$.
3) Hyperbolic secant in control bounds: $K_{f,z} = 250[\frac{1}{\mathrm{s}}]$, $k_h = 500[\frac{1}{\mathrm{m}}]$.

$\boldsymbol{M}_f$ is set to $100\mathrm{N/s}$ for all the components and it is common for all the methods.

A. Comparison Using a Toy Problem

We consider a point mass falling vertically from a given height and approaching ground. We assume the ground to be infinitely rigid, and the ground level to be at zero. These settings render the analyzed toy problem interesting also from a robotics perspective since it exemplifies the case of a humanoid robot foot approaching ground. Then, the mass position $x_m \in \mathbb{R}$ has to satisfy the following constraint:

$$x_m \geq 0. \tag{65}$$

When the mass hits the ground, a force is applied to it. This is modeled through the *DCCs* presented in Section III. We assume the impact to be completely inelastic. On the other hand, the dynamics of the mass would not be continuous since its velocity should suddenly go to zero at the moment of the impact. In our toy problem, we circumvent this issue by assuming to be able to control the mass acceleration, as if a propeller is powering the mass.

The resulting system has the following dynamics:

$$\dot{x}_m = v_m \tag{66a}$$

$$\dot{v}_m = g + \frac{1}{m}\left(f_m + p_m\right) \tag{66b}$$

$$\dot{f}_m = u_f \tag{66c}$$

where $v_m, g, f_m, p_m, u_f \in \mathbb{R}$ are, respectively, the mass velocity, the gravity acceleration, the contact force, the propeller thrust force, and the contact force derivative. The control inputs are given by p_m and u_f, and the magnitude of u_f is limited to a maximum value M_{u_f}. The objective is to minimize the use of p_m as if the propeller had fuel constraints.

We insert the dynamical system of (66) and the DCCs in the following optimal control problem:

$$\text{minimize} \quad p_m^2 \tag{67a}$$

subject to:

$$\dot{x}_m = v_m \tag{67b}$$

$$\dot{v}_m = g + \frac{1}{m}\left(f_m + p_m\right) \tag{67c}$$

$$\dot{f}_m = u_f \tag{67d}$$

$$x_m \geq 0 \tag{67e}$$

$$f_m \geq 0 \tag{67f}$$

$$-M_{u_f} \leq u_f \leq M_{u_f} \tag{67g}$$

$$\text{Complementarity, see Section III-C.} \tag{67h}$$

1) Toy Problem Solution

The optimal control problem described by (67) is transcribed into an optimization problem using a direct multiple shooting method and solved through `Ipopt`. We use a prediction window of 2 s, with an integration step of 100ms as in Section V. The initial guess of the optimization problem is as follows. We compute the expected mass position and velocity assuming no propeller is used. The initial guess for the position and velocity is set to zero after the impact. Similarly, the guess for contact force is set equal to the weight after the impact.

Differently from Section V, the implementation has been fast-prototyped using `MATLAB`, exploiting the `CasADi` framework [71]. This excludes possible implementation-specific biases.

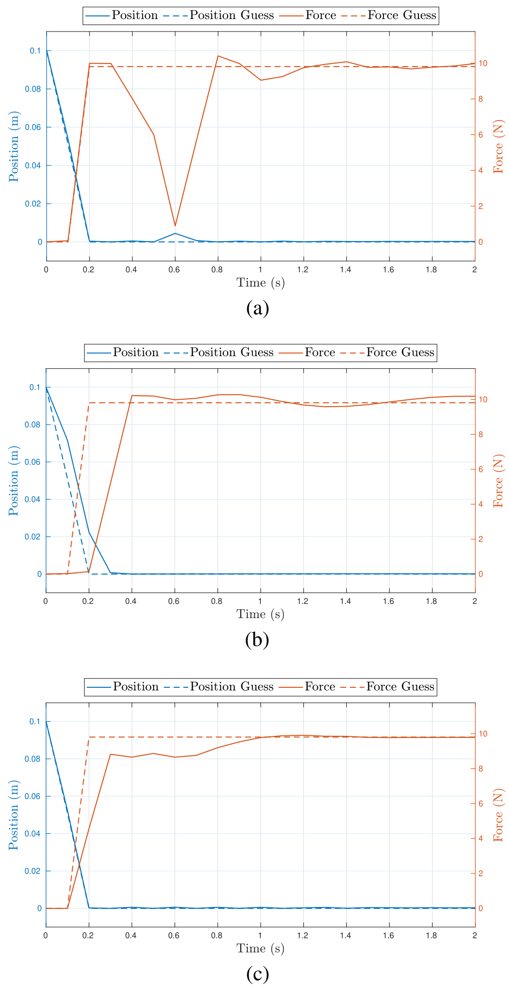


Fig. 6. Planned mass position, and contact force of the toy problem adopting different complementarity constraints. The dashed lines correspond to the initialization provided to the optimal control problem. (a) Relaxed complementarity. (b) Dynamically enforced complementarity. (c) Hyperbolic secant in control bounds.

Fig. 6 presents the output of the trajectory optimization for a unitary mass falling from 10 cm. We present the results using the different complementarity methods. The dashed lines correspond to the initialization fed to the optimizer. Fig. 7 shows the accuracy achieved by each method and the use of the propeller. The red dashed line displays the average accuracy.

The major complexity of the problem is represented by the complementarity constraints, as all the other constraints are linear. Hence, any difference in the output can be safely ascribed to the chosen complementarity method. In the following we analyze the performance of each complementarity method in case of parameter variations.

2) Comparison Using Different Initial Positions

In this section, we analyze the performance of the different complementarity methods by varying the initial mass height. Since the toy problem can be seen as an approximation of a robot foot approaching the ground, we consider an initial height that is compatible with the step heights achievable by a small humanoid robot. Table III presents the computational time, the

TABLE III
COMPUTATIONAL TIME, AVERAGE ACCURACY, AND COST OF EACH COMPLEMENTARITY METHOD WHEN SOLVING THE TOY PROBLEM AT DIFFERENT INITIAL HEIGHTS

Test running on a laptop using the `MA57` linear solver

Initial height	Relaxed			Dynamically enforced			Hyperbolic secant		
	Comp time (s)	Avg accuracy ($\mathrm{N \cdot m}$)	Cost (N^2)	Comp time (s)	Avg accuracy ($\mathrm{N \cdot m}$)	Cost (N^2)	Comp time (s)	Avg accuracy ($\mathrm{N \cdot m}$)	Cost (N^2)
5cm	0.0779	0.0021	**218.31**	0.0848	0.0013	298.48	**0.0509**	**0.0001**	1493.0
7cm	0.0680	0.0021	366.36	0.0702	**0.0019**	**305.35**	**0.0525**	0.0026	458.95
9cm	0.1040	0.0021	602.05	**0.0808**	**0.0013**	**325.93**	0.5353	0.0021	696.14
11cm	0.0758	0.0020	**363.75**	**0.0476**	**0.0012**	365.32	0.0609	0.0025	1060.9
13cm	0.0709	0.0021	**390.14**	**0.0437**	**0.0013**	429.06	0.0598	0.0025	1500.0
15cm	0.0630	0.0020	**458.18**	**0.0547**	**0.0011**	469.49	0.1627	0.0017	1110.2

Test running on `Github Actions` using the `MUMPS` linear solver

Initial height	Relaxed			Dynamically enforced			Hyperbolic secant		
	Comp time (s)	Avg accuracy ($\mathrm{N \cdot m}$)	Cost (N^2)	Comp time (s)	Avg accuracy ($\mathrm{N \cdot m}$)	Cost (N^2)	Comp time (s)	Avg accuracy ($\mathrm{N \cdot m}$)	Cost (N^2)
5cm	0.0756	0.0021	**219.43**	0.0586	0.0013	288.55	**0.0509**	**0.0001**	1493.0
7cm	0.2036	0.0020	347.64	**0.0656**	**0.0015**	**335.32**	0.0770	0.0026	458.95
9cm	0.1231	0.0021	600.02	**0.0449**	**0.0013**	**350.52**	0.1831	0.0025	701.91
11cm	0.0927	0.0021	**323.58**	**0.0475**	**0.0012**	365.32	0.0907	0.0026	1061.2
13cm	0.0712	0.0020	**403.46**	**0.0426**	**0.0011**	420.55	0.0923	0.0025	1500.3
15cm	**0.0578**	0.0020	**458.10**	0.0587	**0.0012**	473.44	0.1744	0.0020	1120.5

The results are the average of 100 repetitions. In bold the best resulting method by row.

average accuracy of the output trajectory, and the corresponding cost when using different complementarity methods, averaged over 100 repetitions. We highlight in bold the lowest computational time, the best accuracy, and the lowest cost for each initial height. Overall, the *dynamically enforced* method is the one providing best accuracy, except for the 5 cm case, where the *hyperbolic secant* method performs better. The same result holds also when running the test on `Github Actions`. For what concerns the computational time, the *hyperbolic secant* method performs better for heights lower than 9 cm, while the *dynamically enforced* is the fastest in the other cases, with the exception of the 15 cm height. In fact, in the test running on `Github Actions`, the *relaxed* method performed better. This latter method is also the one providing the smallest cost in the majority of the cases. We can notice a correlation between the accuracy and the cost value. In fact, a higher use of the propeller reduces the time in which the mass is in contact with the ground, resulting in a lower average accuracy. Hence, there is a trade-off between the use of the propeller and the accuracy. Such trade-off is visible in the 5 cm case, where the *hyperbolic secant* method has very little accuracy, but with the highest cost. Another thing to notice is that the cost increases with the initial height. In fact, higher initial heights involve higher impact velocities that are counteracted by the use of the propeller.

Finally, we highlight that the *dynamically enforced* method is the one providing the least variations in terms of computational time across different heights.

3) Comparison After Parameter Variations
In this section, we compare the complementarity methods by varying the following parameters associated with each complementarity method:

1) ϵ for the *relaxed complementarity*;
2) K_{bs}, ε for the *dynamically enforced complementarity*;
3) $K_{f,z}$ and k_h for the *hyperbolic secant in control bounds*.

The results are depicted in Fig. 8, where each point has an associated label displaying the parameters being used. The initial mass height is set to 0.1cm and the points represent the average result over 100 repetitions. The *dynamically enforced* method with the parameters $(\varepsilon\, 0.05,\ K_{bs}\, 20)$ is the one providing the best accuracy. At the same time, the *hyperbolic secant* method with parameters $(k_h\, 400,\ K_{f,z}\, 250)$ is the one producing the fastest solution. In general, we can observe that the *dynamically enforced* and the *hyperbolic secant* method are those closer to the origin, thus providing the most accurate solutions at the lowest time. On the other hand, we can notice that the *hyperbolic secant* method is more sensitive to parameter variations, since the computational time can largely increase. Moreover, when using `MUMPS` on `Github Actions`, the *hyperbolic secant* method is the one suffering the highest performance loss. Indeed, it can be noticed that its corresponding points appear to be shifted toward right in Fig. 8(b). For what concerns the *relaxed complementarity* points, a higher ϵ corresponds to a lower accuracy, trading off with the computational time.

B. Comparison Using the Planner of Section IV

In this section, we use the nonlinear trajectory planning framework presented in Section IV to compare the *dynamic complementarity conditions*. We perform a first comparison using the same setup presented in Section V. The accuracy, namely the product ${}_i p_z \cdot {}_i f_z$, is depicted in Fig. 9 for each contact point of both feet, and numerically summarized in Table IV. The *dynamically enforced* complementarity method is the one with the best numeric performance, although very close to the *relaxed* complementarity method. In this context, the *hyperbolic secant* method is outperformed. The particularity of this method is that when the contact points are raised from the ground, the force drops to zero consistently on all the points, as it is possible to notice from Fig. 9. In fact, with this method, we force the normal

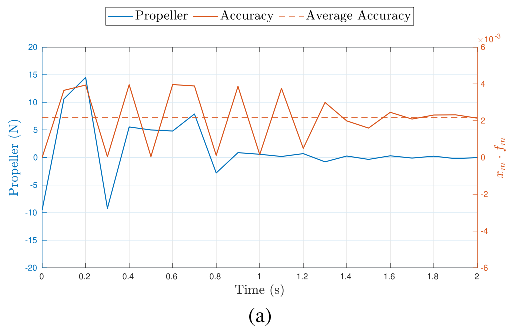
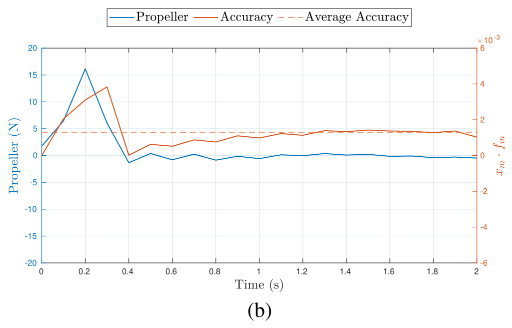
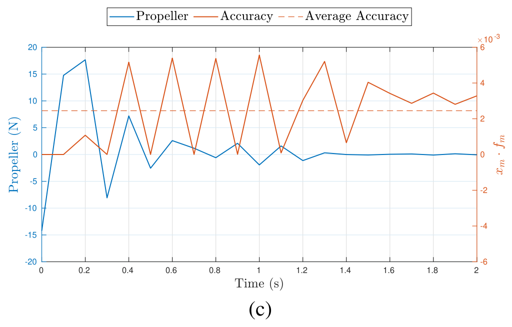




Fig. 7. Propeller value, and complementarity accuracy, of the toy problem adopting different complementarity constraints. The dashed red line is the average accuracy. (a) Relaxed complementarity. (b) Dynamically enforced complementarity. (c) Hyperbolic secant in control bounds.

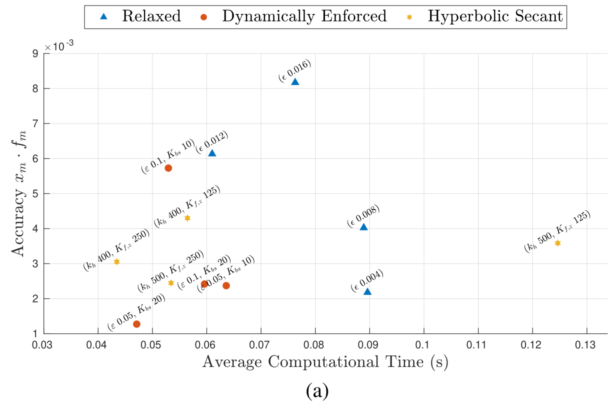
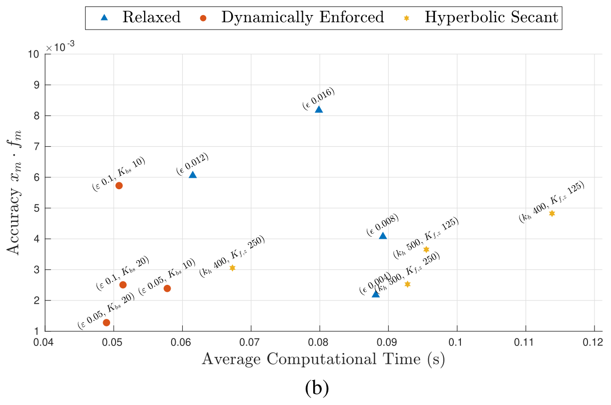



Fig. 8. Performance of the different complementarity methods in the toy problem after parameter changes. The closer to the origin, the better is the performance. (a) Test running on a laptop using the `MA57` linear solver. (b) Test running on `Github Actions` using the `MUMPS` linear solver.

TABLE IV
ACCURACY OF THE DIFFERENT COMPLEMENTARITY METHODS IN THE SETUP OF SECTION V

$p_z \cdot f_z$	Relaxed		Dynamically enforced		Hyperbolic secant	
	Left	Right	Left	Right	Left	Right
Average	0.0022	0.0023	**0.0014**	**0.0014**	0.0076	0.0081
Variance	0.0007	0.0007	**0.0006**	**0.0005**	0.0036	0.0042
Maximum	0.0040	0.0040	**0.0039**	**0.0038**	0.0148	0.0172

We consider the left and right foot separately. The best resulting method is in bold.

TABLE V
TIME PERFORMANCES USING DIFFERENT COMPLEMENTARITY METHODS IN THE SETUP OF SECTION V

[s]	Relaxed	Dynamically enforced	Hyperbolic secant
Average	6.47	5.93	**5.11**
Variance	7.99	5.65	**5.17**
Minimum	0.80	1.17	**0.73**
Maximum	44.93	30.91	**23.71**

All times are showed in seconds and obtained after 100 runs of the solver. The best result for each row is in bold.

force derivative to be strongly negative when the point is not on the ground. At the same time, this method does not prevent the point to move when a force is applied, possibly explaining the worst average accuracy compared to the other methods.

Fig. 10 presents the computational time required by the optimizer to complete the 100 planner runs that define the walking trajectory presented in Section V. Table V presents the numerical results. In this case, the *hyperbolic secant* method is the best performing, requiring in average more than a second less compared to worst performing method, the *relaxed* complementarity. Nonetheless, a reason for this difference of performance can be explained by the longer initial double support phase observed when using the *hyperbolic secant* method. In fact, by looking at Fig. 10, it is possible to notice that the computational time in the first runs was much lower than the other methods. For all the methods, it is possible to notice some pattern in the peaks. In particular, we have a high variation in the computational time when we move the reference for the centroid of the contact points. Hence, the planner has to predict a full new step. Since we initialize the planner with the previously computed solution, in this instant, the optimal point is far from the initialization. Hence, more time is required to find a solution. This issue can be addressed by providing the planner with a more effective initialization.

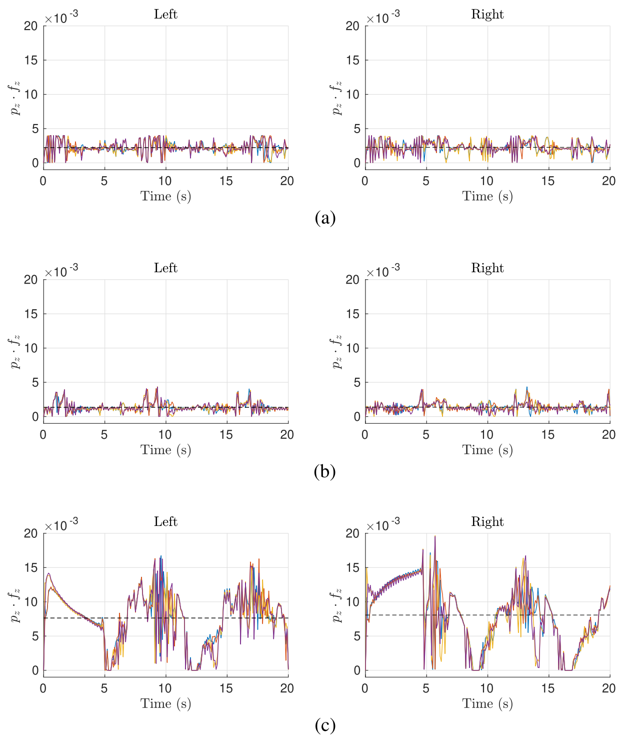


Fig. 9. Product between the vertical position and the normal force of each contact point, separated by foot, when using the different complementarity methods summarized in Section III-C in the setup of Section V. The black dashed lines indicate the mean values. By plotting the result of $p_z \cdot f_z$ for each point, we show the accuracy of each method in simulating a rigid contact. (a) Relaxed complementarity. (b) Dynamically enforced complementarity. (c) Hyperbolic secant in control bounds.

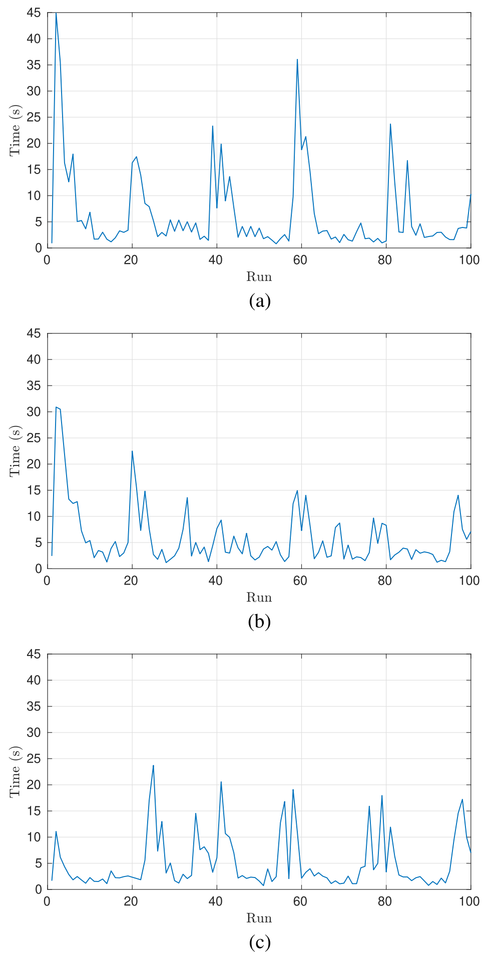


Fig. 10. Computational time required by each nonlinear trajectory planner run. (a) Relaxed complementarity. (b) Dynamically enforced complementarity. (c) Hyperbolic secant in control bounds.

TABLE VI
AVERAGE COMPUTATIONAL TIME USING DIFFERENT COMPLEMENTARITY METHODS AT DIFFERENT DESIRED WALKING SPEEDS

Test running on a laptop using the `MA57` linear solver

Speed (m/s)	Relaxed	Dynamically enforced	Hyperbolic secant
0.05	6.47	5.93	**5.11**
0.06	–	**6.89**	7.16
0.07	6.99	**5.64**	–

Test running on `Github Actions` using the `MUMPS` linear solver

Speed (m/s)	Relaxed	Dynamically enforced	Hyperbolic secant
0.05	13.19	**9.88**	11.70
0.06	–	**9.51**	12.72
0.07	11.57	13.07	**11.07**

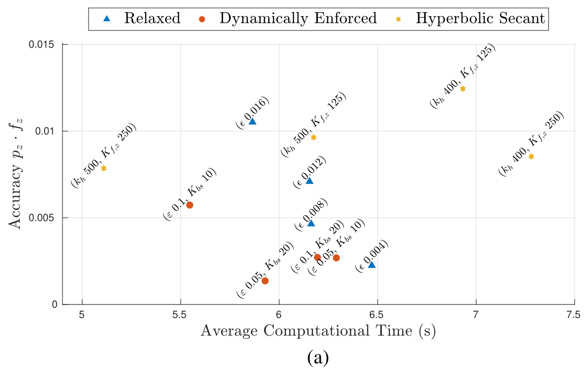
The symbol – indicates that the planner was not able to plan a walking motion. The best result for each row is in bold.

1) *Comparison With Different Walking Velocities*

As a second comparison, we test the output of the planner at different desired walking speeds, measuring the corresponding computational time. Table VI presents the numerical results. Compared to Section V, the only parameter difference is the desired walking velocity. We notice that this (apparently) small change can have a strong impact, eventually making the planner to fail in finding walking patterns, keeping the robot constantly in double support. This has been the case for the *relaxed* complementarity method at 0.06 m/s, and the *hyperbolic secant* method at 0.07 m/s (when using the `MA57` linear solver). In general, this behavior can be avoided with small variations of the initial guess, suggesting that the planner might get stuck in local minima. Nonetheless, for the sake of the comparison, we avoided additional changes. In this setting, the *dynamically enforced* method appears to be the most robust and consistent, even when running the test on different machines using different linear solvers.

2) *Comparison After Parameter Variations*

As a third comparison, we keep the desired forward velocity fixed while varying the same parameters as in Section VI-A3. The results are depicted in Fig. 11, where each point is labeled with the parameters used. The plot presents the trade-off between the computational time and the accuracy. This is particularly evident by looking at the *relaxed* complementarity points. The higher the parameter ϵ, the lower is the computational time, but at the same time, the accuracy decreases. When running the test using a personal laptop running `Ipopt` with `MA57`, the *dynamically enforced* method seems to remain always below the line traced by these points, hence showing better overall performances in this setting. On the contrary, the *hyperbolic secant* method has proved to be very fast with the parameters



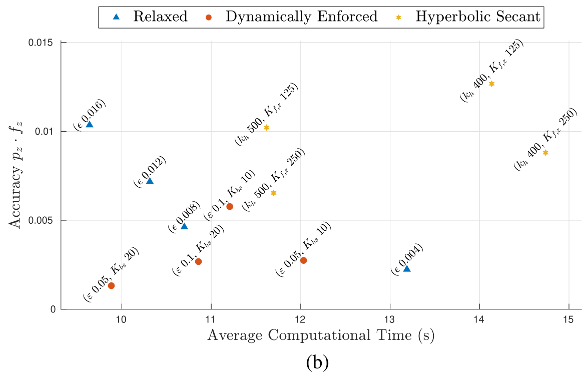


Fig. 11. Performance of the different complementarity methods in case of parameter change with respect to the setup of Section V. The closer to the origin, the better is the performance. (a) Test running on a laptop using the `MA57` linear solver. (b) Test running on `Github Actions` using the `MUMPS` linear solver.

used in Section V, but the average computational time tends to increase considerably when changing the parameters.

Similarly to Fig. 8(b), when running the test on `Github Actions`, the *hyperbolic secant* method points get shifted toward right, suggesting a higher performance loss compared to the other methods. Nonetheless, also in this case, the *dynamically enforced* method seems to provide the best trade-off between accuracy and computational time.

VII. Conclusions

This article showed that walking trajectories can emerge by specifying a moving reference for the contact points' centroid and the desired CoM velocity only. The planner considered relatively large time-steps. This enabled the insertion of another control loop at higher frequency, whose goal was to stabilize the planned trajectories. The main bottleneck was represented by the computational time. A single planner run may take from slightly less than a second to minutes, also according to the machine used. This prevents an online implementation on the real robot. Nevertheless, the continuous formulation of the problem allowed the application of techniques which solved the problem through the iterative application of fast LQR solvers [44], [72].

Among the different complementarity methods, the *dynamically enforced* complementarity method proved to be the best performing, both in a toy problem and when using the presented nonlinear trajectory planning framework. In particular, it is the most accurate and consistent in case of parameter variations. The *hyperbolic secant* method provided the lowest average computational time in generating walking trajectories, but this result is sensitive to parameter variations.

Finally, given the nonconvex nature of the problem defined in Section IV-H, it is fundamental to provide a meaningful initial guess. Indeed, local minima may bring the planner to "procrastinate," as anticipated in Section IV-I. An interesting future work consists in adopting reinforcement learning techniques, like [73], [74], to warm-start the optimization problem. In addition, the definition of references can affect the time necessary to find a solution. In future works, we will investigate both the adoption of faster solvers and the definition of a more sophisticated and efficient way of providing references.

VIII. Code and Multimedia Material

The code of this project is available at the link.[6] The definition of the `Github Actions` used for the comparisons of Section VI and the instructions on how to reproduce the results are available at the link.[7] A video describing the paper is available at the link https://youtu.be/Uc9o8TE32cw.

References

[1] K. Harada, E. Yoshida, and K. Yokoi, *Motion Planning for Humanoid Robots*. Berlin, Germany: Springer Science & Business Media, 2010.
[2] DRC-Teams, "What happened at the DARPA robotics challenge?" 2015. [Online]. Available: www.cs.cmu.edu/cga/drc/events
[3] S. Feng, E. Whitman, X. Xinjilefu, and C. G. Atkeson, "Optimization-based full body control for the DARPA robotics challenge," *J. Field Robot.*, vol. 32, no. 2, pp. 293–312, 2015.
[4] J. Carpentier et al., "Multi-contact locomotion of legged robots in complex environments-the loco 3d project," in *Proc. RSS Workshop Challenges Dynamic Legged Locomotion*, 2017, p. 3.
[5] G. Romualdi et al., "A benchmarking of DCM-based architectures for position, velocity and torque-controlled humanoid robots," *Int. J. Humanoid Robot.*, vol. 17, no. 1, 2020, Art. no. 1950034.
[6] D. Mayne, J. Rawlings, C. Rao, and P. Scokaert, "Constrained model predictive control: Stability and optimality," *Automatica*, vol. 36, no. 6, pp. 789–814, 2000.
[7] D. E. Orin and A. Goswami, "Centroidal momentum matrix of a humanoid robot: Structure and properties," in *Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst.*, 2008, pp. 653–659.
[8] G. Romualdi, S. Dafarra, G. L'Erario, I. Sorrentino, S. Traversaro, and D. Pucci, "Online non-linear centroidal MPC for humanoid robot locomotion with step adjustment," in *Proc. IEEE Int. Conf. Robot. Automat.*, 2022.
[9] S. Caron and A. Kheddar, "Multi-contact walking pattern generation based on model preview control of 3D COM accelerations," in *Proc. IEEE-RAS 16th Int. Conf. Humanoid Robots*, 2016, pp. 550–557.
[10] H. Dai and R. Tedrake, "Planning robust walking motion on uneven terrain via convex optimization," in *Proc. IEEE-RAS 16th Int. Conf. Humanoid Robots*, 2016, pp. 579–586.
[11] P. Fernbach, S. Tonneau, O. Stasse, J. Carpentier, and M. Taïx, "C-Croc: Continuous and convex resolution of centroidal dynamic trajectories for legged robots in multi-contact scenarios," *IEEE Trans. Robot.*, vol. 36, no. 3, pp. 676–691, Jun. 2020.
[12] A. Herzog, N. Rotella, S. Schaal, and L. Righetti, "Trajectory generation for multi-contact momentum control," in *Proc. IEEE-RAS 15th Int. Conf. Humanoid Robots*, 2015, pp. 874–880.
[13] M. Kudruss et al., "Optimal control for whole-body motion generation using center-of-mass dynamics for predefined multi-contact configurations," in *Proc. IEEE-RAS 15th Int. Conf. Humanoid Robots*, 2015, pp. 684–689.

[6][Online]. Available: https://github.com/ami-iit/dynamical-planner.
[7][Online]. Available: https://github.com/ami-iit/paper_dafarra_2022_tro_dcc-planner.

[14] M. Posa, S. Kuindersma, and R. Tedrake, "Optimization and stabilization of trajectories for constrained dynamical systems," in *Proc. IEEE Int. Conf. Robot. Automat.*, 2016, pp. 1366–1373.

[15] D. Serra, C. Brasseur, A. Sherikov, D. Dimitrov, and P.-B. Wieber, "A newton method with always feasible iterates for nonlinear model predictive control of walking in a multi-contact situation," in *Proc. IEEE-RAS 16th Int. Conf. Humanoid Robots.*, 2016, pp. 932–937.

[16] P. Fernbach, S. Tonneau, and M. Taïx, "Croc: Convex resolution of centroidal dynamics trajectories to provide a feasibility criterion for the multi contact planning problem," in *Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst.*, 2018, pp. 1–9.

[17] S. Feng, X. Xinjilefu, W. Huang, and C. G. Atkeson, "3D walking based on online optimization," in *Proc. 13th IEEE-RAS Int. Conf. Humanoid Robots*, 2013, pp. 21–27.

[18] R. Budhiraja, J. Carpentier, C. Mastalli, and N. Mansard, "Differential dynamic programming for multi-phase rigid contact dynamics," in *Proc. 2018 IEEE-RAS 18th Int. Conf. Humanoid Robots*, 2018, pp. 1–9.

[19] K. Giraud et al., "Motion planning with multi-contact and visual servoing on humanoid robots," in *Proc. Int. Symp. System Integration*, 2020, pp. 156–163.

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

[21] E. Dantec et al., "Whole body model predictive control with memory of motion: Experiments on a torque-controlled talos," in *Proc. IEEE Int. Conf. Robot. Automat.*, 2021, pp. 8202–8208.

[22] J. Carpentier, S. Tonneau, M. Naveau, O. Stasse, and N. Mansard, "A versatile and efficient pattern generator for generalized legged locomotion," in *Proc. IEEE Int. Conf. Robot. Automat.*, 2016, pp. 3555–3561.

[23] S. Caron and Q.-C. Pham, "When to make a step? Tackling the timing problem in multi-contact locomotion by TOPP-MPC," in *Proc. IEEE-RAS 17th Int. Conf. Humanoid Robot.*, 2017, pp. 522–528.

[24] A. W. Winkler, D. C. Bellicoso, M. Hutter, and J. Buchli, "Gait and trajectory optimization for legged systems through phase-based end-effector parameterization," *IEEE Robot. Automat. Lett.*, vol. 3, pp. 1560–1567, Jul. 2018.

[25] R. Deits and R. Tedrake, "Footstep planning on uneven terrain with mixed-integer convex optimization," in *Proc. 14th IEEE-RAS Int. Conf. Humanoid Robots*, 2014, pp. 279–286.

[26] S. Mason, N. Rotella, S. Schaal, and L. Righetti, "An MPC walking framework with external contact forces," in *Proc. IEEE Int. Conf. Robot. Automat.*, 2018, pp. 1785–1790.

[27] R. Mirjalili, A. Yousefi-Korna, F. A. Shirazi, A. Nikkhah, F. Nazemi, and M. Khadiv, "A whole-body model predictive control scheme including external contact forces and com height variations," in *Proc. IEEE-RAS 18th Int. Conf. Humanoid Robots*, 2018, pp. 1–6.

[28] A. Ibanez, P. Bidaud, and V. Padois, "Emergence of humanoid walking behaviors from mixed-integer model predictive control," in *Proc. IEEE RSJ Int. Conf. Intell. Robots Systems.*, 2014, pp. 4014–4021.

[29] B. Aceituno-Cabezas et al., "Simultaneous contact, gait, and motion planning for robust multilegged locomotion via mixed-integer convex optimization," *IEEE Robot. Automat. Lett.*, vol. 3, no. 3, pp. 2531–2538, Jul. 2018.

[30] E. Drumwright and D. A. Shell, "Modeling contact friction and joint friction in dynamic robotic simulation using the principle of maximum dissipation," in *Algorithmic Foundations of Robotics IX*. Berlin, Germany: Springer, 2010, pp. 249–266.

[31] E. Todorov, "A convex, smooth and invertible contact model for trajectory optimization," in *Proc. IEEE Int. Conf. Robot. Automat.*, 2011, pp. 1071–1076.

[32] I. Mordatch, E. Todorov, and Z. Popović, "Discovery of complex behaviors through contact-invariant optimization," *ACM Trans. Graph.*, vol. 31, no. 4, pp. 1–8, 2012.

[33] Y. Tassa, T. Erez, and E. Todorov, "Synthesis and stabilization of complex behaviors through online trajectory optimization," in *Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst.*, 2012, pp. 4906–4913.

[34] T. Erez, K. Lowrey, Y. Tassa, V. Kumar, S. Kolev, and E. Todorov, "An integrated system for real-time model predictive control of humanoid robots," in *Proc. 13th IEEE-RAS Int. Conf. Humanoid Robots*, 2013, pp. 292–299.

[35] S. Kajita, F. Kanehiro, K. Kaneko, K. Yokoi, and H. Hirukawa, "The 3D linear inverted pendulum mode: A simple modeling for a biped walking pattern generation," in *Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst.*, 2001, pp. 239–246.

[36] S. Kajita, F. Kanehiro, K. Kaneko, K. Yokoi, and H. Hirukawa, "The 3D linear inverted pendulum mode: A simple modeling for a biped walking pattern generation," in *Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst. Expanding Societal Role Robot. Next Millennium*, 2001, pp. 239–246.

[37] J. Pratt, J. Carff, S. Drakunov, and A. Goswami, "Capture point: A step toward humanoid push recovery," in *Proc. 6th IEEE-RAS Int. Conf. Humanoid Robots*, 2006, pp. 200–207.

[38] T. Takenaka, T. Matsumoto, and T. Yoshiike, "Real time motion generation and control for biped robot-1 st report: Walking gait pattern generation," in *Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst.*, 2009, pp. 1084–1091.

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

[40] M. Missura and S. Behnke, "Balanced walking with capture steps," in *Robot Soccer World Cup*. Berlin, Germany: Springer, 2014, pp. 3–15.

[41] M. Naveau, M. Kudruss, O. Stasse, C. Kirches, K. Mombaur, and P. Souères, "A reactive walking pattern generator based on nonlinear model predictive control," *IEEE Robot. Automat. Lett.*, vol. 2, no. 1, pp. 10–17, Jan. 2017.

[42] R. J. Griffin and A. Leonessa, "Model predictive control for dynamic footstep adjustment using the divergent component of motion," in *Proc. IEEE Int. Conf. Robot. Automat.*, 2016, pp. 1763–1768.

[43] H.-M. Joe and J.-H. Oh, "Balance recovery through model predictive control based on capture point dynamics for biped walking robot," *Robot. Auton. Syst.*, vol. 105, pp. 1–10, 2018.

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

[45] G. Romualdi, S. Dafarra, and D. Pucci, "Modeling of visco-elastic environments for humanoid robot motion control," *IEEE Robot. Automat. Lett.*, vol. 6, no. 3, pp. 4289–4296, Jul. 2021.

[46] B. Ponton, M. Khadiv, A. Meduri, and L. Righetti, "Efficient multicontact pattern generation with sequential convex approximations of the centroidal dynamics," *IEEE Trans. Robot.*, vol. 37, no. 5, pp. 1661–1679, Oct. 2021.

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

[48] M. Posa, C. Cantu, and R. Tedrake, "A direct method for trajectory optimization of rigid bodies through contact," *Int. J. Robot. Res.*, vol. 33, no. 1, pp. 69–81, 2014.

[49] S. Dafarra, G. Romualdi, G. Metta, and D. Pucci, "Whole-body walking generation using contact parametrization: A non-linear trajectory optimization approach," in *Proc. IEEE Int. Conf. Robot. Automat.*, 2020, pp. 1511–1517.

[50] J. E. Marsden and T. S. Ratiu, *Introduction to Mechanics and Symmetry: A Basic Exposition of Classical Mechanical Systems*. New York, NY, USA: Springer, 2010.

[51] S. Traversaro, "Modelling, estimation and identification of humanoid robots dynamics," Ph.D. dissertation, Univ. of Genoa, Apr. 2017. [Online]. Available: https://doi.org/10.5281/zenodo.3564796

[52] 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, 2013.

[53] P. M. Wensing and D. E. Orin, "Improved computation of the humanoid centroidal dynamics and application for whole-body control," *Int. J. Humanoid Robot.*, vol. 13, no. 01, 2016, Art. no. 1550039.

[54] P. M. Wensing and D. E. Orin, "Generation of dynamic humanoid behaviors through task-space control with conic optimization," in *Proc. IEEE Int. Conf. Robot. Automat.*, 2013, pp. 3103–3109.

[55] M. Azad and R. Featherstone, "A new nonlinear model of contact normal force," *IEEE Trans. Robot.*, vol. 30, no. 3, pp. 736–739, Jun. 2014.

[56] M. Liu and V. Padois, "Reactive whole-body control for humanoid balancing on non-rigid unilateral contacts," in *Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst.*, 2015, pp. 3981–3987.

[57] S. Caron, Q.-C. Pham, and Y. Nakamura, "Leveraging cone double description for multi-contact stability of humanoids with applications to statics and dynamics," *Robot.: Sci. Syst.*, vol. 11, pp. 1–9, 2015.

[58] J. T. Betts, *Practical Methods for Optimal Control and Estimation Using Nonlinear Programming*, Philadelphia USA: SIAM, 2010, vol. 19.

[59] J. J. Moreau, "Unilateral contact and dry friction in finite freedom dynamics," in *Nonsmooth Mechanics and Applications*. Berlin, Germany: Springer, 1988, pp. 1–82.

[60] D. Stewart and J. C. Trinkle, "An implicit time-stepping scheme for rigid body dynamics with coulomb friction," in *Proc. Millennium Conf. IEEE Int. Conf. Robot. Automat. Symp.*, 2000, pp. 162–169.

[61] J. Baumgarte, "Stabilization of constraints and integrals of motion in dynamical systems," *Comput. Methods Appl. Mechanics Eng.*, vol. 1, no. 1, pp. 1–16, 1972.

[62] H. Khalil, *Nonlinear Systems*, ser. Pearson Education. Hoboken, NJ, USA: Prentice Hall, 2002. [Online]. Available: https://books.google.it/books?id=t_d1QgAACAAJ

[63] B. Graf, "Quaternions and dynamics," 2008. [Online]. Available: https://arxiv.org/pdf/0811.2889.pdf

[64] B. Ravani and B. Roth, "Motion synthesis using kinematic mappings," *J. Mechanisms, Transmissions, Automat. Des.*, vol. 105, no. 3, pp. 460–467, Sep. 1983. [Online]. Available: https://doi.org/10.1115/1.3267382

[65] D. Mayne and H. Michalska, "Receding horizon control of nonlinear systems," *IEEE Trans. Autom. Control*, vol. 35, no. 7, pp. 814–824, Jul. 1990.

[66] L. Natale, C. Bartolozzi, D. Pucci, A. Wykowska, and G. Metta, "ICUB: The not-yet-finished story of building a robot child," *Sci. Robot.*, vol. 2, no. 13, 2017, Art. no. eaaq1026.

[67] A. Wächter and L. T. Biegler, "On the implementation of an interior-point filter line-search algorithm for large-scale nonlinear programming," *Math. Program.*, vol. 106, no. 1, pp. 25–57, 2006.

[68] A. HSL, "Collection of Fortran codes for large-scale scientific computation," 2007. [Online]. Available: http://www.hsl.rl.ac.uk

[69] S. Dafarra, "Predictive whole-body control of humanoid robot locomotion," Ph.D. dissertation, Università degli studi di Genova, Istituto Italiano di Tecnologia, 2020. [Online]. Available: https://iris.unige.it//handle/11567/1004529

[70] P. R. Amestoy, I. S. Duff, J.-Y. L'Excellent, and J. Koster, "Mumps: A general purpose distributed memory sparse solver," in *Proc. Int. Workshop Appl. Parallel Comput.*, 2000, pp. 121–130.

[71] J. A. Andersson, J. Gillis, G. Horn, J. B. Rawlings, and M. Diehl, "Casadi: A software framework for nonlinear optimization and optimal control," *Math. Program. Computation*, vol. 11, no. 1, pp. 1–36, 2019.

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

[73] X. B. Peng, P. Abbeel, S. Levine, and M. van de Panne, "Deepmimic: Example-guided deep reinforcement learning of physics-based character skills," *ACM Trans. Graph.*, vol. 37, no. 4, pp. 1–14, 2018.

[74] P. M. Viceconte et al., "Adherent: Learning human-like trajectory generators for whole-body control of humanoid robots," *IEEE Robot. Automat. Lett.*, vol. 7, no. 2, pp. 2779–2786, Apr. 2022.

Stefano Dafarra (Member, IEEE) received the M.S. degree in automation and control engineering from Politecnico di Milano, Milan, Italy, in 2016, and the Ph.D degree in advanced and humanoid robotics from the University of Genoa, Genoa, Italy, in 2020.

He is currently a Postdoctoral Researcher with Istituto Italiano di Tecnologia, Genoa, Italy. His research interests include optimization, optimal control, and humanoid locomotion.

Giulio Romualdi (Graduate Student Member, IEEE) received the B.S. degree in biomedical engineering and the M.S. degree in robotics and automation engineering from the University of Pisa, Pisa, Italy, in 2014 and 2018, respectively. He is currently working toward the Ph.D. degree in online control of humanoid robot locomotion with the University of Genova, Genoa, Italy and Istituto Italiano di Tecnologia, Genoa, Italy.

His research interests include bipedal locomotion, humanoid robots, and nonlinear control theory.

Daniele Pucci (Associate Member, IEEE) received the Bachelor's and Master's degrees in control engineering (with highest honors) from "Sapienza" University of Rome, Rome, Italy, in 2007 and 2009, respectively, and the Ph.D. degree in control engineering from INRIA Sophia Antipolis, Valbonne, France and "Sapienza" University of Rome, in 2013.

From 2013 to 2017, he was a Postdoctoral Researcher with the Istituto Italiano di Tecnologia (IIT), working within the EU project CoDyCo. From August 2017 to August 2021, he was the Head of the Dynamic Interaction Control Lab, where the research focus was on the humanoid robot locomotion and physical interaction problem, with specific attention on the control and planning of the associated nonlinear systems. In 2020, in the context of the split site Ph.D. supervision program, he was a Visiting Lecturer with the University of Manchester. Since September 2021, he has been the PI, leading the artificial and mechanical intelligence research line with IIT. He has been the Scientific PI of the H2020 European Project AnDy, Task Leader of the H2020 European Project SoftManBot, and the Coordinator of the Joint Laboratory between IIT and Honda JP.

Dr. Pucci has received the "Excellence Path Award" from Sapienza, in 2009, and the Innovator of the Year Award Under 35 Europe from the *MIT Technology Review* magazine, in 2019. In July 2020, he was selected as a member of the Global Partnership on Artificial Intelligence (GPAI).