Capturability-based Pattern Generation for Walking with Variable Height
Stéphane Caron, Adrien Escande, Leonardo Lanari, Bastien Mallein

To cite this version:
Stéphane Caron, Adrien Escande, Leonardo Lanari, Bastien Mallein. Capturability-based Pattern Generation for Walking with Variable Height. IEEE Transactions on Robotics, IEEE, In press, 10.1109/TRO.2019.2923971. hal-01689331v3

HAL Id: hal-01689331
https://hal.archives-ouvertes.fr/hal-01689331v3
Submitted on 11 Apr 2019

HAL is a multi-disciplinary open access archive for the deposit and dissemination of scientific research documents, whether they are published or not. The documents may come from teaching and research institutions in France or abroad, or from public or private research centers.

L’archive ouverte pluridisciplinaire HAL, est destinée au dépôt et à la diffusion de documents scientifiques de niveau recherche, publiés ou non, émanant des établissements d’enseignement et de recherche français ou étrangers, des laboratoires publics ou privés.

Distributed under a Creative Commons Attribution - ShareAlike 4.0 International License
Capturability-based Pattern Generation for Walking with Variable Height
Stéphane Caron, Adrien Escande, Leonardo Lanari, and Bastien Mallein

Abstract—Capturability analysis of the linear inverted pendulum (LIP) model enabled walking with constrained height based on the capture point. We generalize this analysis to the variable-height inverted pendulum (VHIP) and show how it enables 3D walking over uneven terrains based on capture inputs. Thanks to a tailored optimization scheme, we can compute these inputs fast enough for real-time model predictive control. We implement this approach as open-source software and demonstrate it in dynamic simulations.

Index Terms—Bipedal walking, Capturability, Uneven terrain

I. INTRODUCTION

Capturability quantifies the ability of a system to come to a stop. For a humanoid walking in the linear inverted pendulum (LIP) mode, it is embodied by the capture point, the point on the ground where the robot should step in order to bring itself to a stop [1]. In recent years, one of the main lines of research in LIP-based studies has explored the question of walking by feedforward planning and feedback control of the capture point [2], [3], [4], [5], [6], [7].

The LIP owes its tractability to two assumptions: no angular-momentum variation around the center of mass (CoM), and a holonomic constraint on the CoM height. As a consequence of the latter, a majority of LIP-based walking controllers assume a flat terrain. Removing this holonomic constraint from the LIP leads to the variable-height inverted pendulum (VHIP) model, for which our understanding is at an earlier stage. Previous studies [8], [9], [10] focused on its balance control for planar motions (sagittal and vertical only). In a preliminary version of this work [11], we extended the analysis from 2D to 3D balance control. In the present work, we bridge the gap from balancing to walking.

Our contribution is three-fold. First, we provide a necessary and sufficient condition for the capturability of the VHIP model (Section II). Second, we show how to turn this condition into an optimization problem (Section III) for which we develop a tailored optimization scheme (Section IV). Finally, we adapt this optimization into a model-predictive walking pattern generator for rough terrains (Section V) that we demonstrate in dynamic simulations (Section VI).

II. CAPTURABILITY OF INVERTED PENDULUM MODELS

The critical part of the dynamics for a walking biped lies in the Newton-Euler equation that drives its unactuated floating-base coordinates:

\[
\begin{bmatrix}
\dot c_L \\
\dot c_r \\
\end{bmatrix} = \begin{bmatrix}
\frac{1}{m} f \\
b_c \\
0
\end{bmatrix} + 
\begin{bmatrix}
g \\
0
\end{bmatrix}
\] (1)

where \( c \) is the position of the center of mass (CoM) of the robot, \( g \) is the gravity vector (also written \( g = -ge_z \) with \( g \) the gravitational constant), \( m \) is the total robot mass and \( L_c \) is the angular momentum around \( c \). The net contact wrench \((f, \tau_c)\) consists of the resultant \( f \) of external contact forces and their moment \( \tau_c \) around the CoM.

A. Inverted pendulum models

The linear inverted pendulum (LIP) [12] model is based on two constraints: a constant angular momentum around the center of mass, and a constant CoM height with respect to a reference plane:

\[
\begin{align*}
\dot L_c &= 0 \quad (2) \\
n \cdot (c - o) &= h \quad (3)
\end{align*}
\]

where \( n \), \( o \) and \( h \) are respectively the normal vector, reference point and reference height that define the CoM motion plane. As a consequence of these two assumptions, the Newton-Euler equation (1) simplifies\(^1\) to:

\[
\dot c = \omega_{LIP}^2 (c - r) + g
\] (4)

where \( \omega_{LIP} := \sqrt{g/h} \) is a constant and \( r \) is the center of pressure (CoP), or zero-tilting moment point (ZMP) [14] when there are multiple contacts. A strong limitation of the LIP is the holonomic constraint (3) on the CoM, which can be kinematically problematic in scenarios such as stair climbing. One line of research sought to overcome this by constraining the CoM to parametric or piecewise-planar surfaces [15], [16], but the next question of deciding such surfaces based on terrain topology has not attracted a lot of attention so far.

The variable-height inverted pendulum (VHIP) model strips away the holonomic CoM constraint altogether [17], [18]. Its equation of motion is:

\[
\dot c = \lambda (c - r) + g
\] (5)

\(^1\) See e.g. [13] for a reminder of the steps of this derivation.
Center of Pressure $r$
Center of Mass Acceleration $\dot{e}$
Normalized Stiffness $\lambda$
Center of Pressure $r$
Center of Mass Acceleration $\dot{e}$
Normalized Stiffness $\lambda$

Fig. 1. Inputs and output of the linear inverted pendulum (LIP) and variable-height inverted pendulum (VHIP) models. In the former, the stiffness coefficient is fixed to a constant $\omega_L^2$, while it becomes an additional input $\lambda$ of the latter.

where $\lambda > 0$ is now a time-varying stiffness coefficient. The two control inputs of this system are the center of pressure $r$ and the stiffness $\lambda$, as shown in Figure 1.

B. Feasibility conditions

To be feasible, the CoP $r$ must belong to the contact area $C$ under the supporting foot. This area is also time-varying but changes only discretely. The transition from one support contact to the next is called a contact switch. A trajectory with $N$ contact switches is called an $N$-step trajectory.

We assume that all contact areas are planar and polygonal. Let us denote by $o$ the center of the area $C$ and by $n$ its normal (such that $n \cdot e_z \neq 0$). The CoP $r$ belongs to the plane of contact if and only if $(r - o) \cdot n = 0$. The height of the CoM $c$ above the contact area is the algebraic distance $h(c)$ such that $c - h(c)e_z$ belongs to the contact plane:

$$h(c) := \frac{(c - o) \cdot n}{(e_z \cdot n)} \quad (6)$$

Note how, when walking on a horizontal floor, $e_z$ and $n$ are aligned and $h$ is simply the $z$ coordinate of the center of mass.

To be feasible, the stiffness $\lambda$ must be non-negative by unilaterality of contact. We furthermore impose that $\lambda \in [\lambda_{\text{min}}, \lambda_{\text{max}}]$ so that contact pressure is never exactly zero and remains bounded. Note that we do not model Coulomb friction conditions here: having found in a previous work that CoP remains bounded. Note that we do not model Coulomb friction $\lambda$ unilaterality of contact. We furthermore impose that $h(c)$ is simply the $z$ coordinate of the center of mass.

To be feasible, the stiffness $\lambda$ must be non-negative by unilaterality of contact. We furthermore impose that $\lambda \in [\lambda_{\text{min}}, \lambda_{\text{max}}]$ so that contact pressure is never exactly zero and remains bounded. Note that we do not model Coulomb friction conditions here: having found in a previous work that CoP remains bounded. Note that we do not model Coulomb friction $\lambda$ unilaterality of contact. We furthermore impose that $h(c)$ is simply the $z$ coordinate of the center of mass.

C. Capture inputs and capture trajectories

A natural choice of the pendulum state consists of its CoM position and velocity $x = (c, \dot{c})$.

**Definition 1** (Static equilibrium). A state $x = (c, \dot{c})$ is a static equilibrium when its velocity $\dot{c}$ is zero and can remain zero with suitable constant controls $(\lambda t, r t)$.

2 All quantities being normalized by mass, we call $\lambda$ a stiffness although its unit is $s^{-2}$ and not kg.s$^{-2}$. Similarly, we will refer to frequencies (unit: s$^{-1}$) as dampings.

Static equilibria, also called *capture states* [1], are the targets of capturability analysis, the desired locations that the CoM should converge to. A static equilibrium is characterized by its CoM position $c_f$ and the contact $C_f$ upon which it rests at height $h_f = h(c_f)$. The only control input $\lambda t, r t$ that maintains the pendulum in static equilibrium is such that $c_f = r t - g / \lambda t$, that is:

$$\lambda t(c_f) = \frac{g}{h_f} \quad r t(c_f) = c_f - h_f e_z \quad (7)$$

Given an $N$-step contact sequence, we say that a state $x_i$ is $(N$-step) *capturable* when there exists a feasible input function $\lambda(t), r(t)$ such that applying Equation (5) from $x_i$ brings the system asymptotically to an equilibrium $x_f$. We call such functions *capture inputs* of the capturable state $x_i$, and denote their set by $I_{x_i,x_f}$. We call *capture trajectory* the CoM trajectory $c(t)$ resulting from a capture input. In what follows, we use the subscript $\square t$ to denote the “initial” or instantaneous state of the system, and the subscript $\square t$ for its “final” or asymptotic state.

By definition, $x_i$ is capturable if and only if there exists $x_f$ such that $I_{x_i,x_f} \neq \emptyset$. The set $I_{x_i,x_f}$ contains however several solutions, including mathematical oddities such as functions with ever-increasing frequencies. In what follows, we restrict it to inputs that converge asymptotically:

$$I_{x_i,x_f} = \left\{ (\lambda(t), r(t)) \in I_{x_i,x_f} : \lim_{t \to \infty} \lambda(t) = \lambda_f \lim_{t \to \infty} r(t) = r_f \right\} \quad (8)$$

Property 5 shows that this choice does not cause any loss of generality (see Appendix A).

D. Dichotomy of the components of motion

One of the main findings in the control of the LIP model is to focus on its divergent component of motion (DCM), the capture point. Among other benefits, controlling only the capture point reduces second-order dynamics to first order and maximizes the basin of attraction of feedback controllers [2], [3], [4]. An important step in the analysis of the VHIP model is therefore to identify its DCM. Interestingly, the answer can be found in a study of motorcycle balance [20], which we recall here.

The equation of motion (5) of the VHIP can be interpreted as either nonlinear or linear time-variant, depending on whether one focuses respectively on feedback or open-loop control. We focus on the latter for walking pattern generation. Let us then rewrite this equation as a first-order linear time-variant system:

$$\dot{c} = [0 \ I] c + g - \lambda r \quad (9)$$

where $I$ is the $3 \times 3$ identity matrix. This equation has the form $\dot{x} = A(t)x + b(t)$ where the system matrix $A$ depends on the stiffness input $\lambda$, while the forcing term $b$ varies with both inputs $\lambda$ and $r$.

Hauser et al. showed [20] how to obtain an exponential dichotomy [21] of the state $x$ (that is, how to decompose it into
convergent and divergent components) by applying a change of coordinates \( x = Sz \) with:

\[
S = \frac{1}{\gamma + \omega} \begin{bmatrix} I & I \\ -\omega I & \gamma I \end{bmatrix} \Leftrightarrow S^{-1} = \begin{bmatrix} \gamma I & -I \\ \omega I & +I \end{bmatrix} \tag{10}
\]

The two functions \( \gamma(t) \) and \( \omega(t) \) are positive and of class \( C^1 \). We will refer to them as dampings in accordance with their (normalized) physical unit. The new state vector \( z = S^{-1}x \) then consists of two components \( \zeta \) and \( \xi \) defined by:

\[
\zeta = \gamma c - \dot{c} \tag{11}
\]
\[
\xi = \omega c + \dot{c} \tag{12}
\]

They respectively correspond to the convergent and divergent component of motion (DCM). In the case of a linear inverted pendulum with constant \( \omega_{LIP} \), the DCM \( \xi \) is simply proportional to the capture point \( c + \dot{c}/\omega_{LIP} \).

Collectively, the state vector \( z \) is subject to:

\[
\dot{z} = \tilde{A}z + \tilde{b} \tag{13}
\]
\[
\tilde{A} = S^{-1}(AS - \dot{S}) \tag{14}
\]
\[
\tilde{b} = S^{-1}b \tag{15}
\]

The calculation of \( \tilde{b} \) is straightforward. That of \( \tilde{A} \) yields:

\[
\tilde{A} = \frac{1}{\gamma + \omega} \begin{bmatrix} (\bar{\gamma} - \gamma\omega - \lambda)I & (\bar{\gamma} + \gamma^2 - \lambda)I \\ (\bar{\omega} - \omega^2 + \lambda)I & (\bar{\omega} + \omega\gamma + \lambda)I \end{bmatrix} \tag{16}
\]

To decouple the system, we can eliminate non-diagonal terms in this state matrix by imposing the two following Riccati equations:

\[
\bar{\gamma} = \lambda - \gamma^2 \tag{17}
\]
\[
\bar{\omega} = \omega^2 - \lambda \tag{18}
\]

This results in the following state dynamics:

\[
\dot{z} = \begin{bmatrix} \bar{\gamma} & 0 \\ 0 & \bar{\omega} \end{bmatrix} \begin{bmatrix} \zeta \\ \xi \end{bmatrix} + \begin{bmatrix} \lambda r - g \\ g - \lambda r \end{bmatrix} \tag{19}
\]

The linear time-varying system has thus been decoupled into two linearly independent components \( \zeta \) and \( \xi \) that evolve according to their own dynamics, provided that there exists two \( C^1 \) positive finite solutions to (17) and (18). A proof of this and a detailed analysis of damping functions are given in Appendix A-B.

Although we defer the detailed analysis of damping functions to this Appendix, its takeaway point is that \( \gamma \) and \( \omega \) are, in themselves, convergent and divergent components. A parallel can be drawn between the DCM–CoP and \( \omega - \lambda \) systems:

- In the LIP, with the CoP restricted to a support area, the CoP is a repulsor of the DCM, and the DCM is controllable if and only if it is above the support area [3].
- In the VHIP, with \( \lambda \) restricted to \( [\lambda_{\min}, \lambda_{\max}] \), \( \lambda \) is a repulsor of \( \omega \), and \( \omega \) is controllable if and only if it belongs to \( [\sqrt{\lambda_{\min}}, \sqrt{\lambda_{\max}}] \) (Property 6).

This remark is central to the walking pattern generation method in Section III, which reduces three-dimensional capturability to one dimension.

It can be shown that, regardless of the initial state \( x_i \) of the system, any input function from \( T_{x_i} \) makes \( \zeta \) converge as well (Property 9), owing it its name of convergent component of motion. From a control perspective, spending additional inputs to control this component is not necessary and can even be wasteful.\(^3\) The main concern of capturability analysis is therefore to prevent the other component \( \xi \) from diverging.

E. Boundedness condition

The divergent component of motion \( \xi \) corresponding to the damping \( \omega \) is subject to the differential equation:

\[
\dot{\xi} = \omega \xi + g - \lambda r \tag{20}
\]

The general solution to this equation is given by:

\[
\xi(t) = \left( \xi(0) + \int_0^t e^{-\Omega(t)}(g - \lambda r)(\tau)\,d\tau \right) e^{\Omega(t)} \tag{21}
\]

where \( \Omega(t) = \int_0^t \omega(\tau)\,d\tau \). In our working assumptions, this integral is well-defined and finite (details in Appendix A-D).

Set aside the particular condition that we are about to discuss, the function \( \xi(t) \) diverges as \( t \to \infty \), giving \( \xi \) its name of divergent component of motion (DCM) \([4]\).\(^4\) However, a careful match between future capture inputs and the initial condition \( \xi \) can guarantee that \( \xi(t) \) converges as well. This choice is known as the boundedness condition \([22]\):

Property 1 (Boundedness condition). Consider an input function \( \lambda(t), r(t) \) such that \( \lim_{t \to \infty} \lambda(t) = \lambda_f \) and \( \lim_{t \to \infty} r(t) = r_f \). Then, there exists a unique \( \xi_i = \omega_i c_i + \dot{c}_i \) such that the solution \( \xi \) of (20) with \( \xi(0) = \xi_i \) remains finite at all times. This initial condition is given by:

\[
\xi_i = \int_0^\infty e^{-\Omega(t)}(\lambda(t)r(t) - g)\,dt \tag{22}
\]

where \( \omega_i \) is the initial value of the unique bounded solution \( \omega \) to the Riccati equation \( \dot{\omega} = \omega^2 - \lambda \) and \( \Omega \) is the antiderivative of \( \omega \) such that \( \Omega(0) = 0 \).

The proof of this property is given in Appendix A-D. In the familiar setting of the LIP where \( \lambda = \omega^2_{LIP} \) is a constant, taking a constant CoP \( r_c \) yields:

\[
\omega_{LIP} c_i + \dot{c}_i = \int_0^\infty (\omega_{LIP}^2 r_c - g) e^{-\omega_{LIP} t} \,dt \tag{23}
\]

\[
= \omega_{LIP} r_c - \frac{g}{\omega_{LIP}} \tag{24}
\]

Over horizontal coordinates, this equation implies that \( r_v = c_i + \dot{c}_i/\omega_{LIP} \), i.e. the CoP is located at the capture point. Over the \( z \) coordinate, it yields \( \omega_{LIP} = \sqrt{g/h} \), the known expression of the natural frequency of the LIP. Overall, the boundedness condition characterizes the capturability of the LIP. We will now conclude our capturability analysis by showing how this is also the case for the VHIP.

\(^3\) For the LIP, it reduces the basin of attraction of feedback controllers [3].

\(^4\) More specifically, our analysis considers a time-varying divergent component of motion [17]. While previous works such as [3], [6], [17] chose to write their DCMs as positions \( c(t) + \dot{c}(t)/\omega(t) \), we cast them as velocities here to simplify calculations (consider the derivative of a product \( uv \) compared to that of a ratio \( u/v \)). The formula of the DCM itself is not a crucial design choice, as we will discuss at the end of this Section.
F. Capturability of the variable-height inverted pendulum

What we have established so far is a necessary condition: if an input function belongs to $\mathcal{I}_{\mathfrak{x}_1, \mathfrak{x}_f}$, then it is feasible, converging and satisfies the boundedness condition. The key result of our capturability analysis is that this condition is also sufficient, i.e. it characterizes capture inputs:

Property 2. Let $\mathfrak{x}_1 = (c_1, \dot{c}_1)$ denote a capturable state and $\mathfrak{x}_f = (c_f, 0)$ a static equilibrium. Then, $t \mapsto \lambda(t), r(t)$ is a capture input from $\mathfrak{x}_1$ to $\mathfrak{x}_f$ if and only if:

(i) its values $\lambda(t)$ and $r(t)$ are feasible for all $t \geq 0$,
(ii) $\lim_{t \to \infty} \lambda(t) = \lambda_1(c_f)$ and $\lim_{t \to \infty} r(t) = r_1(c_f)$,
(iii) it satisfies the boundedness condition:

$$\int_0^{\infty} \left( \lambda(t) r(t) - g \right) e^{-\Omega(t)} dt = \omega_1 c_1 + \dot{c}_1$$

where $\omega_1$ is the initial value of the unique bounded solution $\omega$ to the Riccati equation $\dot{\omega} = \omega^2 - \lambda$ and $\Omega$ is the antiderivative of $\omega$ such that $\Omega(0) = 0$.

A proof of this property is given in Appendix A-E. In the remainder of this manuscript, we will see how this is not only a theoretical but also a practical result with applications to balance control and walking pattern generation.

A noteworthy methodological point here is that the expression of the divergent component of motion is not unique. Rather, a DCM is chosen by the roboticist. For example, in [11] we considered a different DCM $\xi := (c - r) + \dot{c} - \dot{r}$ yielding a boundedness condition written:

$$\int_0^{\infty} \left( \dot{\xi}(t) - g \right) e^{-\Omega(t)} dt = \omega_1 c_1 + \dot{c}_1$$

This condition is the same as (25), which can be seen by applying a double integration by parts. In the present work, we chose the DCM from Equation (20) as it makes calculations simpler. We preferred a velocity-based rather than position-based DCM for the same reason, as the differential of a product involves less operations than that of a ratio.

III. BALANCE CONTROL WITH VARIABLE HEIGHT

Let us consider first the problem of balance control, i.e. zero-step capturability. The robot pushes against a stationary contact area $\mathcal{C}$ in order to absorb the linear momentum of its initial state $\mathfrak{x}_1$, eventually reaching a static equilibrium $\mathfrak{x}_f$. This level of capturability enables push recovery [1], [23], [10], [24] up to post-impact fall recovery in worst-case scenarios [25], [26].

The gist of the method we propose thereafter is to reduce the three-dimensional capturability condition over $\lambda, r$ (Property 2) into a one-dimensional condition over $\lambda$. To do so, we couple the evolution of $\lambda$ and $r$ by a suitably-defined intermediate variable $s$. The complete pipeline goes as follows:

- Change variable from time $t$ to a new variable $s$
- Define the CoP evolution $\omega(s)$ as a function of $s$
- Reduce capturability to an optimization over $\omega(s)$
- Compute the optimal solution $\omega^*(s)$ of this problem
- Change variable from $s$ to time $t$

From $\omega^*(t)$, it is then straightforward to compute the full capture input $t \mapsto \lambda^*(t), r^*(t)$ as well as the capture trajectory $c^*(t)$. Figure 2 compares this pipeline with the traditional LIP feedback control strategy with time-invariant DCM. Let us now detail each step of this process.

A. Change of variable

Define the adimensional quantity:

$$s(t) = e^{-\Omega(t)}$$

This new variable ranges from $s = 1$ when $t = 0$ to $s \to 0$ when $t \to \infty$. Its time derivatives are:

$$\dot{s}(t) = -\omega(s) \dot{\omega}(t) \quad \ddot{s}(t) = \lambda(t) s(t)$$

Owing to the bijective mapping between $t$ and $s$, we can define $\omega, \gamma$ and $\lambda$ as functions of $s$ rather than as functions of $t$. This approach is e.g. common in time-optimal control [27]. Let us denote by $\frac{\partial}{\partial s}$ derivation with respect to $s$, as opposed to $\frac{\partial}{\partial t}$ for derivation with respect to $t$. The Riccati equation (18) of $\omega$ becomes:

$$\lambda = \omega^2 - \dot{\omega} - s \omega' = \omega (s \omega')' = \omega(s \omega)'$$

Injecting this expression into the time integral (25) of the boundedness condition yields:

$$\int_0^{\infty} \left( \lambda(t) r(t) - g \right) s(t) dt = \int_0^1 \left( \omega(s \omega)' r(s) - g \right) \frac{ds}{\omega}$$

We can then characterize capture inputs as functions of $s$:

Property 3. Let $\mathfrak{x}_1 = (c_1, \dot{c}_1)$ denote a capturable state and $\mathfrak{x}_f = (c_f, 0)$ a static equilibrium. Then, $s \mapsto \lambda(s), \omega(s)$ is a capture input from $\mathfrak{x}_1$ to $\mathfrak{x}_f$ if and only if:

(i) its values $\lambda(s)$ and $r(s)$ are feasible for all $s \in [0, 1]$
(ii) $\lim_{s \to 0} \lambda(s) = \lambda_1(c_f)$ and $\lim_{s \to 0} \omega(s) = \omega_1(c_f)$
(iii) it satisfies the boundedness condition:

$$\int_0^1 \omega(s \omega)' ds - g \int_0^1 \frac{ds}{\omega(s)} = \omega_1 c_1 + \dot{c}_1$$

where $\omega_1$ denotes the initial value (at $s = 1$) of the solution $\omega$ to the differential equation $\omega(s \omega)' = \lambda$. With this reformulation, the infinite-time integral has become finite over the $[0, 1]$ interval and the antiderivative $\Omega$ has been replaced by $\omega$ itself.
where \( f \) is a variable. In the example depicted in Figure 3, it will be chosen with zero velocity, but the instantaneous CoP moves along a line segment parameterized by \( \alpha \) parameterized by \( \alpha \):

\[
\int_0^1 r_{xy}(s) \omega(s) \, ds = \omega_1 c_{xy} + c_{xy}^2
\]

\[
\int_0^1 \frac{ds}{\omega(s)} = \frac{\omega_1 h_i + \dot{h}_i}{g}
\]

where \( h_i := h(c_i) \) is the initial CoM height and \( \dot{h}_i \) its velocity, both known from the initial state \( c_i \). The gravity term (33) only involves \( \omega(s) \), but the CoP term (32) involves both \( r(s) \) and \( \omega(s) \). We reduce it to an integral over \( \omega(s) \) by making the CoP move along a line segment from \( r_1 \) to \( r_1^c \):

\[
r(s) = r_1 + (r_1 - r_1^c) f(s \omega)
\]

where \( f \) can be any smooth function that satisfies:

- \( f(\omega_1) = 1 \): the CoP is initially located at \( r_1 \).
- \( f(0) = 0 \): the CoP converges towards \( r_1^c \).
- \( f \) is increasing: we exclude solutions where the CoP would move back and forth along the line segment.
- \( f \) is integrable: let \( F \) denote its antiderivative \( F(x) := \int_0^x f(y) \, dy \). It is positive by positivity of \( f \).

The final CoP location \( r_1^c \) is already known from the desired capture state \( x_1 \), where the CoM is at the vertical above contact with zero velocity, but the instantaneous CoP \( r_1 \) is a decision variable. In the example depicted in Figure 3, it will be chosen on the other side of the line \( c_i + R \hat{e}_i \) compared to \( r_1 \) in order to progressively reorient \( \hat{c}(t) \) toward the capture state \( r_1^c \), similarly to the behavior observed in the LIP with linear capture-point feedback control [3], [5], [6].

Combining Equations (32) and (34) yields:

\[
r_1^{xy} = r_1^{xy} + \frac{\omega_i(c_{xy}^2 - r_1^{xy}) + c_{xy}^2}{F(\omega_1)}
\]

At this stage, the roboticist can explore different CoP strategies via the choice of a function \( F \). We choose a power law parameterized by \( \alpha \in (0, 1) \):

\[
f(s \omega) = \left( \frac{s \omega}{\omega_1} \right)^\alpha \Rightarrow F(s \omega) = (1 - \alpha) \omega_1 \left( \frac{s \omega}{\omega_1} \right)^\frac{1}{\alpha}
\]

With this choice, \( F(\omega_1) = (1 - \alpha) \omega_1 \) and the horizontal coordinates of the instantaneous CoP imposed by the boundedness condition become:

\[
r_i^{xy} = r_i^{xy} + \frac{1}{1 - \alpha} \left[ c_{xy}^2 + \frac{c_{xy}^2}{\omega_i} - r_i^{xy} \right]
\]

where we recognize the same expression as in capture-point feedback control of the LIP [3], [5], [6]. Note that the three-dimensional position \( r_i \) of the CoP is readily available from \( r_i^{xy} \) by vertical projection:

\[
r_i = r_i^{xy} - h(r_i^{xy}) e_z
\]

The current state \( x_1 \) and target capture state \( x_1 \) being given, the only decision variable left on the right-hand side of Equation (37) is \( \omega_1 \). At this point, we have almost reduced the CoP capturability conditions to \( \omega_1 \): with respect to Property 3, \( r(s) \) converges to \( r_1 \) (ii) and satisfies the boundedness condition by selecting \( r_1 \) from Equation (37) (iii). We now need to make sure that the CoP trajectory is feasible (i).

By convexity of the contact area, the CoP trajectory is feasible if and only if both its ends \( r_1 \) and \( r_1^c \) are in the area. We assume the latter does by construction. For the former, the constraint that \( r_1 \) belongs to the contact polygon can be described in halfspace representation by a matrix-vector inequality \( H r_1^{xy} \leq p \), with \( H \) an \( m \times 2 \) matrix and \( p \) an \( m \)-dimensional vector. For example, a rectangular capture area written in the contact frame \( (t, b, n) \) as:

\[
\pm t \cdot (r_1 - o) \leq X
\]

\[
\pm b \cdot (r_1 - o) \leq Y
\]

can be reformulated equivalently in the horizontal plane:

\[
\pm (b \times e_z)(r_1^{xy} - o^{xy}) \leq X(e_z \cdot n)
\]

\[
\pm (t \times e_z)(r_1^{xy} - o^{xy}) \leq Y(e_z \cdot n)
\]

Injecting Equation (37) into inequalities \( H r_1^{xy} \leq p \) yields:

\[
[a H r_1^{xy} + (1 - \alpha) p - H c_i^{xy}] \omega_1 \geq H c_i^{xy}
\]

Each line of this vector inequality \( u \omega_1 \geq v \) provides a lower or upper bound on \( \omega_1 \) depending on the sign of the factor in front of it:

\[
\omega_{1, \text{min}} = \max \left( \sqrt{\lambda_{\text{min}}}, \max_j \left\{ \frac{v_j}{u_j} : u_j > 0 \right\} \right)
\]

\[
\omega_{1, \text{max}} = \min \left( \sqrt{\lambda_{\text{max}}}, \min_j \left\{ \frac{v_j}{u_j} : u_j < 0 \right\} \right)
\]

We have thus reduced the feasibility condition on the CoP to an inequality constraint on \( \omega_1 \):

\[
\forall s, r(s) \in C \iff \omega_{1, \text{min}} \leq \omega_1 \leq \omega_{1, \text{max}}
\]

Overall, the CoP strategy allows us to reduce the three-dimensional capturability condition over \( s \mapsto \lambda(s), r(s) \) (Property 3) into a one-dimensional condition over \( s \mapsto \lambda(s) \):

**Property 4** (1D capturability). Let \( x_1 = (c_i, \dot{c}_i) \) denote a capturable state and \( x_1 = (c_i, 0) \) a static equilibrium. Under the time-varying CoP strategy:

\[
r(s) = r_1 + (r_1 - r_1^c) \left( \frac{s \omega}{\omega_1} \right)^\alpha \quad \alpha \in (0, 1),
\]
s → \lambda(s) is a capture input from x_i to x_i if and only if:

(i) \omega_1 \in [\omega_{1\text{min}}, \omega_{1\text{max}}] and \forall s \in [0,1], \lambda(s) \in [\lambda_{\text{min}}, \lambda_{\text{max}}],
(ii) \lim_{s\to 0} \lambda(s) = \lambda_1(c_j),
(iii) it satisfies the boundedness condition:

\[ \int_0^1 \frac{ds}{\omega(s)} = \frac{\omega_2 h_i + \dot{h}_i}{g} \] (48)

where \omega_i denotes the initial value (at s = 1) of the solution \omega to the differential equation \omega(s') = \lambda.

C. Formulation as an optimization problem

Let us compute piecewise-constant functions s \mapsto \lambda(s) that satisfy the three conditions from Property 4. We partition the interval [0,1] into n – 1 fixed segments 0 = s_0 < s_1 < ... < s_{n-1} < s_n = 1 such that \forall s \in [s_j, s_{j+1}], \lambda(s) = \lambda_j. Note how the interval is closed to the right (\lambda(s_{j+1}) = \lambda_j) but open to the left. Define:

\[ \varphi(s) := s^2 \omega^2 \quad \delta_j := s_j^2 - s_{j+1}^2 \] (49)

The quantity \varphi represents a squared velocity and is commonly considered in time-optimal retiming [27], while \delta_j's can be interpreted as corresponding discretization steps. In practice, we use s_j = j/n so that \delta_j = (2j+1)/n^2, which is one choice among many. Remarking that \varphi' = 2s\lambda from the Riccati equation (29), we can compute \varphi(s) for s \in [s_j, s_{j+1}] as:

\[ \varphi(s) = \sum_{k=0}^{j-1} \lambda_k \delta_k + \lambda_j (s^2 - s_j^2) = \varphi_j(s) + \lambda_j (s^2 - s_j^2) \] (50)

In what follows, we use the shorthand \varphi_j := \varphi(s_j). The values \lambda(s) and \omega(s) for s \in [s_j, s_{j+1}] can be computed back from \varphi using Equations (49) and (50):

\[ \lambda_j = \frac{\varphi_{j+1} - \varphi_j}{\delta_j} \quad \omega(s) = \frac{1}{s} \sqrt{\varphi_j + \lambda_j (s^2 - s_j^2)} \] (51)

Owing to this property, we choose the vector \varphi := [\varphi_1 \ldots \varphi_n] to be the decision variable of our optimization problem. Note that this vector starts from \varphi_1, as \varphi_0 = 0 by definition.

a) Feasibility (i): noting how \varphi_n = \omega_{1i}^2 from the equation above, both feasibility conditions can be expressed as:

\[ \omega_{1\text{min}}^2 \leq \varphi_n \leq \omega_{1\text{max}}^2 \] (52)

\[ \forall j < n, \quad \lambda_{\text{min}} \delta_j \leq \varphi_{j+1} - \varphi_j \leq \lambda_{\text{max}} \delta_j \] (53)

b) Convergence (ii): \lambda(s) converges to \lim_{s\to 0} \lambda(s) = \lambda_1 = \varphi_1/\delta_0. Convergence to \lambda_f can thus be expressed as:

\[ \varphi_1 = \delta_0 \lambda_f = \frac{\delta_0 g}{h_f} \] (54)

(Recall that \varphi_1 corresponds to the last time interval by definition of s.) The parameter h_f corresponds to the CoM height of the capture state.

D. Computation and behavior of CoM capture trajectories

Suppose that we know the solution \varphi^* to the capture problem, e.g. computed using the solver from Section IV. The first step to return from s to time trajectories is to calculate the times \tau_j = t(s_j) where the stiffness \lambda switches from one value to the next. With this definition, \tau_{n} = 0 and the partition 1 = s_n > ... > s_0 = 0 corresponds to the time partition 0 = t_n < t_{n-1} < ... < t_1 < \infty.

Recall how the piecewise-constant values of \lambda are readily computed from \varphi via Equation (51). On an interval \{t_{j+1}, t_j\} where \lambda(t) = \lambda_j is constant, we can solve the differential equation \ddot{s} = \lambda_j \dot{s} to obtain:

\[ s(t) = s_{j+1} \left[ \cosh(x_j(t)) - \frac{\omega(s_{j+1})}{\sqrt{\lambda_j}} \sinh(x_j(t)) \right] \] (60)
where \( x_j(t) = \sqrt{\lambda_j}(t - t_{j+1}) \). Solving for the boundary condition \( s(t_j) = s_j \) yields the next time \( t_j = t(s_j) \):

\[
t(s_j) = t_{j+1} + \frac{1}{\sqrt{\lambda_j}} \log \left( \frac{\sqrt{\phi_{j+1}} + \sqrt{\lambda_j s_j}}{\sqrt{\phi_j} + \sqrt{\lambda_j s_j}} \right)
\]

(61)

By backward recursion, we can thus compute the time partition \( 0 = t_n < t_{n-1} < \ldots < t_1 < \infty \). The set of values \((\lambda_j, t_j)\) characterizes \( \lambda(t) \) as a function of time.

Given both \( s(t) \) from Equation (60) and \( \omega(s) \) from Equation (51), we can compute the CoP trajectory:

\[
r(t) = r_i + (r_f - r_i) \left( \frac{s(t)\omega(s(t))}{\sqrt{\phi_n}} \right)^{\frac{1}{\alpha}}
\]

(62)

Finally, once both components \( \lambda(t) \) and \( r(t) \) of the capture input are known, we can compute the CoM capture trajectory by forward integration of the VHIP equation of motion (5).

Figure 4 shows various such trajectories for zero-step capture scenarios with the same initial state \( x_i \) but different contact locations.

### IV. Optimization of Capture Problems

While the capture problem (59) belongs to the general class of nonconvex optimization, we can leverage its structural properties to solve it much faster than a generic nonconvex problem. This Section presents a dedicated solver that can solve capture problems within tens of microseconds. Readers more interested in walking pattern generation can skip directly to Section V.

In what follows, we assume that the reader is already familiar with common knowledge in numerical optimization, including the active-set method for quadratic programming (QP) and the sequential quadratic programming (SQP) method for nonlinear optimization. An overview of this background is provided for reference in Appendix B.

#### A. Problem reformulation

The capture problem (59) has a linear least squares cost, linear constraints and a single one-dimensional nonlinear equality constraint. Its objective (59a) can be written \( \|J\phi\|^2 \) where the cost matrix \( J \) is the \((n-1) \times n \) matrix given by:

\[
J = \begin{bmatrix}
-d_0 - d_1 & d_1 \\
d_1 & -d_1 - d_2 & d_2 \\
& \ddots & \ddots & \ddots \\
& & -d_{n-2} - d_{n-1} & d_{n-1}
\end{bmatrix}
\]

with \( d_j = \delta_j^{-1} \). Equation (59b) rewrites to \( b(\phi) = 0 \) with:

\[
b(\phi) = \sum_{j=1}^{n-1} \frac{\delta_j}{\sqrt{\phi_{j+1}} + \sqrt{\phi_j}} - \frac{h_i \sqrt{\phi_n} + \bar{h}_i}{g}
\]

(63)

where \( h_i > 0 \) and \( \bar{h}_i \in \mathbb{R} \) are problem parameters, i.e. constant during the optimization. Linear constraints (59c)-(59e) have the form:

\[
l \leq C\phi \leq u
\]

(64)

where the two vectors \( l \in \mathbb{R}^{n+1} \) and \( u \in \mathbb{R}^{n+1} \) are also problem parameters, and \( C \) is the \((n+1) \times n \) matrix:

\[
C = \begin{bmatrix}
C_Z & e_n^T
\end{bmatrix}
\]

where \( C_Z = \begin{bmatrix} 1 \\ -1 \\ 1 \\ \vdots \\ \vdots \\ -1 \\ 1 \end{bmatrix} \)

Finally, once both components \( \lambda(t) \) and \( r(t) \) of the capture input are known, we can compute the CoM capture trajectory by forward integration of the VHIP equation of motion (5).

Figure 4 shows various such trajectories for zero-step capture scenarios with the same initial state \( x_i \) but different contact locations.

### B. Application of an SQP approach

We apply the SQP method to the reformulation (65). Let us denote by \( f(\phi) \) the objective (65a) of the problem and \( J := \nabla_{\phi} b \) the gradient of the nonlinear constraint. The Lagrangian of problem (65) is

\[
\mathcal{L}(\phi, \lambda_- \lambda_+) = f(\phi) + \lambda_-^T (l - C\phi) + \lambda_+^T (C\phi - u)
\]

with \( \lambda_-, \lambda_+ \in \mathbb{R}^{n+1} \) the corresponding Lagrange multipliers. Let us index by \( \square_k \) the value of any quantity at iteration \( k \) of the SQP method. The Hessian matrix at iteration \( k \) is then:

\[
(\nabla^2_{\phi\phi} \mathcal{L})_k = J^T J + \mu^2 J_k J_k^T + \mu b_k (\nabla^2_{\phi\phi} b)_k
\]

(66)

We adopt the Gauss-Newton approximation \( \nabla^2_{\phi\phi} \mathcal{L}_k \approx J^T J + \mu^2 J_k J_k^T \), a classical approach for nonlinear least squares. It is particularly well-suited to the present case, as \( \nabla^2_{\phi\phi} \mathcal{L} \) is exactly equal to \( J^T J + \mu^2 J_k J_k^T \) when the boundedness condition
\( b = 0 \) is satisfied. Under this approximation, the problem for one iteration of the SQP method becomes:

\[
\begin{align*}
\text{minimize} & \quad \frac{1}{2} \| Jp + J\varphi_k \|^2 + \frac{\mu^2}{2} \| J_k^T p + b_k \|^2 \\
\text{subject to} & \quad l'_k \leq Cp \leq u'_k
\end{align*}
\]  
\tag{67a}

where \( l'_k := l - C\varphi_k \) and \( u'_k := u - C\varphi_k \). If \( \varphi_k \) satisfies the linear constraints \((65b)\), then \( p = 0 \) is a feasible point for the problem \((67)\). This problem is a linear least squares with inequality constraints \((\text{LSI})\), a particular case of QP, that we can solve using the active-set method (Algorithm 1). Adopting \( d \) and \( j \) for the step and iteration number of the QP (keeping \( p \) and \( k \) for the SQP), an iteration of the active-set method solves in this case:

\[
\begin{align*}
\text{minimize} & \quad \frac{1}{2} \| J(d + p_j + \varphi_k) \|^2 + \frac{\mu^2}{2} \| J_k^T (d + p_j) + b_k \|^2 \\
\text{subject to} & \quad C_{W_j} d = 0
\end{align*}
\]  
\tag{68a}

with \( W_j \) the set of active constraints at the current iteration \( j \) and \( C_{W_j} \) the corresponding matrix.

This can be solved in two steps using the nullspace approach. First, compute a matrix \( N \) with \( W \) and \( p \) and \( k \) can solve using the active-set method (Algorithm 1). Adopting \( d \) and \( j \) for the step and iteration number of the QP (keeping \( p \) and \( k \) for the SQP), an iteration of the active-set method solves in this case:

\[
\begin{align*}
\text{minimize} & \quad \frac{1}{2} \| J(d + p_j + \varphi_k) \|^2 + \frac{\mu^2}{2} \| J_k^T (d + p_j) + b_k \|^2 \\
\text{subject to} & \quad C_{W_j} d = 0
\end{align*}
\]  
\tag{68b}

Second, solve this unconstrained problem: taking \( T \) and \( u \) such that the above objective writes \( \frac{1}{2} \| Tz + u \|_n^2 \), compute the QR decomposition \( T = QR \), and solve \( QRz = -u \). The latter is equivalent to \( z = -R^{-1}Q^Tu \) if \( R \) has full rank [28, Chapter 10]. Both of these steps can be significantly tailored to the case of capture problems.

C. Tailored operations

In the SQP, most of the time is spent in solving the underlying LSI: the computation of \( N_W \), the post-multiplication by \( N_W \) to obtain \( T \), and the QR decomposition of \( T \) and the computation of the Lagrange multipliers needed to check KKT conditions can be computed in \( O(n^2) \) by taking advantage of the structure of \( C_W \) and \( J \) (Appendix B-D).

\begin{itemize}
  \item The matrix \( N_W \) does not need to be computed explicitly.
  \item Rather, the cost matrix \( T \) of the unconstrained least squares problem can be built directly in \( O(n) \) operations by taking advantage of the structure of \( C_W \) and \( J \) (Appendix B-D).
  \item Lagrange multipliers needed to check KKT conditions can be computed in \( O(n^2) \) by taking advantage of the structure of \( C_W \).
\end{itemize}

D. Numerical and algorithmic considerations

The implementation of a general-purpose QP or SQP solver is an extensive work due to the numerous numerical difficulties that can arise in practice: active-set methods need to perform a careful selection of their active constraints in order to keep the corresponding matrix well conditioned, while SQPs require several refinements, some of which imply solving additional QPs at each iteration [28]. While the tailored operations we presented reduce the theoretical complexity w.r.t. general-purpose solvers, there are also a number of features of Problem \((65)\) that allow us to stick with a simple, textbook implementation, and contribute to the general speed-up.

On the QP side, the matrix \( C_Z \) is always full rank and well conditioned, while the last row \( e_j^T \) of \( C \) is a linear combination of all rows from \( C_Z \) \((e_n = C_{Z1n})\). As a consequence, all matrices \( C_W \) are full rank and well conditioned, save for the case where all \( n + 1 \) constraints are active. This case can be easily detected and avoided.\(^7\) It is thus safe to use a basic active-set scheme.

All QR decompositions are performed on matrices with rank deficiency of at most one. As a consequence, it is not necessary to use more involved column-pivoting algorithms, and the rank deficiency can be detected by simply monitoring the bottom-right element of the triangular factor. While we don’t prove that the matrices \( J_W \) are well conditioned, we verified this assertion for \( n \leq 20 \) in a systematic way. Even for large values of \( n \), the QR decomposition of \( T \) is stable as the row with largest norm appears first [30, p. 169].

On the SQP side, the odds are very favorable: constraints are linear and the Gauss-Newton approach offers a good approximation of the Hessian matrix. As a consequence, we observed that the method takes full steps 98.5\% of the time in practice, and converges in very few iterations (four on average). Since we start from a feasible point, all subsequent iterates are guaranteed to be feasible and the line search needs only monitor the objective function, in an unconstrained-optimization fashion.

E. Pre-computation of QR decompositions

It is important to note that the matrices \( J \) and \( C \) only depend on the problem size \( n \) and partition \( s_0, \ldots, s_n \), which are the same across all capture problems that we solve in this work. If \( n \) is small enough (say \( n \leq 20 \)), we can precompute and store the QR decompositions of all possible \( J_W := JN_W \). Note that there are \( 2^{n+1} - 1 \) different sets \( W \), including all

\[^7\] With the notations of Appendix B-G, this can only happen if \( a^- \) or \( a^+ \) is equal to 0 or 1. A workaround is then to slightly perturb \( \omega_{\min} \) or \( \omega_{\max} \).
combinations of up to \( n \) active constraints among \( n+1 \). These pre-computations can be done in a reasonable amount of time thanks to our specialized QR decompositions, and result in even faster resolution times.

### F. Performance comparison

In [11], Problem (59) was solved with the state-of-the-art solver IPOPT [31], which is written in Fortran and C. We compare its performances with our tailored SQP approach, implemented in C++.\(^8\) Taking \( \mu = 10^6 \), the solutions returned by both methods are numerically equivalent (within \( 10^{-7} \) of one another, and \(|b(\varphi)| \approx 10^{-8}\) in both cases). Computation times over representative problems produced during walking pattern generation are reported in Table I, where our approach is denoted \( \text{SQP} + \text{cLS} \) (custom least squares), and the abbreviation \( \text{pre.} \) denotes the use of QR pre-computations for \( J_W \). In practice, we work with \( n = 10 \), for which our solver is 300–400 times faster than the generic solver IPOPT.

Computation times for QR pre-computations range from 2 ms for \( n = 10 \), 100 ms for \( n = 15 \), to 4.9 s for \( n = 20 \). This is not limiting in practice, as these computations are performed only once at startup. The limit rather lies with memory consumption, which follows an exponential law ranging from 2 MB for \( n = 10 \) to 5 GB for \( n = 20 \).

To break down how much of the speed-up is due to the problem reformulation and how much is due to our custom least squares implementation, we also test our SQP method using the state-of-the-art least squares solver LSSOL [32]. This variant is denoted by \( \text{SQP} + \text{LSSOL} \). For \( n \geq 25 \), \( \text{SQP} + \text{LSSOL} \) starts to fail on some problems, with a failure rate of roughly 25% for \( n = 50 \). This suggests that the least squares component of our solver is more robust. There are two plausible explanations for this. First, LSSOL assumes all matrices are dense, while we leverage sparsity patterns of \( J \) and C. Second, LSSOL treats all coefficients as floating-point numbers, while knowing that the elements of \( C \) and \( J \) matrices are dense, while we leverage sparsity patterns of \( C_W \) for the nullspace and pseudoinverse of \( C_W \).

### V. WALKING PATTERN GENERATION

While zero-step capturability enables push recovery, one-step capturability is the minimum price to pay for walking.\(^9\)

We will now see how this next level of capturability can be solved with a pipeline similar to the balance control one:

- Change variable from time \( t \) to a new variable \( s \)
- Define the CoP \( r(s) \) as piecewise-constant function of \( s \)
- Parameterize its contact-switch time by a new variable \( \alpha \)
- Reduce capturability to an optimization over \( \varphi(s, \alpha) \)
- Compute the optimal solution \( \varphi^*(s, \alpha) \) of this problem
- Change variable from \( s \) to time \( t \)

From \( \varphi^*(t) \), it is then straightforward to compute the full capture input \( t \mapsto \lambda^*(t), r^*(t) \) as well as the capture trajectory \( c^*(t) \). Let us now detail each step of this pipeline.

#### A. Time-varying CoP strategy

In a one-step capture setting, the contact area switches instantaneously at a given instant \( s_c \in (0, 1) \):

\[ C(s) = \begin{cases} C_1 & \text{for } s_c < s \leq 1 \\ C_2 & \text{for } 0 < s \leq s_c \end{cases} \]  
(70)

Due to this discontinuity, we cannot adopt the same line-segment CoP strategy that we used in Section III for balance control. Let us then adopt a piecewise-constant CoP trajectory:

\[ r(s) = \begin{cases} r_1 & \text{for } s_c < s \leq 1 \\ r_2 & \text{for } 0 < s \leq s_c \end{cases} \]  
(71)

This choice yields the following boundedness condition (31):

\[ -g \int_0^1 \frac{ds}{\dot{\omega}(s)} = \omega_i (c_i - r_i) + (r_1 - r_i) s_c \dot{\omega}(s_c) + \epsilon_i \]  
(72)

In terms of our optimization variable \( \varphi \), the right-hand side of this equation is:

\[ (c_i - r_i) \sqrt{\varphi_n} + (r_1 - r_i) \sqrt{\varphi(s_c)} + \epsilon_i \]  
(73)

where \( \varphi(s_c) \) is a linear combination of \( \varphi_j \) and \( \varphi_{j+1} \) when \( s_c \in [s_j, s_{j+1}] \) from Equation (50).

Let us define an external parameter \( \alpha \in (0, 1) \). This time, we parameterize the contact switch \( s_c \) of the CoP trajectory by \( \alpha \) as follows:

\[ s_c = \text{the scalar s.t. } \sqrt{\varphi(s_c)} = \alpha \sqrt{\varphi_n} \]  
(74)

Under this assumption, the right-hand side of the boundedness condition then simplifies to:

\[ (c_i - r_i) \sqrt{\varphi_n} + \epsilon_i \]  
(75)

where \( r_\alpha := \alpha r_1 + (1 - \alpha) r_i \) is the equivalent constant CoP [2] of the time-varying CoP trajectory. In the LIP with constant \( \omega_{LP} \), Equation (74) rewrites to \( \alpha = e^{-\omega_{LP} t_c} \),\(^10\) which shows how \( \alpha \) parameterizes the time of contact switch. This observation still holds for the VHIP, where we will use \( \alpha \) as an external variable to optimize over contact switch times.

Taking the dot product of this equation with the normal \( n_i \) of the initial contact \( C_i \) yields:

\[ \int_0^1 \frac{ds}{\dot{\omega}(s)} = \frac{h_\alpha \sqrt{\varphi_n} + \dot{h}_i}{g} = h_\alpha := \frac{(c_i - r_\alpha) \cdot n_i}{(e_z \cdot n_i)} \]  
(76)

\(^8\) We only report averages for IPOPT computation times as they lie on a different scale. These averages are higher than those reported in [11] because we evaluate both feasible and infeasible problems (for reasons made clear in the next section), while all random initial conditions in [11] were zero-step capturable. For \( n = 10 \) and projecting performance statistics on feasible problems only, IPOPT’s computation times decrease to 1600 ± 790 \( \mu s \).

\(^9\) https://github.com/jrl-umi3218/CaptureProblemSolver

\(^{10}\) This expression corresponds to the limit of the scalar \( w' \) when \( \Delta t_1 \to \infty \) in Equation (25) of [2]. It also plays an important role in the step timing adjustment method from [33].
This constraint is exactly of the form (59b) of the capture problem (59), where the parameter $h_i$ has simply been replaced by $h_{\alpha}$.

The horizontal components of Equation (75) give:

$$r_{xy}^i = r_{xy}^f + \frac{1}{1-\alpha} \left[ c_{xy}^i + \frac{c_{xy}^f}{\omega_i} - r_{xy}^f \right]$$

(77)

This expression is identical to Equation (37). Following the same steps (38)–(46) as in the zero-step setting, we therefore conclude that capture inputs under the CoP strategy (71)–(74) are characterized by the 1D capturability condition (Property 4). In particular, we can compute them using our capture problem solver.

**B. External optimization over contact switching time**

The contact switching time $t_c(\alpha)$ plays a crucial role in one-step capture trajectories. Given the solution $\varphi_\alpha$ to the capture problem parameterized by $\alpha$, we can compute it as follows. First, compute $s_c$ using Equation (51):

$$s_c = \sqrt{s_j^2 + \alpha^2 \varphi_{\alpha,n} - \varphi_{\alpha,j}}$$

(78)

where $\varphi_\alpha(s_c) = \alpha^2 \varphi_{\alpha,n} \in [\varphi_{\alpha,j}, \varphi_{\alpha,j+1}]$. Then, apply the mapping $t_c = t(s_c)$ provided by Equation (61).

While this mapping is straightforward to compute numerically, its high nonlinearity suggests that introducing time constraints into capture problems would radically affect the design of the dedicated solver from Section IV. Fortunately, computation times achieved by this solver allow us to solve hundreds of capture problems per control cycle. We therefore choose to optimize jointly over $\varphi$ and $\alpha$ using a two-level decomposition: an external optimization over $\alpha$ in $(0,1)$, wrapping an internal optimization where $\alpha$ is fixed and the existing capture problem solver is called to compute $\varphi_\alpha$.

A straightforward way to carry out the external optimization is to test for a large number of values $\alpha \in (0,1)$. This approach is however inefficient in practice as it sends a significant amount of infeasible problems to the internal optimization. The main cause for such infeasibility is that whole intervals of values for $\alpha$ result in $\omega_{\text{min}}(\alpha) > \omega_{\text{max}}(\alpha)$, for which the CoP constraint (59c) is obviously unfeasible. To avoid this, we propose in Appendix C-A an algorithm to pre-compute the set of intervals $[\alpha_{\text{min}}, \alpha_{\text{max}}] \subset (0,1)$ on which it is guaranteed that $\omega_{\text{min}} \leq \omega_{\text{max}}$.

During bipedal walking, contact switches can only be realized after the free foot has completed its swing trajectory from a previous to a new contact. We therefore need to make sure that $t_c(\alpha)$ is greater than the remaining duration $t_{\text{swing}}$ of the swing foot motion. This gives us an additional constraint to be enforced by the external optimization:

$$t_c(\alpha) \geq t_{\text{swing}}$$

(79)

where each evaluation of $t_c(\alpha)$ costs the resolution of a full capture problem. We find such solutions by sampling $n_\alpha$ values of $\alpha$ per feasible interval $[\alpha_{\text{min}}, \alpha_{\text{max}}]$. This approach provides automatic step timings, and tends to output pretty dynamic gaits.

Alternatively, when step timings are provided one may want to enforce an equality constraint:

$$t_c(\alpha) = t_{\text{swing}}$$

(80)

In this case, the external optimization implements an approximate gradient descent, calling the capture problem solver multiple times to evaluate $\nabla_{\alpha} t_c$ and converge to a local optimum on each feasible interval $[\alpha_{\text{min}}, \alpha_{\text{max}}]$. Although we don’t prove it formally, we observed in practice that the mapping $t_c(\alpha)$ seems to be always monotonic (as in the LIP) and the optimum thus found global on its interval.

**C. Pattern generation from capture trajectories**

We can generate walking from capture trajectories via a two-state strategy [19], [34] combining zero-step and one-step capture problems. The robot starts in a double-support posture, and computes a one-step capture trajectory to transfer its CoM to its first support foot.

- **Single support phase**: swing foot tracks its pre-defined trajectory while the CoM follows a one-step capture trajectory updated in a model predictive control fashion. The phase ends at swing foot touchdown.
- **Double support phase**: the CoM follows a zero-step capture trajectory towards the target contact. Meanwhile, the capture problem solver is called to compute a one-step capture trajectory towards the next contact. The phase ends as soon as such a trajectory is found.

The behavior realized by this state machine is conservative: after touchdown, the robot uses its double support phase to slow down until a next one-step capture trajectory is found. If the next contact is not one-step capturable, the robot stops walking and balances in place. Otherwise, walking continues toward the next pair of footsteps.

A limitation of this walking pattern generator is that double-support phases do not include a continuous CoP transfer from one contact to the next. The one exception to this is when both contacts are coplanar, in which case we can use their convex hull as halfspace representation $(H, p)$ in Equation (43) and thus apply the line-segment CoP strategy from Section III-B. In practice, we let CoP discontinuities occur in the walking pattern during double support and rely on the underlying robot stabilizer to handle them (see Section VI-B).

**VI. SIMULATIONS**

We validated this walking pattern generator in two successive implementations. First, we evaluate the kinematics of whole-body tracking of the VHIP reference in a prototyping environment. Then, we implement our method as a full-fledged C++ controller and evaluate its performance compared to the state of the art in two dynamic simulators.

**A. Kinematics**

We implemented our walking pattern generation method in the pymanoid prototyping environment. Whole-body inverse kinematics is carried out using a standard quadratic-programming formulation (see e.g. [35, Section 1] for a

11 https://github.com/stephane-caron/pymanoid
survey). Tracking of the inverted pendulum and swing foot trajectories is realized by the following set of tasks:

<table>
<thead>
<tr>
<th>Task group</th>
<th>Task</th>
<th>Weight</th>
</tr>
</thead>
<tbody>
<tr>
<td>Foot tracking</td>
<td>Support foot</td>
<td>1</td>
</tr>
<tr>
<td>Foot tracking</td>
<td>Swing foot</td>
<td>10−3</td>
</tr>
<tr>
<td>VHIP tracking</td>
<td>Center of mass</td>
<td>1 × 10−2</td>
</tr>
<tr>
<td>VHIP tracking</td>
<td>Min. ang. mom. variations</td>
<td>1 × 10−4</td>
</tr>
<tr>
<td>Regularization</td>
<td>Keep upright chest</td>
<td>1 × 10−4</td>
</tr>
<tr>
<td>Regularization</td>
<td>Min. shoulder extension</td>
<td>1 × 10−5</td>
</tr>
<tr>
<td>Regularization</td>
<td>Min. upper-body velocity</td>
<td>5 × 10−6</td>
</tr>
<tr>
<td>Regularization</td>
<td>Reference upright posture</td>
<td>1 × 10−6</td>
</tr>
</tbody>
</table>

We considered the three scenarios depicted in Figure 5. The first one, an elliptic staircase with randomly-tilted footsteps (Figure 5a), tests the ability to adapt to general uneven terrains. It illustrates the main advantage of the capture problem formulation: it provides real-time model predictive control (CoP and unilaterality) constraint saturation over rough terrains. Existing methods either enforce constraints on regular terrains such as floors or stairs [36], [37], or do not take constraint saturation into account [17], [6], or compute costly nonconvex optimizations offline [38], [39] (see Section VII for a complete discussion).

We also consider a regular staircase with 15-cm high steps (Figure 5b), and the real-life scenario provided by Airbus Group depicted in Figure 5c. It consists of a 1:1 scale model of an A350 aircraft under construction in a factory environment. To reach its desired workspace configuration, the humanoid has to walk up an industrial-grade staircase (step height 18.5 cm, except the last one which is 14.5 cm), then across a flat floor area and finally inside the fuselage where the ground consists of temporary wooden slabs. All three walking patterns are depicted in Figure 5 and in the accompanying video.

We use the same set of parameters on all scenarios. For capture problems, we choose \( n = 10 \) discretization steps with a partition \( s_i = i/n \) and set \( h_t = 0.8 \) m, which is a suitable CoM height for HRP-4 at rest with an extended leg. The external optimization for one-step problems samples \( n_\alpha = 5 \) values of \( \alpha \) per feasibility interval, which is sufficient to find solutions that satisfy the inequality constraint (79) on contact switching time. For the VHIP stiffness feasibility, we set \( \lambda_{\text{min}} = 0.1g \) and \( \lambda_{\text{max}} = 2g \).

At each control cycle, the capture problem solver is called on both zero-step and one-step capture problems. On a consumer laptop computer, zero-step problems were solved in \( 0.38 \pm 0.13 \) ms while one-step ones were solved in \( 2.4 \pm 1.1 \) ms (average and standard deviations over 10,000 control cycles). Note that these computation times reflect both calls to the C++ solver (Section IV) and the external optimization over \( \alpha \) implemented in the prototyping environment in Python.

We release the code of this prototype as open source software.\(^{12}\)

B. Dynamics

To test our method in dynamic simulations and compare it to the existing, we extend the C++ LIP-based stair climbing controller from [40]. This controller consists of two main components: a LIP-based pattern generator by model predictive control (i.e., the pattern is produced online in receding horizon) and a stabilizer based on whole-body admittance control. We implement our solution as a second pattern generator in this controller, using the same stabilizer in both cases for comparison.

The planar CoM constraint of the LIP is applicable to small step heights (e.g., Kajita et al. applied it to 10-cm high steps in [41]) but can become problematic for higher steps. Another option is to constrain the CoM to a non-horizontal inclined plane parallel to the staircase, yet in this case either the CoP is kept constant [42] or the CoP constraint becomes nonlinear and harder to enforce [13]. From experiments on the HRP-4 robot, [40] chose to decompose the CoM constraint in two horizontal plane segments (thus keeping linear CoP inequalities), introducing the vertical height variation at toe liftoff. These segments could be further decomposed for better adaption to terrain, but as of today there is no known algorithm

\(^{12}\) [Link](https://github.com/stephane-caron/capture-walkgen)
for this. Using a VHIP-based pattern generator addresses the question by removing the specification of plane segments altogether.

Thanks to the mc rtc control framework,\footnote{Developed in the course of the COMANOID and JRP projects, this control framework is available at \url{https://gite.lirmm.fr/multi-contact/mc rtc} upon request and will be released soon.} we validated the feasibility of our method in two different dynamic simulators:

- **Choreonoid**\footnote{\url{http://choreonoid.org/en/}}, the most reliable one at our disposal, which implements the projected Gauss-Seidel method\footnote{\url{http://choppelarobotics.com/}} and runs in real time.
- **V-REP**\footnote{\url{http://choreonoid.org/en/}} with the Newton dynamics engine. For a realistic behavior, we set simulations to a relatively small time step of 5 ms, in which case they did not run in real time.

The same set of controller parameters, and thus the same walking patterns, were applied in both simulators. Testing in two simulators allowed us to validate the robustness of our walking controller to varying environment responses: ground contacts in V-REP with Newton are more flexible, with smoothed and oscillatory contact forces, while those of Choreonoid are more rigid and closer to what is observed on the real robot.

Figures 6 and 7 compare LIP-based and VHIP-based controller simulations at different time instants (full simulation recordings are shown in the accompanying video). The main difference between the two lies in knee flexion-extension (Figure 9): thanks to its reference height going up after the first step, the VHIP pattern brings the robot to knee extension while the LIP one bends down then raises again (Figure 8) owing to its choice of a horizontal rather than inclined LIP plane. In the horizontal plane, DCM-ZMP tracking looks similar to a step of 5 ms, in which case they did not run in real time.

In the 2D horizontal plane similarly to the LIP, yet with a time-variant rather than time-invariant equation of motion. \footnote{\url{https://github.com/stephane-caron/capture_walking_controller}} See the existing literature. To start with, the exponential dichotomy of the time-varying inverted pendulum was proposed in 2004 by J. Hauser et al.\footnote{Both [45] and [17] use polynomial CoM height functions, which makes it easy to satisfy boundary conditions but yields non-integrable dynamics. Terada and Kuniyoshi\footnote{Terada and Kuniyoshi\footnote{https://github.com/stephane-caron/capture_walking_controller} proposed a symmetric alternative where the system becomes integrable, yet where enforcing boundary conditions is a nonlinear root finding problem.} proposed a symmetric alternative where the system becomes integrable, yet where enforcing boundary conditions is a nonlinear root finding problem.} to address a question of motorcycle balance. Its application to the linear inverted pendulum can be found in the motion generation framework of the Honda ASIMO humanoid\footnote{Both [45] and [17] use polynomial CoM height functions, which makes it easy to satisfy boundary conditions but yields non-integrable dynamics. Terada and Kuniyoshi\footnote{Terada and Kuniyoshi\footnote{https://github.com/stephane-caron/capture_walking_controller} proposed a symmetric alternative where the system becomes integrable, yet where enforcing boundary conditions is a nonlinear root finding problem.} proposed a symmetric alternative where the system becomes integrable, yet where enforcing boundary conditions is a nonlinear root finding problem.}

Fig. 6. **Dynamic simulations in Choreonoid:** snapshots at the maximum knee flexion time instant. \textit{Left:} LIP-based pattern generation. \textit{Right:} VHIP-based pattern generation. The former causes HRP-4 to bend its knees more, increasing the risk of shank collision with the staircase. We can also observe the more abrupt CoP transfer of the latter, which is specific to our method.

Fig. 7. **Dynamic simulations in V-REP:** snapshots at the toe liftoff time instant. Simulations use the Newton dynamics engine with a time step of 5 ms and one pass per frame. \textit{Left:} LIP-based pattern generation. \textit{Right:} VHIP-based pattern generation. The former causes HRP-4 to bend its knees more, raising the risk of shank collision with the staircase.

We release the code of this controller as open source software.\footnote{https://github.com/stephane-caron/capture_walking_controller}
Interestingly, in [17] Hopkins et al. use the Riccati equation (18) to compute $\omega(t)$ from $e^x(t)$, while in the present study we compute $e(t)$ from $\omega(t)$. More generally, our strategy can be seen as mapping the whole problem onto the damping $\omega$ and solving for $\omega(t)$, while the underlying strategy behind those other approaches is to fix $\omega$ and map the remainder of the problem onto $e^x$. Another noteworthy example of the latter can be found in the linearized MPC proposed by Brasseur et al. [37], where $\omega$ variations are this time abstracted using polyhedral bounds rather than a pre-planned height trajectory.

In this regard, our present study is more akin to works on capturability proposed for the 2D nonlinear inverted pendulum [8], [9], [10]. All of them share a design choice dating back to the seminal work of Pratt and Drakunov [8]: they interpolate CoM trajectories in a 2D vertical plane with a fixed center of pressure (CoP). The key result of [8] is the conservation of the “orbital energy” of a CoM path, a variational principle that we can now interpret as a two-dimensional formulation of the boundedness condition. This principle was later translated into a predictive controller in an equally inspirational study by Koolen et al. [10]. Ramos and Hauser [9] also noticed that the capture point, interpreted as point where to step, was a function of the CoM path. They proposed a single-shooting method to compute what we would now call 2D capture trajectories.

All of these works hinted at key features of 3D capture trajectories, but applied only to two-dimensional CoM motions in vertical planes. The key to lift this restriction is the 3D boundedness condition, which was first formulated in the case of the LIP by Lanari et al. [22] and applied to model predictive control of the LIP in [46]. This condition can be more generally applied to different asymptotic behaviors, including but not restricted to stopping. For instance, infinite stepping is another option [47]. The exploration of these more general asymptotic behaviors is an open question.

Another important choice of the present study is to focus on zero- and one-step capture trajectories. Walking controllers based on one-step capturability have been proposed for both even [48], [33], [34] and uneven terrains [49], [19]. The latter follow a single line of work leading to the present study: [49] finds rough-terrain (even multi-contact) solutions but tends to produce conservatively slow trajectories; [19] discovers dynamic walking patterns, but suffers from numerical instabilities when used in a closed control loop. In our understanding, these instabilities are due to the direct transcription of centroidal dynamics, which has proved successful for planning [38], [39] but where closed-loop controllers suffer from frequent switches between local optima [19]. The optimization of capture problems provides an alternative transcription for which we do not observe this numerical sensitivity.

Finally, the last key choice of the present study is the change of variable from $t$ to $s$. This choice is one possible generalization of the seminal idea by Pratt and Drakunov [8] to make the CoM height a function $e^x(e^z)$ of the 2D CoM abscissa: as it turns out, $e^z(t)$ and $s(t)$ are proportional in their 2D setting [11], although that is not the case any more in 3D. An alternative 3D generalization is to solve for the remaining lateral motion $e^x(t)$ after the sagittal motion has been computed by the 2D method [50]. Both cases, as noted in [50] and in the present study, bear a close connection with time-optimal path parameterization (see e.g. [27] for a survey). Future works may explore this connection, and perhaps bring to light computational complexity results regarding the best case performance one can hope for this kind of problems.

The ability to solve capture problems in tens of microseconds opens new perspectives for motion planning and control. For instance, planners can use this tool for fast evaluation of contact reachability, while stepping stabilizers can evaluate...
several contact candidates in parallel in reaction to e.g. external pushes. These extensions are open to future works.

ACKNOWLEDGMENT

The author would like to thank Boris van Hofslot, Alain Micaelli and Vincent Thomasset for their feedback on preliminary versions of this paper. Our thanks also go to Pierre Gergondet for developing and helping us with the mc rtc framework. This research was supported in part by H2020 EU project COMANOID http://www.comanoid.eu/, RIA No 645097, and the CNRS–AIST–AIRBUS Joint Research Program (JRP).

REFERENCES

APPENDIX A

MATHEMATICAL COMPLEMENT

In this Appendix, we provide formal proofs for claims made in Section II and verify that all quantities are soundly defined.

A. Restriction to convergent input functions

Property 5. For every pair of states \( x_i \) and \( x_f = (c_i, 0) \), \( T_{x_i, x_f} \) is non-empty if and only if \( T^c_{x_i, x_f} \) is non-empty.

In other words, if there exists a capture input \( \lambda(t), r(t) \) steering an initial state \( x_i \) to a static equilibrium \( x_f \), then there exists another input \( \lambda^c(t), r^c(t) \) accomplishing the same while also converging.

Proof. As \( T^c_{x_i, x_f} \subset T_{x_i, x_f} \), it is enough to prove that \( T^c_{x_i, x_f} \) is non-empty as soon as \( T_{x_i, x_f} \) is non-empty, i.e. that given an input function \( \lambda(t), r(t) \in T_{x_i, x_f} \), we can find another input \( \lambda^c(t), r^c(t) \) steering to the same state while converging. To this end, consider the following state-dependent inputs:

\[
\sqrt{\lambda(x)} = 2 \sqrt{\frac{g(5h - h_i) + h^2 - h_i}{5h_i - h_i}} \quad (81)
\]
\[
\bar{r}(x) = c + \frac{c - c_i}{4} + \frac{c}{\sqrt{\lambda(x)}} + \frac{g}{\lambda(x)} \quad (82)
\]

This definition is chosen so that \( \bar{\lambda}(x) \) is the solution of:

\[
\left[ c - a + \frac{c - c_i}{4} \right] \cdot n + \frac{\bar{c} \cdot n}{\sqrt{\lambda(x)}} + \frac{g \cdot n}{\lambda(x)} = 0 \quad (83)
\]

As a consequence, \((\bar{r}(x_i) - o)\cdot n = 0\) and the state-dependent CoP belongs to the contact area. Moreover, \( \bar{\lambda} \) and \( \bar{r} \) are continuous functions of \( x_i \) in a neighbourhood of \( x_i \), and \( \bar{\lambda}(x_i) = \lambda_i(c_i) \) and \( \bar{r}(x_i) = r_i(c_i) \). Hence, as long as \( x(t) \) is close enough to \( x_i \), both \( \bar{\lambda}(x(t)) \) and \( \bar{r}(x(t)) \) are feasible.

Injecting those inputs into (5) yields the nonlinear differential equation:

\[
\dot{\bar{c}}(t) = -\frac{\lambda(x(t))}{4} (c(t) - c_i) - \sqrt{\lambda} \dot{\bar{c}}(t) \quad (84)
\]

It is immediate that \( x_i \) is an equilibrium for this dynamics. The linearized system around this equilibrium is:

\[
\bar{c}(t) = -\frac{\lambda_i}{4} (c(t) - c_i) - \sqrt{\lambda} \bar{c}(t) \quad (85)
\]

for which the equilibrium is stable. Therefore the equilibrium \( x_i \) of (84) is locally stable: if \( x(0) \) is close enough to \( x_i \), then \( x(t) \) remains close to \( x_i \) and converges toward this limit.

We now consider a generic input function \( \lambda(t), r(t) \in T_{x_i, x_f} \). By definition, the solution of (5) converges to \( x_i \) as \( t \to \infty \). Then, there exists some time \( T \) such that \( x(T) \) is close enough to \( x_i \) so that, starting from this position, the state-dependent control remains feasible and converges to \( x_i \).

We conclude by noting that the input function that switches at time \( T \) from \( \lambda, r \) to \( \bar{\lambda}, \bar{r} \) belongs to \( T^c_{x_i, x_f} \).

B. Solutions to the Riccati equations

Let us verify the existence of damping solutions and exhibit some of their properties that will prove useful to characterize those that don’t diverge.

Property 6. Assume that we are given \( \lambda \) such that \( \lambda(t) \in [\lambda_{\min}, \lambda_{\max}] \) at all times \( t \). Then, there exists a unique \( \omega > 0 \) such that the solution \( \omega(t) \) of (18) with \( \omega(0) = \omega_0 \) is positive and finite at all times. Moreover, this solution is such that:

\[
\forall t > 0, \quad \omega(t) \in [\sqrt{\lambda_{\min}}, \sqrt{\lambda_{\max}}] \quad (86)
\]

In other words, there is a one-to-one mapping between the stiffness function \( \lambda(t) \) and its non-diverging filtered damping \( \omega(t) \).

Proof. Consider first the case of a constant input \( \lambda \). One can note that the differential equation \( \ddot{y} = y^2 - \lambda \) has two equilibrium points: one stable \(-\sqrt{\lambda}\) and one unstable \(\sqrt{\lambda}\). More precisely, given \( y_0 \in \mathbb{R} \), the only solution of this equation satisfying \( y(0) = y_0 \) is given by

\[
y(t) = \begin{cases} \sqrt{\lambda} \tan(\sqrt{\lambda}(T-t)) & \text{if } |y_0| > \sqrt{\lambda} \\ \frac{\sqrt{\lambda}}{\lambda} & \text{if } |y_0| < \sqrt{\lambda} \\ y_0 & \text{if } |y_0| = \sqrt{\lambda} \end{cases} \quad (87)
\]

The initial condition \( y_0 \) settles the behavior of the solution at all times. Define the time \( T = \frac{1}{2\sqrt{\lambda}} \log \frac{|y_0|}{|y_0 + \sqrt{\lambda}|} \), then:

- If \( 0 \leq y_0 < \sqrt{\lambda} \), then \( \lim_{t \to \infty} y(t) = -\sqrt{\lambda} \) and the solution \( y \) becomes negative after time \( T \).
- If \( y_0 = \sqrt{\lambda} \), then \( y(T) = \sqrt{\lambda} \) for all \( t > 0 \).
- If \( y_0 > \sqrt{\lambda} \), then \( \lim_{t \to \infty} y(t) = +\infty \): the solution explodes in finite time.

Let us move now to the general case where \( \lambda(t) \) is time-varying, and denote by \( \omega \) a non-negative, non-explosive solution to (18). If \( \omega(t_0) > \sqrt{\lambda_{\max}} \) at some time \( t_0 > 0 \), then choosing the solution \( \omega(t) = y^2 - \lambda_{\max} \) with \( y(t_0) = \omega(t_0) \), we observe that, as long as \( 0 \leq y(t) \leq \omega(t) \),

\[
\omega(t) - y(t) = y^2(t) - y^2(t) + \lambda_{\max} - \lambda(t) \geq 0 \quad (88)
\]
Therefore, $\omega - y$ is nondecreasing, and $y(t) \leq \omega(t)$ holds until the explosion time $T$ of $y$. This shows that $\omega$ explodes in finite time, in contradiction with the hypothesis. Similarly, if $\omega(t_0) < \sqrt{\lambda_{\text{min}}}$ for some $t_0 > 0$, we can upper-bound $\omega(t)$ by the solution to $\dot{y} = y^2 - \lambda_{\text{min}}$ such that $\omega(t_0) = \omega(t_0)$, which becomes negative in finite time, once again contradicting the hypothesis. The bounds (86) must therefore hold.

We now prove the uniqueness of the non-negative non-explosive solution of (18). Suppose that one could find two such solutions, $\omega_1$ and $\omega_2$. As observed above, these functions remain in the interval $[\sqrt{\lambda_{\text{min}}}, \sqrt{\lambda_{\text{max}}}]$. Consider without loss of generality that $\omega_1(t) > \omega_2(t)$. Then, as long as $\omega_1(t) > \omega_2(t)$,

$$\omega_1 - \omega_2 = (\omega_1 - \omega_2)(\omega_1 + \omega_2) \geq 2\sqrt{\lambda_{\text{min}}}(\omega_1 - \omega_2) \quad (89)$$

As a consequence, $\omega_1(t) - \omega_2(t) \geq (\omega_1(0) - \omega_2(0))e^{2\sqrt{\lambda_{\text{max}}}t}$ at all times, showing that the two functions cannot be bounded at the same time.

To finally prove the existence of the solution, we observe there exists a unique $\omega(0) \in [\sqrt{\lambda_{\text{min}}}, \sqrt{\lambda_{\text{max}}}]$ such that for all $t > 0$, we have $\gamma_1(t) > \omega(t) > \gamma_2(t)$ where $\gamma_1$ and $\gamma_2$ are the solutions of (18) with conditions $\gamma_1(t) = \sqrt{\lambda_{\text{max}}}$ and $\gamma_2(t) = \sqrt{\lambda_{\text{min}}}$. The solution $\omega$ starting from this value $\omega(0)$ remains bounded by construction (the time it crosses $\sqrt{\lambda_{\text{min}}}$ or $\sqrt{\lambda_{\text{max}}}$ is greater than any finite time $t$).

Let us now turn to the other damping $\gamma$. While there is a unique solution $\omega$ corresponding to a given $\lambda$, there are many different non-negative finite functions $\gamma$ that satisfy (17). As a matter of fact, each choice of $\gamma(0) > 0$ yields an admissible solution:

**Property 7.** Assume that we are given $\lambda$ such that $\lambda(t) \in [\lambda_{\text{min}}, \lambda_{\text{max}}]$ at all times $t$. For all $\gamma(0) > 0$, the solution $\gamma$ of (17) is non-negative and finite at all times. Moreover:

$$\sqrt{\lambda_{\text{min}}} \leq \liminf_{t \to \infty} \gamma(t) \leq \limsup_{t \to \infty} \gamma(t) \leq \sqrt{\lambda_{\text{max}}} \quad (90)$$

**Proof.** The existence and the uniqueness of the solution on a maximal interval are consequences of the Cauchy-Lipschitz theorem. As in the previous proof, we can compare $\gamma$ with the functions $y$ and $\gamma$, respectively solutions to $\dot{y} = \lambda_{\text{min}} - y^2$ and $\dot{\gamma} = \lambda_{\text{max}} - \frac{y^2}{2}$ with $y(0) = \gamma(0) = \gamma(0)$. Then, $\gamma(t) \leq \gamma(t) \leq \gamma(t)$ at all times $t$, and the rest of the proof is a consequence of $\lim_{t \to \infty} \gamma(t) = \sqrt{\lambda_{\text{min}}}$ and $\lim_{t \to \infty} \gamma(t) = \sqrt{\lambda_{\text{max}}}$.

An interesting consequence of these two properties is the following asymptotic behavior:

**Corollary 8.** If $\lim_{t \to \infty} \lambda(t) = \lambda_t$, then

$$\lim_{t \to \infty} \omega(t) = \lim_{t \to \infty} \gamma(t) = \sqrt{\lambda_t} \quad (91)$$

**Proof.** We only consider the case of $\omega$, the proof for $\gamma$ following the same derivation. By definition of the limit, for any $\varepsilon > 0$, there exists $t_0 > 0$ large enough so that $\forall t > t_0$, $|\lambda(t) - \lambda_t| < \varepsilon$. Next, remark that the time-shifted function $\omega(t) := \omega(t + t_0)$ is a solution of the equation $\dot{\omega}(t) = \omega(t)^2 - \lambda$, where $\forall t > 0$, $\lambda(t) = \lambda_t(t_0) \in [\lambda_t - \varepsilon, \lambda_t + \varepsilon]$. Property 6 then shows that $\omega(t + t_0) \in [\sqrt{\lambda_t - \varepsilon}, \sqrt{\lambda_t + \varepsilon}]$ for all $t > 0$. As a consequence, for any $\varepsilon > 0$:

$$\sqrt{\lambda_t - \varepsilon} \leq \liminf_{t \to \infty} \omega(t) \leq \limsup_{t \to \infty} \omega(t) \leq \sqrt{\lambda_t + \varepsilon} \quad (92)$$

Taking the limit as $\varepsilon \to 0$, we conclude that the limit of $\omega$ as $t \to \infty$ exists and is equal to $\sqrt{\lambda_t}$.

We conclude from the above properties that the solutions $\zeta$ and $\xi$ to Equation (19) are well-defined.

**C. Convergent component of motion**

The component $\zeta$ corresponding to the damping $\gamma$ is subject to the differential equation:

$$\dot{\zeta} = -\gamma \zeta + (\lambda - \eta) \quad (93)$$

The general solution to this equation is given by:

$$\zeta(t) = \left( \zeta(0) + \int_0^t e^{-\Gamma(s)}(\lambda(s)\eta(s) - \eta)\text{d}s \right) e^{-\Gamma(t)} \quad (94)$$

where $\Gamma$ is the antiderivative of $\gamma$ such that $\Gamma(0) = 0$, i.e., $\Gamma(t) = \int_0^t \gamma(s)\text{d}s$. It satisfies the following two identities:

$$\frac{\text{d} \Gamma}{\text{d} t} = \gamma e^{\Gamma}, \quad \frac{\text{d}^2 \Gamma}{\text{d} t^2} = (\gamma + \gamma^2)e^{\Gamma} = \lambda e^{\Gamma}$$

The asymptotic behavior of $\zeta$ is tied to that of the two inputs $\lambda$ and $\eta$ of the inverted pendulum:

**Property 9.** Consider an input function $\lambda(t), \eta(t)$ such that $\lim_{t \to \infty} \lambda(t) = \lambda_t$ and $\lim_{t \to \infty} \eta(t) = \eta_t$, and let $\gamma$ denote any solution to (17). Then, the solution $\zeta$ of (93) satisfies:

$$\lim_{t \to \infty} \zeta(t) = \sqrt{\lambda_t r - \frac{g}{\lambda_t}} = \sqrt{\lambda_t \xi_t} \quad (96)$$

**Proof.** By Corollary 8, $\lim_{t \to \infty} \gamma(t) = \sqrt{\lambda_t}$, therefore its antiderivative $\Gamma(t)$ satisfies $\lim_{t \to \infty} \Gamma(t)/t = \sqrt{\lambda_t}$ as well. In particular, $\Gamma(t)$ diverges to $\infty$, so that:

$$\zeta(t) \sim e^{-\Gamma(t)} \int_0^t e^{\Gamma(s)}(\lambda(s)\eta(s) - \eta)\text{d}s \quad (97)$$

where the notation $f \sim g$ means that the ratio $f/g$ goes to 1 as $t \to \infty$. Applying l’Hôpital’s rule, we conclude that:

$$\lim_{t \to \infty} \zeta(t) = \lim_{t \to \infty} \frac{\int_0^t e^{\Gamma(s)}(\lambda(s)\eta(s) - \eta)\text{d}s}{e^{\Gamma(t)}} = \lim_{t \to \infty} \frac{e^{\Gamma(t)}(\lambda(s)\eta(s) - \eta)}{\gamma(t)e^{\Gamma(t)}} = \frac{\lambda_t \eta_t - \eta}{\sqrt{\lambda_t}} \quad (98)$$

**D. Divergent component of motion**

The general solution (21) of the divergent component of motion is based on the antiderivative $\Omega(t) = \int_0^t \omega(t)\text{d}t$. Recalling from Property 6 that $\omega \in [\sqrt{\lambda_{\text{min}}}, \sqrt{\lambda_{\text{max}}}]$, we see that $\Omega$ grows at least linearly. Therefore, as long as $\lambda$ and $\eta$ remain bounded, the integral $\int_0^t e^{-\Omega(s)}(\lambda(s)\eta(s) - \eta)\text{d}s$ is well-defined and finite. Let us now prove Property 1.

**Proof.** The proof is very similar to that of Property 9. The solution (21) with $\xi(t) = \xi_t$ from Equation (22) becomes:

$$\xi(t) = e^{\Omega(t)} \int_t^\infty e^{-\Omega(s)}(\lambda(s)\eta(s) - \eta)\text{d}s \quad (99)$$

Applying l’Hôpital’s rule, we conclude by Corollary 8 that:

$$\lim_{t \to \infty} \xi(t) = \lim_{t \to \infty} \frac{e^{\Omega(t)}(\lambda(t)\eta(t) - \eta)}{\omega(t)e^{\Omega(t)}} = \frac{\lambda_t \eta_t - \eta}{\sqrt{\lambda_t}} \quad (100)$$

That is to say, similarly to the convergent component of motion, the solution $\xi$ of (20) converges to $\sqrt{\lambda_t} \xi_t$. \[\square\]
E. Proof of Property 2

Proof of the ⇒ implication. Let \( \lambda(t), r(t) \) denote a capture input from \( \mathcal{T}_{x_i, x_i} \), with \( x(t) \) the smooth trajectory resulting from this input via the equation of motion (5). Its boundary values \( x(0) = x_1 \) and \( x(\infty) = x_2 \) being bounded, this trajectory must be bounded as well. As \( \xi(t) \) is always bounded by Property 9, this in turns implies that its the divergent component \( \xi(t) \) is bounded, and must therefore satisfy Equation (25) by Property 1. Next, let us denote by \( \lambda_i, r_i \) the limits of \( \lambda(t), r(t) \) as time goes to infinity. Using Properties 9 and 1, the two components converge to:

\[
\lim_{t \to \infty} \xi(t) = \lim_{t \to \infty} \xi(t) = \sqrt{\xi_i} \left( r_i - \frac{g}{\xi_i} \right) \tag{101}
\]

Recalling from Corollary 8 that \( \gamma \) and \( \omega \) converge to \( \sqrt{\xi_i} \), we can take the limit in the mapping (10)–(12):

\[
\lim_{t \to \infty} c(t) = \lim_{t \to \infty} \xi(t) + \xi(t) + \omega(t) = r_i - \frac{g}{\xi_i} = c_i \tag{102}
\]

Therefore, \( \lambda_i = \lambda_i(c_i) \) and \( r_i = r_i(c_i) \). \( \square \)

Proof of the ⇐ implication. Reciprocally, assuming (i)–(iii), Equation (101) holds again by Properties 9–1 and Corollary 8. Furthermore,

\[
\lim_{t \to \infty} \dot{c}(t) = \lim_{t \to \infty} \frac{-\omega(t)\xi(t) + \gamma(t)\xi(t)}{\gamma(t) + \omega(t)} = 0 \tag{103}
\]

Thus, the pendulum driven by \( \lambda(t), r(t) \) converges to the static equilibrium \( x_i \). \( \square \)

F. Note on fixed-CoP strategies

When the CoP input is stationary, i.e. in a point-foot model, the Gram determinant \( G = ((c - r) \times \dot{c}) \cdot g \) becomes invariant.

Short proof. Take the cross-product of Equation (5) with \( \dot{c} \). Then, the scalar product of the resulting expression with \( g \) yields \( ((c - r) \times \dot{c}) \cdot g = 0 \). Conclude by noting that this formula is the time derivative of \( ((c - r) \times \dot{c}) \cdot g \). \( \square \)

There are two possible outcomes: either \( G = 0 \), which means the three vectors are coplanar and the robot may stabilize using a 2D strategy [8, 9, 10]; or \( G \neq 0 \) and it is impossible to bring the system to an equilibrium where \( \dot{c} = 0 \). This shows simultaneously two properties: first, that sagittal 2D balance control is the most general solution for point-foot models, and second, that these models have a very limited ability to balance, as they need to re-step at the slightest lateral change in linear momentum. The ability of flat-footed bipeds to absorb these perturbations (to some extent) without stepping comes from continuous CoP variations.

APPENDIX B

NUMERICAL OPTIMIZATION COMPLEMENT

In this Appendix, we first recall terminology and state-of-the-art algorithms for numerical optimization. We essentially rewrite treatment from [28] for double-sided inequality constraints. We then detail all tailored operations mentioned in Section IV.

A. Definitions and notations

Consider the optimization problem:

\[
\begin{align*}
\text{minimize } & f(x) \\
\text{subject to } & l \leq h(x) \leq u
\end{align*} \tag{104a}
\]

where \( f \) and \( h \) are smooth functions, \( f \) being 1-dimensional and \( h \) \( m \)-dimensional. Lower and upper bound constraints are represented by vectors \( l, u \in \mathbb{R}^m \), with equality constraints specified by taking \( l_j = u_j \). A point \( x \) is feasible if it satisfies all constraints. For a given \( x \), we say the \( j \)th constraint is active at its lower (resp. upper) bound when \( h_j(x) = l_j \) (resp. \( h_j(x) = u_j \)). We denote by:

\[
\begin{align*}
E := & \{ j \in [1, m], l_j = u_j \} \tag{105} \\
A(x)^- := & \{ j \notin E, h_j(x) = l_j \} \tag{106} \\
A(x)^+ := & \{ j \notin E, h_j(x) = u_j \} \tag{107}
\end{align*}
\]

These three sets are disjoint. For a set of indexes \( S \) and a matrix \( M \), we define \( M_S \) the matrix made of the rows of \( M \) whose indexes are in \( S \) (this notation also applies to vectors). The Lagrangian of the problem is defined as \( \mathcal{L}(x, \lambda, \lambda^+) := f(x) + \lambda^T (h(x) - l) + \lambda^{+T} (h(x) - u) \) where \( \lambda^+, \lambda^- \in \mathbb{R}^m \) are the Lagrange multipliers, \( \nabla_x g \) and \( \nabla^2_x g \) are respectively the gradient and Hessian of a function \( g \) with respect to \( x \). We note \( \lambda := \lambda^+ + \lambda^- \). We can work with it instead of \( \lambda^- \) and \( \lambda^+ \) (see [51, §4.3.5]).

The Karush–Kuhn–Tucker (KKT) conditions give necessary conditions on \( x \) and \( \lambda \) for \( x \) to be a minimizer of Problem (104) (see [28, chap. 12]). They are often used as termination conditions in solvers.

B. Active-set method for Quadratic Programming

When the objective \( f \) is quadratic, \( f(x) = \frac{1}{2} x^T Q x + q^T x \) with \( Q \) symmetric positive semidefinite and \( h \) linear, \( h(x) = C x \) for some matrix \( C \), Problem (104) is a (convex) Quadratic Program with Inequality constraints (QPI). One of the main approaches to solve it is the active-set method. This method iteratively discovers the set of constraints active at the solution\(^{18} \) by solving at each iteration \( k \) the following Quadratic Program with only Equality constraints (QPE):

\[
\begin{align*}
\text{minimize } & \frac{1}{2} p^T Q p + (Q x_k + q)^T p \\
\text{subject to } & C_{W_k} p = 0
\end{align*} \tag{108a}
\]

where \( W_k \) is a set of indexes. The solution \( p^* \) to this QPE is used to determine the next iterate \( x_{k+1} \).

Unlike QPIs, QPEs admit analytical solutions as their KKT conditions reduce to a linear system. For a given \( x_k \), we can retrieve \( \lambda \) with:

\[
\lambda_{A_k} = -C_{A_k}^T \nabla_{x}^T f(x_k), \quad \lambda_{i} = 0, \forall i \notin A_k \tag{109}
\]

where \( \square \) denotes the Moore-Penrose pseudo-inverse and \( A_k = A(x_k)^- \cup A(x_k)^+ \cup E \) is the active set at \( x_k \), i.e. the set of constraints that are active at this point.

The active-set method for convex QPIs is given in Algorithm 1. See [28, chap. 16] for more details on this method.

\(^{18} \) We ignore here for the sake of simplicity a subtlety arising when active constraints are linearly dependent.
Algorithm 1 Active-set algorithm for convex QPI

Given a feasible point \( x_0 \)

Let \( \mathcal{W}_0 = \mathcal{A}(x_0)^\ominus \) \( \mathcal{W}_0^+ = \mathcal{A}(x_0)^+ \)

for \( k = 0, 1, 2, \ldots \) do

Compute \( p \) from (108) with \( \mathcal{W}_k = \mathcal{W}_k^- \cup \mathcal{W}_k^+ \cup \mathcal{E} \)

if \( p = 0 \) then

Compute \( \lambda \) using Equation (109)

if \( x \) and \( \lambda \) verify the KKT conditions then

return the solution \( x_k = x \)

else

Choose \( j \) such that \( \lambda_j \) violates the KKT conditions

\( x_{k+1} = x_k \), \( \mathcal{W}_k^- = \mathcal{W}_k^- \{ j \}, \mathcal{W}_k^+ = \mathcal{W}_k^+ \{ j \} \)

if \( \alpha \) and \( \lambda \) are acceptable, see [28, chap. 18] for details.

end if

end while

C. Sequential Quadratic Programming

Sequential quadratic programming (SQP) is an iterative optimization technique for solving general constrained problems such as (104). At each iteration \( k \), a QP approximation of (104) is formed and solved:

\[
\begin{align*}
\text{minimize} & \quad f(x_k) + \nabla_x f(x_k)^T p + \frac{1}{2} p^T B_k p \\
\text{subject to} & \quad l - h(x_k) \leq \nabla_x h(x_k)^T p \leq u - h(x_k)
\end{align*}
\]

(110a)

(110b)

where \( B_k \) can be \( \nabla^2_{xx} L(x_k, \lambda_k) \) or, for faster computations, some positive-definite approximation of it.

Algorithm 2 Line search SQP

Given a stepping parameter \( \tau \in (0, 1) \)

Choose \( (x_0, \lambda_0) \)

while the KKT conditions are not satisfied do

Compute \( p \) from (110)

Let \( \lambda \) be the corresponding multiplier and \( p_\lambda = \lambda - \lambda_k \)

\( \alpha = 1 \)

while \( \alpha p \) does not yield an acceptable step do

\( \alpha = \tau \alpha \)

end while

\( x_{k+1} = x_k + \alpha p \), \( \lambda_{k+1} = \lambda_k + \alpha p_\lambda \)

end while

D. Cost matrix of the unconstrained least squares problem

Consider the active set \( \mathcal{W} \) for a given SQP iteration. Starting at the first constraint, count the number of active constraints (possibly 0 if the first constraint is not active), then \( j \) the number of following consecutive inactive constraints, \( a \), the number of following active constraints, \( n = a + j + 1 \). The set \( \mathcal{W} \) is then fully described by the sequence \( (a_0, a_1, a_2, \ldots, a_j, a_p) \), where only \( a_0 \) and \( a_p \) can be 0. Note that \( \sum_k a_k + \sum_k j_k = n + 1 \), and let us define \( n_\mathcal{W} := \sum_k a_k \). For example, if \( \mathcal{W} = \{1, 2, 6, 9, 10, 11, 13, 14\} \) for \( n = 15 \) optimization variables, we get the sequence \( (2, 3, 1, 2, 3, 1, 2, 0) \) and \( n_\mathcal{W} = 8 \). The constraint matrix \( C_{\mathcal{W}} \) is then the \( n_\mathcal{W} \times n \) matrix:

\[
C_{\mathcal{W}} = \begin{bmatrix}
C_0 & 0_{a_1,j_1-1} & C_1 & 0_{a_2,j_2-1} & C_2 & \cdots & 0_{a_p,j_p-1} & C_p
\end{bmatrix}
\]

(111)

where \( 0_{m,q} \) is the \( m \times q \) zero matrix, while \( C_0 \), \( C_k \) \( (k \in [1, p-1]) \) and \( C_p \) are respectively \( a_0 \times a_0 \), \( a_k \times (a_k + 1) \) and \( a_p \times a_p \) matrices (\( C_0 \) and \( C_p \) can be empty) of the form:

\[
C_0 = \begin{bmatrix}
1 & -1 & 1 \\
\vdots & \ddots & \ddots & \ddots \\
-1 & 1 & \ddots & -1 & 1 \\
1 & \cdots & \cdots & \cdots & \cdots & 1
\end{bmatrix}
\]

(112)

Denoting by \( \mathbf{1}_a \) the vector of size \( a \) filled with ones, the nullspace projection matrix for the active set \( \mathcal{W} \) is:

\[
N_\mathcal{W} = \begin{bmatrix}
0_{a_0,j_1-1} & \mathbf{1}_{i_1-1} & 1_{a_1+1} & \mathbf{1}_{i_2-1} & 1_{a_2+1} & \cdots & \mathbf{1}_{i_p-1} & 0_{a_p,j_p-1}
\end{bmatrix}
\]

(113)

Noting that \( C_k \mathbf{1}_{a_k+1} = 0_{a_k, 1} \), we can directly verify that \( C_{\mathcal{W}} N_\mathcal{W} = 0 \). The matrix \( N_\mathcal{W} \) is \( n \times (n - n_\mathcal{W}) \) full column rank. It is thus a basis of the nullspace of \( C_{\mathcal{W}} \).

Computing the product \( MN_\mathcal{W} \) for a given matrix \( M \) does not actually require to perform any multiplication: multiplying by \( \mathbf{1} \) amounts simply to the summation of columns of \( M \). Likewise, \( N_\mathcal{W} z \) just requires to copy the elements of \( z \). It is not thus necessary to form \( N_\mathcal{W} \), and \( \mathbf{T} \) can be obtained by \( \sum_k a_k \) \( \mathbf{z} \) vector additions. Taking into account the tridiagonal structure of \( J \), this can be done in \( O(n) \).

E. Computation of Lagrange multipliers

The computation of the Lagrange multipliers, needed to check KKT conditions, relies on the pseudoinverse of \( C_{\mathcal{W}} \) (see
Due to its block structure, expressing the latter is done by finding the pseudo-inverse for each $C_k$:

$$
C_W^t = \begin{bmatrix}
C_0^{-1} & 0_{1,1} & 0_{1,2} & \cdots & 0_{1,\nu}
\end{bmatrix}
$$

where $\Box^t$ denotes the pseudo-inverse. It can be verified that

$$
C_0^{-1} = \begin{bmatrix}
1 & \vdots & \vdots & \vdots & 1
\end{bmatrix}, \quad C_p^{-1} = \begin{bmatrix}
-1 & \cdots & -1 & 1
\end{bmatrix}
$$

$$
C_k^t = \frac{1}{a_k + 1} \begin{bmatrix}
-ak - (ak - 1) & \cdots & -ak - 2 & 1
1 & \cdots & -ak - 2 & -1
1 & 2 & \cdots & 3 & \cdots & -1
1 & 2 & 3 & \cdots & ak
\end{bmatrix}
$$

The computation of Lagrange multipliers can thus be done in $O(n^2)$ without forming the pseudo-inverse explicitly.

F. QR decomposition of the cost matrix $T$

The decomposition can be performed in two steps: first the QR decomposition $J_W = Q_W R_W$ of $J_W := JN_W$, followed by the QR decomposition $[\mu_j^T N_W] = [q_j^T] R$. Combining these two yields $T$ by:

$$
\begin{bmatrix}
\mu_j^T N_W \\
JN_W
\end{bmatrix} = \begin{bmatrix}
1 & 0 \\
Q_W
\end{bmatrix} \begin{bmatrix}
\mu_j^T N_W \\
R_W
\end{bmatrix} = \begin{bmatrix}
1 & 0 \\
Q_W
\end{bmatrix} \begin{bmatrix}
q_j^T \\
Q_W Q_2
\end{bmatrix} R
$$

The matrix $[\mu_j^T N_W]$ is upper Hessenberg, so that its QR decomposition is computed in $O(n^2)$ [29, Chapter 5].

The decomposition of $J_W$ can be achieved in $O(n)$ by taking advantage of its structure. To avoid going through several corner cases, we sketch informally how this is done with the help of the example in Figure 12 (more details can be found in [52]). Because the sum of three non-zero elements on any row (ignoring the first) of $J$ is zero, a careful study reveals that $J_W$ is made of $p$ tridiagonal blocks, one for each group of consecutive inactive constraints. Blocks $j$ and $j + 1$ are separated by $a_j - 1$ rows of zeros, and the last column of the first block is aligned with the first column of the second block (Figure 12, left). We can perform QR decompositions for each blocks separately and denote by $Q_W$ the product of all orthogonal matrices. All blocks, except possibly the last one, have a rank equal to their row size minus one, so that the triangular factor of the decomposition has zeros on its last line (Figure 12, middle left). Multiplying by a permutation matrix $P_W$, all zero rows can be moved to the bottom, and we get a quasi-tridiagonal matrix (Figure 12, middle right). The latter can be made triangular with a last tridiagonal QR decomposition in $O(n)$ (Figure 12, right).

G. Search for a feasible initial point

Denoting by $l_x$ and $u_x$ the bounds corresponding to $C_x$, the set $Z := \{ \phi \in \mathbb{R}^n, l_z \leq C_z \phi \leq u_z \}$ is a zonotope equal to

$$
Z = L_Z l_Z + L \text{diag}(u_Z - l_Z) [0, 1]^n
$$

where $L_Z := C_Z^{-1}$ is the $n \times n$ lower triangular matrix with all coefficients equal to 1. Feasible points for the whole problem are those in $Z$ such that $\omega_{\text{min}} \leq \varphi_n \leq \omega_{\text{max}}$.

Consider the point:

$$
\varphi_0(a) := L_Z l_Z + \sum a(u_i - l_i) L_i
$$

where $L_i$ is the $i$th column of $L$. This point is in $Z$ for any value $a \in [0, 1]$. Its last component $\varphi_1(a)$ is an increasing linear function $\varphi_1(a) = s_l + a s_d$ with $s_l = \sum j l_j$ and $s_d = \sum (u_j - l_j) \geq 0$. Let us denote by $a^-$ and $a^+$ the two values such that $\varphi_1(a^-) = \omega_{\text{min}}$ and $\varphi_1(a^+) = \omega_{\text{max}}$. The linear constraints of the capture problem are then feasible if and only if $[a^-, a^+] \subset [0, 1] \neq \emptyset$. In this case, any $a$ in this intersection yields a feasible point $\varphi(a)$ for the problem, for instance the middle value $a_m := \frac{1}{2}(\max(a^- , 0) + \min(a^+, 1))$. We can therefore initialize our SQP with:

$$
\varphi_0 = \varphi_1(a_m) \quad \lambda_0 = 0
$$

which is guaranteed to be a feasible point.

Appendix C

External Optimization Details

A. Feasibility of outer-optimization problems

There are three ways a bad choice of $\alpha$ can yield an unfeasible capture problem (59):
(a) The bounds of the inequality (59c) are such that \( \omega_{i, \text{min}} > \omega_{i, \text{max}} \). Recall that these bounds represent feasibility of the CoP \( r_1 \), which involves \( \alpha \) by Equation (77).

(b) The intersection between the nonlinear inequality constraint (59b) and the polytope (59c)–(59d) is empty. The influence of \( \alpha \) on this comes from Equation (76).

(c) The right cylinder given by the linear constraint (59c) does not intersect the zonotope (59d). The role of \( \alpha \) in this comes once again from its influence on Equation (77).

Case (c) can be caught efficiently before solving the capture problem (Appendix B-G). While anticipating (b) is still an open question, case (a) can be avoided altogether thanks to a more careful treatment of CoP inequality constraints. Recall from Equation (43) that:

\[
[\alpha H r_i^{xy} + (1 - \alpha)p - H e_i^{xy}] \omega_i \geq H c_i^{xy}
\]  

Let us rewrite this inequality as:

\[
(u - \alpha v) \omega_i \geq w
\]  

To avoid corner cases, let us extend the three vectors \( u, v \) and \( w \) with two additional lines:

- **Line 1:** \( u = 1, v = 0 \) and \( w = \omega_{i, \text{min}} \)
- **Line 2:** \( u = -1, v = 0 \) and \( w = -\omega_{i, \text{max}} \)

Next, note that the two sets \( \mathcal{A}_{\text{min}}(\alpha) := \{i, u_i - \alpha v_i \geq 0\} \) and \( \mathcal{A}_{\text{max}}(\alpha) := \{i, u_i - \alpha v_i \leq 0\} \) are such that:

\[
\omega_{i, \text{min}}(\alpha) = \max \left\{ \frac{w_i}{u_i - \alpha v_i}, i \in \mathcal{A}_{\text{min}}(\alpha) \right\}
\]  

\[
\omega_{i, \text{max}}(\alpha) = \min \left\{ \frac{w_i}{u_i - \alpha v_i}, i \in \mathcal{A}_{\text{max}}(\alpha) \right\}
\]  

Using a technique reminiscent of Fourier-Motzkin elimination, a necessary and sufficient condition for \( \omega_{i, \text{min}} \leq \omega_{i, \text{max}} \) is then that, for all pairs \((i, j) \in \mathcal{A}_{\text{min}}(\alpha) \times \mathcal{A}_{\text{max}}(\alpha)\),

\[
u_i w_j - u_j w_i \leq \alpha (v_i w_j - v_j w_i)
\]

These inequalities are of the form \( \tilde{u} \alpha \geq \tilde{v} \) and can therefore be reduced, similarly to Equations (44)–(45), into a single interval \([\alpha_{\text{min}}, \alpha_{\text{max}}]\). In this interval, it is guaranteed that \( \omega_{i, \text{min}} \leq \omega_{i, \text{max}} \) and failure case (a) is avoided.

The subtlety to notice here is that the index sets \( \mathcal{A}_{i,j}(\alpha) \) change when \( \alpha \) crosses the roots \( u_i/v_i \). (Note that there are few such roots in practice, e.g. at most six with rectangular foot soles.) We take this phenomenon into account in the overall Algorithm 3.

---

**Algorithm 3** Computation of \( \alpha \) feasibility intervals

**Input:** vectors \( u, v \) and \( w \)

**Output:** set \( \mathcal{I} \) of feasible intervals \([\alpha_{\text{min}}, \alpha_{\text{max}}]\)

\[
\mathcal{I} \leftarrow \emptyset
\]

\[\mathcal{R} \leftarrow \{r_j = u_j/v_j | r_j \in (0, 1)\}\]

**for** \((r_j, r_{j+1})\) consecutive roots in \( \text{SORT}(\mathcal{R}) \) **do**

\[
(\alpha_{\text{min}}, \alpha_{\text{max}}, \alpha) \leftarrow (r_{2j}, r_{2j+1}; 1/2(r_{2j} + r_{2j+1}))
\]

Compute index sets \( \mathcal{A}_{\text{min}}(\alpha) \) and \( \mathcal{A}_{\text{max}}(\alpha) \)

Reduce \([\alpha_{\text{min}}, \alpha_{\text{max}}]\) using (125) with \( \mathcal{A}_{\text{min}}(\alpha) \), \( \mathcal{A}_{\text{max}}(\alpha) \)

\[
\mathcal{I} \leftarrow \mathcal{I} \cup \{(\alpha_{\text{min}}, \alpha_{\text{max}})\}
\]

**end for**

**return** \( \mathcal{I} \)

---

Stéphane Caron is a CNRS researcher at LIRMM, CNRS-University of Montpellier, France. He received the M.S. degree from the Ecole Normale Supérieure of Paris, and the Ph.D. in 2016 from the University of Tokyo, Japan. He was then involved in the COMANOID project, which developed industrial locomotion and manipulation applications of humanoids that were demonstrated on-site at Airbus. His research interests include whole-body control, force control and balance control for legged robots.

Adrien Escande received the M.S. degree in 2005 from Ecole des Mines de Paris, France, and the Ph.D. degree in 2008 in robotics from Université d’Évry Val-d’Essonne, France, after spending three years at the CNRS–AIST Joint Robotics Laboratory (JRL), Tsukuba, Japan. He then worked as a research scientist in CEA–LIST at Fontenay-aux-Roses, France, until the end of 2012 and is now back at JRL. His research interests include whole-body planning and control for humanoid robots and numerical optimization for robotics.

Leonardo Lanari received the Ph.D. degree in control engineering from Sapienza University of Rome, Italy, in 1992.

He is currently with the Department of Computer, Control and Management Engineering (DIAG), Sapienza University of Rome, as an Associate Professor of automatic control. His research interests are in the general area of planning and control of robotic systems, with an emphasis on humanoid control and robots with elastic joints and links.

Bastien Mallein is maître de conférences at the Laboratoire Analyse, Géométrie et Applications (LAGA). He was born in Beauvais (France), and studied Mathematics at the Ecole Normale Supérieure of Paris from 2008 to 2012. He obtained a PhD in Probability from UPMC in 2015, under the supervision of Zhan Shi. In 2016, he was a postdoctoral researcher at the Universität Zürich. He became aware of robotics during a hike in southern Tarn.