Partial Yaw Moment Compensation using an Optimization-based Multi-Objective Motion Solver

Rafael Cisneros[1,*], Mehdi Benallegue[1], Mitsuharu Morisawa[1],
Eiichi Yoshida[2], Kazuhito Yokoi[3] and Fumio Kanehiro[1]

***Abstract*— Any arbitrary motion generated by a humanoid robot produces a yaw moment which may exceed the one created by the friction between its feet and the ground, inducing a yaw rotation that deviates the robot from its desired path. This paper describes an on-line compensation scheme for the yaw moment of a humanoid robot about the Zero Moment Point (ZMP), formulated as a task of a Quadratic Program (QP) solving for multiple weighted objectives and constraints. This allows to use the motion of every single link of the robot to contribute to the compensation, according to the relative weight of other primary tasks that it should perform. Within the proposed approach the yaw moment is partially compensated; that is, mostly when exceeding a predefined threshold, allowing to slow down the residual motion of the links triggered by the compensation.**

***Index Terms*— Yaw Moment Compensation, Angular Momentum, Quadratic programming, Humanoid robots**

I. Introduction

One of the most important and current trends in the research of humanoid robots is to allow them to evolve in an unstructured environment, such that they can be successfully introduced into the industry. Aiming to this purpose, it is important that the robot be able to locate itself with respect to the environment in order to travel to the desired position, as well as to successfully manipulate the surrounding objects in order to perform some desired tasks.

To do that, robots can either use vision and rely solely on Self Localization and Mapping (SLAM) routines, or use a Laser Range Finder Sensor (or an RGBD camera) generating a poorly dense and noisy point cloud that needs to be processed for identification, or used also for SLAM. Either way, the resulting algorithms are computationally expensive, preventing their use in real-time [1]; that is, at each controller cycle. It is then expected that the motion of the robot be reliable, by experiencing a minimum deviation from its desired path, either when walking or performing any motion *in situ*, such that self-localization algorithms be able to run with a lower frequency, performing only small corrections.

A common source of deviation of the humanoid robot from the desired path is a yaw rotation, induced by a generated yaw moment that exceeds the one created by the friction between feet and ground (during bipedal locomotion) [2]. Any arbitrary motion performed by the humanoid robot generates yaw moment, unless it is adequately compensated [3].

[1] Humanoid Research Group, AIST, Tsukuba, Japan.
[2] CNRS-AIST JRL, UMI3218/RL, Tsukuba, Japan.
[3] AIST, Tsukuba, Japan.
[*] Corresponding author E-mail: rafael.cisneros@aist.go.jp

Humans instinctively and actively control their arm swing and torso motion to counteract the yaw moment [4] [5], suggesting that it can be performed in a similar way. In the literature, it is possible to find several methods for compensating the yaw moment, which generally fall into one of the following strategies: (a) arm swinging strategy [6] [7] [8] [9] [10], (b) yaw-axis waist joint strategy [11] [12] [2], (c) a combination of the above [13], and (d) swing leg trajectory strategy [14]. Each of those methods is targeted to some specific joints, which are assigned the main task of compensating the yaw moment. However, if the humanoid robot is expected to use some of those joints for an alternate task, then the corresponding compensation approach cannot be used; that is, they don't provide a general solution for a multi-task humanoid robot.

A different approach was proposed by us in a previous paper, that did not specifically use the arm swinging strategy or the yaw-axis joint strategy, but a general offline algorithm that allowed a full compensation to be performed on an arbitrary set of before-hand selected "free" joints [15]. However, this generally led to an unbounded joint motion, unfeasible due to the presence of joint limits. This approach was later improved as an online controller that didn't attempt to compensate the yaw moment fully, but partially, by establishing a threshold value for the measured yaw moment above which the compensation was done by accelerating the before-hand selected "free" joints. Then, once the yaw moment had been taken below the threshold value, the joints would be able to slow down at a lower rate, fulfilling the net yaw moment requirement and, at the same time, keeping the joint values bounded and within the limits without residual drift, or "net" joint displacement [3]. A drawback of our previous approaches is that the selected set of "free" joints are dedicated to compensating the yaw moment.

In this paper, we reformulate and improve the previous controller as a *task* of a Quadratic Programming (QP) based multi-objective motion solver; specifically, the one described with some detail in [16]. The main advantage of this framework is that the motion of every single link of the robot can contribute to the yaw moment compensation, in a degree dictated by the *weight* of this task relative to the weight of other tasks that the robot has to perform. Additionally, some constraints like the joint limits (and others that allow for a feasible dynamical motion) can be taken into account, which couldn't be done within the previous approach.

This paper is organized as follows:

- Section II summarizes the motion solver framework without entering into details, with the purpose of introducing notation and concepts.
- Section III describes the main idea behind the yaw moment compensation, its advantage of adding the corresponding task to the motion solver and its implementation into the new framework. It represents the main contribution of this paper.
- Section IV presents some dynamic simulation results.
- Section V concludes this paper.

II. Multi-objective Control

A. Humanoid Robot Dynamics

Let us consider a humanoid robot with $n+6$ degrees of freedom (dof) such that its configuration can be described as $\boldsymbol{q} = (\boldsymbol{p_B}, \boldsymbol{R_B}, \boldsymbol{q_\theta})$, where $\boldsymbol{p_B} \in \mathbb{R}^3$ is the position of the floating base (e.g., the waist), $\boldsymbol{R_B} \in SO(3)$ represents its orientation and $\boldsymbol{q_\theta} \in \mathbb{R}^n$ comprises all the joint angles. The *configuration velocity* of the robot, $\boldsymbol{\alpha} \in \mathbb{R}^{n+6}$ is given by

$$\boldsymbol{\alpha} = \left[\begin{array}{ccc} \boldsymbol{v_B^T} & \boldsymbol{\omega_B^T} & \boldsymbol{\dot{q}_\theta^T} \end{array} \right]^T. \tag{1}$$

The time derivative of the configuration velocity, $\dot{\boldsymbol{\alpha}} \in \mathbb{R}^{n+6}$, is the configuration acceleration.

The dynamical model of the humanoid robot is written as

$$\boldsymbol{M}(\boldsymbol{q})\dot{\boldsymbol{\alpha}} + \boldsymbol{C}(\boldsymbol{q},\boldsymbol{\alpha})\boldsymbol{\alpha} + \boldsymbol{g}(\boldsymbol{q}) = \boldsymbol{u} + \boldsymbol{u_e}, \tag{2}$$

where $\boldsymbol{M}(\boldsymbol{q}) \in \mathbb{R}^{(n+6)\times(n+6)}$ represents the mass matrix of the robot, $\boldsymbol{C}(\boldsymbol{q},\boldsymbol{\alpha}) \in \mathbb{R}^{(n+6)\times(n+6)}$ is a matrix factorization accounting for the Coriolis and centripetal effects such that $\dot{\boldsymbol{M}}(\boldsymbol{q},\boldsymbol{\alpha}) - 2\boldsymbol{C}(\boldsymbol{q},\boldsymbol{\alpha})$ is a skew-symmetric matrix, $\boldsymbol{g}(\boldsymbol{q}) \in \mathbb{R}^{n+6}$ is the vector of gravitational effects, $\boldsymbol{u_e} = \sum \boldsymbol{J_i^T}(\boldsymbol{q})\boldsymbol{F_i}$ corresponds to the vector of external generalized forces acting through the unilateral contacts with the environment, $\boldsymbol{F_i} \in \mathbb{R}^6$ is the i-th external wrench, $\boldsymbol{J_i}(\boldsymbol{q}) \in \mathbb{R}^{6\times(n+6)}$ is the absolute Jacobian of its point of application, and $\boldsymbol{u} \in \mathbb{R}^{n+6}$ is a vector of input generalized forces which includes actuated and unactuated dof (zero entries), the latter ones corresponding to the position and orientation of the floating base as we cannot exert directly a wrench on it [16].

B. Multi-objective motion solver

A humanoid robot can fulfill several *tasks* simultaneously, while satisfying kinematic and dynamic *constraints*. We use a QP solver to minimize the tracking error for several weighted tasks by computing an optimal reference configuration acceleration, $\dot{\boldsymbol{\alpha}}_{\boldsymbol{r}}$, and a feasible reference of external forces $\boldsymbol{u_{e,r}}$ parameterized by a vector $\boldsymbol{\rho_r}$[1], while satisfying linear equality, inequality and bounding constraints; that is, to solve

$$\left[\begin{array}{c} \dot{\boldsymbol{\alpha}}_{\boldsymbol{r}} \\ \boldsymbol{\rho_r} \end{array} \right] = \arg \min_{\boldsymbol{x}} \frac{1}{2} \left\| \boldsymbol{W} \left(\boldsymbol{A}_{\text{ob}}\boldsymbol{x} - \boldsymbol{b}_{\text{ob}} \right) \right\|^2 + \frac{1}{2}\gamma \left\| \boldsymbol{x} \right\|^2, \\ \text{s.t.} \quad \boldsymbol{A}_{\text{eq}}\boldsymbol{x} = \boldsymbol{b}_{\text{eq}},\ \boldsymbol{A}\boldsymbol{x} \le \boldsymbol{b},\ \boldsymbol{l_b} \le \boldsymbol{x} \le \boldsymbol{u_b}, \tag{3}$$

where $\boldsymbol{x} = \left[\begin{array}{cc} \dot{\boldsymbol{\alpha}}_{\boldsymbol{r}}^{\boldsymbol{T}} & \boldsymbol{\rho_r^T} \end{array} \right]^T$ is the decision variable vector (the optimized output given by the QP), $\boldsymbol{W} = \text{blkdiag}\left(\boldsymbol{W_1}, \ldots, \boldsymbol{W_k}\right)$ is a block diagonal matrix made up of individual diagonal weighting matrices for each task [18] [17], and γ is a small weight (1E-4) introduced to minimize $\dot{\boldsymbol{\alpha}}_{\boldsymbol{r}}$ and $\boldsymbol{\rho_r}$ [19].

The matrices $\boldsymbol{A}_{\text{ob}}$, $\boldsymbol{A}_{\text{eq}}$, $\boldsymbol{A}$, and vectors $\boldsymbol{b}_{\text{ob}}$, $\boldsymbol{b}_{\text{eq}}$, $\boldsymbol{b}$, $\boldsymbol{l_b}$, $\boldsymbol{u_b}$ are made up by vertically concatenating the corresponding ones for each task or constraint.

C. Tasks

For the jth task, $\boldsymbol{A}_{\text{ob},\boldsymbol{j}}$ and $\boldsymbol{b}_{\text{ob},\boldsymbol{j}}$ are calculated as

$$\boldsymbol{A}_{\text{ob},\boldsymbol{j}} = \boldsymbol{J_{g,j}}(\boldsymbol{q}), \qquad \boldsymbol{b}_{\text{ob},\boldsymbol{j}} = \ddot{\boldsymbol{g}}_{\text{ob},\boldsymbol{j}} - \dot{\boldsymbol{J}}_{\boldsymbol{g,j}}(\boldsymbol{q},\boldsymbol{\alpha})\boldsymbol{\alpha}, \tag{4}$$

where, $\ddot{\boldsymbol{g}}_{\text{ob},\boldsymbol{j}}$ is an acceleration objective and $\boldsymbol{J_{g,j}}(\boldsymbol{q})$, $\dot{\boldsymbol{J}}_{\boldsymbol{g,j}}(\boldsymbol{q},\boldsymbol{\alpha})$ are the jth task Jacobian and its time derivative.

The acceleration objectives can be tracked with a PD and a feedforward term. For example, the posture task (in joint space) can be defined as $\ddot{\boldsymbol{g}}_{\text{ob}} = \ddot{\boldsymbol{q}}_{\boldsymbol{\theta},\text{ob}}$, while the position and orientation tasks of a link (or the position task of the CoM) in Cartesian space can be defined as $\ddot{\boldsymbol{g}}_{\text{ob}} = \dot{\boldsymbol{v}}_{\text{ob}}$ and $\ddot{\boldsymbol{g}}_{\text{ob}} = \dot{\boldsymbol{\omega}}_{\text{ob}}$, respectively[2], such that

$$\ddot{\boldsymbol{q}}_{\boldsymbol{\theta},\text{ob}} = \kappa_p \left(\boldsymbol{q_{\theta,d}} - \boldsymbol{q_\theta}\right) + \kappa_v \left(\dot{\boldsymbol{q}}_{\boldsymbol{\theta,d}} - \dot{\boldsymbol{q}}_{\boldsymbol{\theta}}\right) + \ddot{\boldsymbol{q}}_{\boldsymbol{\theta,d}}, \tag{5}$$

$$\dot{\boldsymbol{v}}_{\text{ob}} = \kappa_p \left(\boldsymbol{p_d} - \boldsymbol{p}\right) + \kappa_v \left(\boldsymbol{v_d} - \boldsymbol{v}\right) + \dot{\boldsymbol{v}}_{\boldsymbol{d}}, \tag{6}$$

$$\dot{\boldsymbol{\omega}}_{\text{ob}} = \kappa_p (\log\left\{\boldsymbol{R_d}\boldsymbol{R^T}\right\})^\vee + \kappa_v \left(\boldsymbol{\omega_d} - \boldsymbol{\omega}\right) + \dot{\boldsymbol{\omega}}_{\boldsymbol{d}}, \tag{7}$$

where κ_p and κ_v are scalar PD gains[3], d stands for the desired value and $(\cdot)^\vee : \mathbb{R}^{3\times3} \to \mathbb{R}^3$, such that if $\boldsymbol{S} = -\boldsymbol{S^T} \in \mathbb{R}^{3\times3}$ we have $\boldsymbol{S}\boldsymbol{x} = (\boldsymbol{S})^\vee \times \boldsymbol{x}$, $\forall \boldsymbol{x} \in \mathbb{R}^3$ [16].

D. Constraints

We can consider 4 basic types of constraints:

1) Underactuation / torque constraint: The underactuation constraint ensures the generation of a feasible motion for the floating base. The torque constraint ensures that the required torques are within the limitations of the actuators (minimum and maximum torques: $\underline{\boldsymbol{\tau}}$ and $\bar{\boldsymbol{\tau}}$).

Let us define a matrix $\boldsymbol{D}(\boldsymbol{q})$, such that $\boldsymbol{u_{e,r}} = \boldsymbol{D}(\boldsymbol{q})\boldsymbol{\rho_r}$. Then, according to (2), the underactuation and torque constraints can be, respectively, specified as

$$\left[\begin{array}{cc} \boldsymbol{M_B} & -\boldsymbol{D_B} \end{array} \right] \boldsymbol{x} = -\boldsymbol{C_B}\boldsymbol{\alpha} - \boldsymbol{g_B}, \tag{8}$$

$$\underline{\boldsymbol{\tau}} - \boldsymbol{C_j}\boldsymbol{\alpha} - \boldsymbol{g_j} \le \left[\begin{array}{cc} \boldsymbol{M_j} & -\boldsymbol{D_j} \end{array} \right] \boldsymbol{x} \le \bar{\boldsymbol{\tau}} - \boldsymbol{C_j}\boldsymbol{\alpha} - \boldsymbol{g_j}, \tag{9}$$

where the subscript B stands for the first 6 rows of the matrices $\boldsymbol{M}(\boldsymbol{q})$, $\boldsymbol{D}(\boldsymbol{q})$, $\boldsymbol{C}(\boldsymbol{q},\boldsymbol{\alpha})$ and $\boldsymbol{g}(\boldsymbol{q})$, while the subscript j stands for the remaining rows.

2) Joint limits constraints: Joint range and speed limits can be specified as done in [19].

[1]$\boldsymbol{\rho_r}$ is a coordinate vector with respect to a set of bases that span the volume of admissible forces in world coordinates. These bases are constructed by directing unit vectors along the edges of a pyramidal approximation of each contact's friction cone (see [17]).

[2]Notice that we have omitted the j subscript as it is obvious from the context that each task can be defined in a similar way.

[3]Actually, it is also possible to use diagonal matrices instead, i.e. $\boldsymbol{K_p}$ and $\boldsymbol{K_v}$, if each dof of the task is assigned a different gain.

3) Friction constraint: A set of lumped external forces is normally used, each one located at one vertex of the contacting surface (4 per foot). To ensure that the reference for these external forces be inside of their corresponding friction cone it is only necessary that $\mathbf{0} \leq \boldsymbol{\rho}$.

4) Surface constraint: To constrain a surface relative to the environment we can track it down to a desired position and orientation, in a similar way as the corresponding tasks were defined. See [16] for more details.

E. Torque control

The dynamically feasible reference configuration acceleration ($\dot{\boldsymbol{\alpha}}_{\boldsymbol{r}}$) produced by the QP at each time step is used to feed a low-level control, that generates the joint torques required to follow the given reference. Here, we used the torque control scheme proposed in [16].

F. Is a yaw moment compensation task required?

This question arises after realizing that the *friction constraint* described earlier already prevents the yaw moment from exceeding the slipping threshold defined by the friction cones at the contact points. However, this constraint cannot be considered as a satisfying solution with regard to slippage risk. Indeed, due to the nature of the QP optimization scheme, when this constraint is not active it has no effect on the resulting accelerations and contact forces; this constraint is therefore blind to risky motions as long as they do not hit the constraint. Then, when the constraints activate, they act as equality constraints which suddenly induce a loss in the dimension of the search space, and force the solution to slide on the constraint. This may generate by itself highly dynamic and dangerous motions, and sometimes leads to an empty feasible region, making the QP fail, which is a critical fault that can generally not be recovered from. Another issue with the constraint is that it relies on a model of the friction which may be wrong, and taking conservative solutions would only aggravate the problems mentioned above.

The yaw moment compensation that we present here is a control aiming at continuously compensating the excessive yaw moment while interfering the least possible with other tasks of the robot, thus avoiding the corresponding friction constraints which would be active only as a last resort.

III. Yaw Moment Compensation

A. Main idea

The humanoid robot performing an arbitrary motion has some linear and angular momenta, $\boldsymbol{l} \in \mathbb{R}^3$ and $\boldsymbol{k_0} \in \mathbb{R}^3$ respectively, with respect to the origin, which are related to the forces and moments (wrenches) exerted on the environment through their derivative with respect to time, $\dot{\boldsymbol{l}}$ and $\dot{\boldsymbol{k}}_{\boldsymbol{0}}$.

Let us consider that the only contacts with the environment are exerted through the feet (as in bipedal locomotion), and denote by $\boldsymbol{\tau_p}$ the vertical net moment of the reaction force $\boldsymbol{f}$ acting on the ZMP, whose position is denoted by $\boldsymbol{p}_{\text{zmp}}$. The moment of $\boldsymbol{f}$ about the origin of the world frame, $\boldsymbol{\tau_0}$, is calculated as

$$\boldsymbol{\tau_0} = \boldsymbol{p}_{\text{zmp}} \times \boldsymbol{f} + \boldsymbol{\tau_p}, \tag{10}$$

such that

$$\dot{\boldsymbol{l}} = \tilde{m}\boldsymbol{g} + \boldsymbol{f}, \tag{11}$$

$$\dot{\boldsymbol{k}}_{\boldsymbol{0}} = \boldsymbol{p}_{\text{com}} \times \tilde{m}\boldsymbol{g} + \boldsymbol{\tau_0}, \tag{12}$$

where $\boldsymbol{g} = \begin{bmatrix} 0 & 0 & -g \end{bmatrix}^T$ is the gravity vector and g is the acceleration due to gravity, whereas $\tilde{m}$ stands for the total mass of the robot and $\boldsymbol{p}_{\text{com}}$, for the position of the center of mass (CoM) of the robot in world coordinates. By substituting (10) and (11) into (12) and solving with respect to $\boldsymbol{\tau_p}$, we get [20]

$$\boldsymbol{\tau_p} = \dot{\boldsymbol{k}}_{\boldsymbol{0}} - \boldsymbol{p}_{\text{com}} \times \tilde{m}\boldsymbol{g} - \boldsymbol{p}_{\text{zmp}} \times \left(\dot{\boldsymbol{l}} - \tilde{m}\boldsymbol{g}\right), \tag{13}$$

whose vertical component, the *yaw moment*, is given by

$$\tau_{p,z} = \dot{k}_{0,z} - p_{\text{zmp},x}\dot{l}_y + p_{\text{zmp},y}\dot{l}_x. \tag{14}$$

As it can be seen, there is a close relationship between the yaw moment and the time derivative of both components of the momenta about the origin.

On the other hand, let us consider the angular momentum with respect to another point: the ZMP. To do that, we use the following relation:

$$\boldsymbol{k_0} = \boldsymbol{p}_{\text{zmp}} \times \boldsymbol{l} + \boldsymbol{k_p}, \tag{15}$$

which when differentiated and substituted into (13), leads to the following alternative expression for $\boldsymbol{\tau_p}$:

$$\boldsymbol{\tau_p} = \dot{\boldsymbol{k}}_{\boldsymbol{p}} + \dot{\boldsymbol{p}}_{\text{zmp}} \times \boldsymbol{l} + \boldsymbol{r_{p/c}} \times \tilde{m}\boldsymbol{g}, \tag{16}$$

where $\boldsymbol{r_{p/c}} = \boldsymbol{p}_{\text{zmp}} - \boldsymbol{p}_{\text{com}}$, and does not depend on $\dot{\boldsymbol{l}}$, but $\boldsymbol{l}$.

However, knowing that the x and y components of $\boldsymbol{\tau_p}$ are zero (by definition), the last expression is equivalent to

$$\boldsymbol{\tau_p} = \dot{\boldsymbol{k}}_{\boldsymbol{p}} + \dot{\boldsymbol{p}}_{\text{zmp}} \times \boldsymbol{l}. \tag{17}$$

Following the idea of [3], let us establish a target admissible value for the yaw moment, $\boldsymbol{\tau}^*_{\boldsymbol{p}}$, low enough (and even conservative) such that it can be easily compensated by the static frictional moment between feet and ground. Achieving $\boldsymbol{\tau}^*_{\boldsymbol{p}}$ requires to realize the corresponding target rate of change of the angular momentum $\dot{\boldsymbol{k}}^*_{\boldsymbol{p}}$ and the corresponding target linear momentum $\boldsymbol{l}^*$, according to

$$\boldsymbol{\tau}^*_{\boldsymbol{p}} = \dot{\boldsymbol{k}}^*_{\boldsymbol{p}} + \dot{\boldsymbol{p}}_{\text{zmp}} \times \boldsymbol{l}^*. \tag{18}$$

Subtracting (17) from (18) and defining $\boldsymbol{\Delta\tau_p} = \boldsymbol{\tau}^*_{\boldsymbol{p}} - \boldsymbol{\tau_p}$, $\boldsymbol{\Delta}\dot{\boldsymbol{k}}_{\boldsymbol{p}} = \dot{\boldsymbol{k}}^*_{\boldsymbol{p}} - \dot{\boldsymbol{k}}_{\boldsymbol{p}}$ and $\boldsymbol{\Delta l} = \boldsymbol{l}^* - \boldsymbol{l}$, we get

$$\boldsymbol{\Delta\tau_p} = \boldsymbol{\Delta}\dot{\boldsymbol{k}}_{\boldsymbol{p}} + \dot{\boldsymbol{p}}_{\text{zmp}} \times \boldsymbol{\Delta l}. \tag{19}$$

Given that it is not desirable to modify the trajectory of the CoM by means of the yaw moment compensation, it is reasonable to assume $\boldsymbol{\Delta l} \approx \mathbf{0}$, such that

$$\Delta\tau_{p,z} \approx \Delta\dot{k}_{p,z}; \tag{20}$$

that is, *adding* angular momentum with respect to the ZMP can directly compensate the yaw moment.

B. Momentum Equations

In order to generate angular momentum using the framework described in Section II, it is necessary to understand the relationship between the momenta rate of change and the configuration velocity and acceleration of the robot, $\boldsymbol{\alpha}$ and $\dot{\boldsymbol{\alpha}}$ respectively. This represents an important difference with respect to the velocity-based control framework used in [3].

As for the centroidal momenta rate of change (with respect to the CoM), widely used in whole-body control frameworks [17] [21], the above-mentioned relationship is written as

$$\dot{\boldsymbol{h}}_{\boldsymbol{c}} = \begin{bmatrix} \dot{\boldsymbol{l}} \\ \dot{\boldsymbol{k}}_{\boldsymbol{c}} \end{bmatrix} = \boldsymbol{A}_{\boldsymbol{c}}(\boldsymbol{q})\dot{\boldsymbol{\alpha}} + \dot{\boldsymbol{A}}_{\boldsymbol{c}}(\boldsymbol{q},\boldsymbol{\alpha})\boldsymbol{\alpha}, \tag{21}$$

where $\boldsymbol{A}_{\boldsymbol{c}}(\boldsymbol{q}) \in \mathbb{R}^{6\times(n+6)}$ is the centroidal momentum matrix [22] [23] and $\dot{\boldsymbol{A}}_{\boldsymbol{c}}(\boldsymbol{q},\boldsymbol{\alpha}) \in \mathbb{R}^{6\times(n+6)}$, its time derivative.

On the other hand, and in accordance with the finding expressed at the end of Section III-A, let us calculate the momenta rate of change of the robot with respect to the ZMP instead, by finding an expression equivalent to (21):

$$\dot{\boldsymbol{h}}_{\boldsymbol{p}} = \begin{bmatrix} \dot{\boldsymbol{l}} \\ \dot{\boldsymbol{k}}_{\boldsymbol{p}} \end{bmatrix} = \boldsymbol{A}_{\boldsymbol{p}}(\boldsymbol{q})\dot{\boldsymbol{\alpha}} + \dot{\boldsymbol{A}}_{\boldsymbol{p}}(\boldsymbol{q},\boldsymbol{\alpha})\boldsymbol{\alpha}, \tag{22}$$

where $\boldsymbol{A}_{\boldsymbol{p}}(\boldsymbol{q}) \in \mathbb{R}^{6\times(n+6)}$ is the ZMP momentum matrix and $\dot{\boldsymbol{A}}_{\boldsymbol{p}}(\boldsymbol{q},\boldsymbol{\alpha}) \in \mathbb{R}^{6\times(n+6)}$, its time derivative.

While it is possible to construct $\boldsymbol{A}_{\boldsymbol{p}}$ in a similar way as in [3] and estimate its derivative $\dot{\boldsymbol{A}}_{\boldsymbol{p}}$ by using finite differences, it is better to compute both analytically by realizing that both matrices can be calculated directly from the mass matrix $\boldsymbol{M}$ and the Coriolis factorization matrix $\boldsymbol{C}$, already calculated for the underactuation constraint of the motion solver.

The structure of $\boldsymbol{M}$ can be partitioned as

$$\boldsymbol{M} = \begin{bmatrix} \boldsymbol{M}_{\boldsymbol{v}}^{\boldsymbol{T}} & \boldsymbol{M}_{\boldsymbol{\omega}}^{\boldsymbol{T}} & \boldsymbol{M}_{\boldsymbol{j}}^{\boldsymbol{T}} \end{bmatrix}^{T} \tag{23}$$

where $\boldsymbol{M}_{\boldsymbol{v}} \in \mathbb{R}^{3\times(n+6)}$, $\boldsymbol{M}_{\boldsymbol{\omega}} \in \mathbb{R}^{3\times(n+6)}$ and $\boldsymbol{M}_{\boldsymbol{\theta}} \in \mathbb{R}^{n\times(n+6)}$. The first six lines of $\boldsymbol{M}$ actually correspond to the linear and angular components of the momentum matrix with respect to the position of the floating base, $\boldsymbol{A}_{\boldsymbol{B},\boldsymbol{v}}$ and $\boldsymbol{A}_{\boldsymbol{B},\boldsymbol{\omega}}$, respectively; that is,

$$\boldsymbol{A}_{\boldsymbol{B}} = \begin{bmatrix} \boldsymbol{A}_{\boldsymbol{B},\boldsymbol{v}} \\ \boldsymbol{A}_{\boldsymbol{B},\boldsymbol{\omega}} \end{bmatrix} = \begin{bmatrix} \boldsymbol{M}_{\boldsymbol{v}}, \\ \boldsymbol{M}_{\boldsymbol{\omega}} \end{bmatrix}. \tag{24}$$

In order to obtain the momentum matrix with respect to any other point (like the ZMP) it is necessary to obtain first the centroidal momentum matrix [24], as

$$\boldsymbol{A}_{\boldsymbol{c}} = \begin{bmatrix} \boldsymbol{A}_{\boldsymbol{c},\boldsymbol{v}} \\ \boldsymbol{A}_{\boldsymbol{c},\boldsymbol{\omega}} \end{bmatrix} = \begin{bmatrix} \boldsymbol{M}_{\boldsymbol{v}}, \\ \boldsymbol{M}_{\boldsymbol{\omega}} - \left(\hat{\boldsymbol{r}}_{\boldsymbol{c}/\boldsymbol{B}}\right)\boldsymbol{M}_{\boldsymbol{v}} \end{bmatrix}, \tag{25}$$

where $(\hat{\cdot}) : \mathbb{R}^3 \to \mathbb{R}^{3\times3}$ maps a vector into a skew-symmetric matrix, $\boldsymbol{r}_{\boldsymbol{c}/\boldsymbol{B}} = \boldsymbol{p}_{\text{com}} - \boldsymbol{p}_{\boldsymbol{B}}$ and $\boldsymbol{p}_{\text{com}}$ is the CoM position.

Then, with respect to the ZMP, we have

$$\begin{aligned}\boldsymbol{A}_{\boldsymbol{p}} = \begin{bmatrix} \boldsymbol{A}_{\boldsymbol{p},\boldsymbol{v}} \\ \boldsymbol{A}_{\boldsymbol{p},\boldsymbol{\omega}} \end{bmatrix} &= \begin{bmatrix} \boldsymbol{A}_{\boldsymbol{c},\boldsymbol{v}}, \\ \boldsymbol{A}_{\boldsymbol{c},\boldsymbol{\omega}} + \hat{\boldsymbol{r}}_{\boldsymbol{c}/\boldsymbol{p}}\boldsymbol{A}_{\boldsymbol{c},\boldsymbol{v}} \end{bmatrix} \\ &= \begin{bmatrix} \boldsymbol{M}_{\boldsymbol{v}}, \\ \boldsymbol{M}_{\boldsymbol{\omega}} + \left(\hat{\boldsymbol{r}}_{\boldsymbol{c}/\boldsymbol{p}} - \hat{\boldsymbol{r}}_{\boldsymbol{c}/\boldsymbol{B}}\right)\boldsymbol{M}_{\boldsymbol{v}} \end{bmatrix},\end{aligned} \tag{26}$$

where $\boldsymbol{r}_{\boldsymbol{c}/\boldsymbol{p}} = \boldsymbol{p}_{\text{com}} - \boldsymbol{p}_{\text{zmp}}$.

Once this relation is obtained, $\dot{\boldsymbol{A}}_{\boldsymbol{p}}$ is calculated as

$$\dot{\boldsymbol{A}}_{\boldsymbol{p}} = \begin{bmatrix} \dot{\boldsymbol{M}}_{\boldsymbol{v}} \\ \dot{\boldsymbol{M}}_{\boldsymbol{\omega}} + \left(\hat{\boldsymbol{r}}_{\boldsymbol{c}/\boldsymbol{p}} - \hat{\boldsymbol{r}}_{\boldsymbol{c}/\boldsymbol{B}}\right)\dot{\boldsymbol{M}}_{\boldsymbol{v}} + \left(\dot{\hat{\boldsymbol{r}}}_{\boldsymbol{c}/\boldsymbol{p}} - \dot{\hat{\boldsymbol{r}}}_{\boldsymbol{c}/\boldsymbol{B}}\right)\boldsymbol{M}_{\boldsymbol{v}} \end{bmatrix}, \tag{27}$$

where $\dot{\boldsymbol{M}}_{\boldsymbol{v}}$ and $\dot{\boldsymbol{M}}_{\boldsymbol{\omega}}$ correspond to the first six lines of $\dot{\boldsymbol{M}}$, which can be calculated analytically as

$$\dot{\boldsymbol{M}} = \boldsymbol{C} + \boldsymbol{C}^{\boldsymbol{T}}, \tag{28}$$

and that can be verified to be a direct consequence of the skew-symmetric property of the Coriolis and centripetal matrix factorization.

C. Task Implementation

Let us implement the yaw moment compensation task in such a way that it generates additional angular momentum according to the current exceeding yaw moment; that is, by setting

$$\boldsymbol{J}_{\boldsymbol{g}} = \boldsymbol{A}_{\boldsymbol{p},\boldsymbol{\omega},\boldsymbol{z}}, \qquad \dot{\boldsymbol{J}}_{\boldsymbol{g}} = \dot{\boldsymbol{A}}_{\boldsymbol{p},\boldsymbol{\omega},\boldsymbol{z}}, \qquad \ddot{g}_{\text{ob}} = \Delta\dot{k}_{p,z,\text{ob}}, \tag{29}$$

in reference to (4), where the subscript z stands for the third row (or element) of the corresponding matrices (or vector).

A straightforward application of (20) would imply to simply set $\Delta\dot{k}_{p,z,\text{ob}} = \Delta\tau_{p,z}$; however, there are some technical issues introducing modeling errors that must be considered:

1) The nature of the ZMP: The ZMP can be computed from the force/moment sensors placed at the feet of the robot, but the signal is noisy. Filtering it can diminish the problem within a velocity-based control approach, as only $\boldsymbol{A}_{\boldsymbol{p},\boldsymbol{\omega},\boldsymbol{z}}$ is required, as done in [3]. However, within the current approach $\dot{\boldsymbol{A}}_{\boldsymbol{p},\boldsymbol{\omega},\boldsymbol{z}}$ is also required, and this one depends on the velocity of the ZMP, estimated by finite differences.

A better solution is to use the *desired* trajectory of the ZMP instead of the measured signal, but this introduces modeling error.

2) The nature of the yaw moment: In the same way as the ZMP, the yaw moment can also be computed from the force/moment sensors, such that it is a direct sum of the z-th component of the moment at each foot; that is, $\tau_{p,z} = \tau_{r,p,z} + \tau_{l,p,z}$. However, this signal is also noisy.

A cleaner signal can be obtained by using the yaw moment computed from the optimized reference of external forces given by the QP ($\boldsymbol{u}_{\boldsymbol{e},\boldsymbol{r}}$), but there is a delay of one time step, leading to a late control action that is unable of compensating the yaw moment. One solution to this issue is to consider an additional term proportional to the time derivative of the yaw moment compensation, $\Delta\dot{\tau}_{p,z}$, which can be computed by finite differences. This requires an even cleaner signal of the yaw moment, obtained by using a simple low-pass filter.

3) Partial yaw moment compensation: In [3], the target value for the yaw moment ($\tau_{p,z}^{*}$) was obtained by saturating the actual yaw moment with a threshold value, $\tau_{p,z}^{th}$; that is,

$$\tau_{p,z}^{*} = \begin{cases} \tau_{p,z}^{th} & \text{if} \quad \tau_{p,z}^{r} > \tau_{p,z}^{th} \\ \tau_{p,z} & \text{if} \quad -\tau_{p,z}^{th} \le \tau_{p,z}^{r} \le \tau_{p,z}^{th} \\ -\tau_{p,z}^{th} & \text{if} \quad \tau_{p,z}^{r} < -\tau_{p,z}^{th} \end{cases}, \tag{30}$$

but this solution has some disadvantages:

- It produces a discontinuous $\Delta\dot{\tau}_{p,z}$, leading to a discontinuity in the task and in the reference signals outputted by the QP, including the reference of external forces.
- It produces a compensation signal only after the yaw moment has crossed the threshold.

A solution that overcomes both of these issues is to use a sigmoid function instead:

$$\tau^*_{p,z} = \tau^{th}_{p,z} \cdot \tanh\left(\frac{\tau_{p,z}}{\tau^{th}_{p,z}}\right). \tag{31}$$

As it is a smooth approximation of the saturation function, $\Delta\tau_{p,z} = \tau^*_{p,z} - \tau_{p,z}$ will produce values increasing in magnitude as $\tau_{p,z}$ approaches to $\tau^*_{p,z}$.

4) Modification of the dynamics of the generated angular momentum: Due to the nature of the velocity-based control proposed in [3], and in the absence of competing tasks and constraints, a compensation action produced a rapid increment (or decrement) of the added angular momentum, which remained constant after the compensation. A constant angular momentum is produced by joints moving with a constant velocity, leading to drift. The solution found in that paper was to modify the dynamics of the generated angular momentum, to slowly take it to zero by means of two integral actions, such that the yaw moment produced by that modification was below of the threshold value. In this way, the motion of the joints would also stop and the net joint displacement would be zero.

Within the current framework the task does not compensate the yaw moment when

$$\Delta\dot{k}_{p,z,\mathrm{ob}} - \dot{\boldsymbol{A}}_{\boldsymbol{p},\boldsymbol{\omega},\boldsymbol{z}}\boldsymbol{\alpha} = 0, \tag{32}$$

as seen from (4); that is, if the generated angular momentum is constant (even if it is not zero) and if $\boldsymbol{\alpha} \in \ker(\dot{\boldsymbol{A}}_{\boldsymbol{p},\boldsymbol{\omega},\boldsymbol{z}})$. This means that there could be a residual motion due to the latter term even if it is not drifting, unless another task in the QP provides some damping, aimed to take $\boldsymbol{\alpha}$ to zero.

Based on the above discussion, it must be clear that $\Delta\dot{k}_{p,z}$ should be defined as a control signal, proposed here as

$$\Delta\dot{k}_{p,z,\mathrm{ob}} = \kappa_p\Delta\tau_{p,z} + \kappa_v\Delta\dot{\tau}_{p,z} - \kappa_i\Delta k_{p,z,\mathrm{ob}} - \kappa_{ii}\int\Delta k_{p,z,\mathrm{ob}}, \tag{33}$$

where we have included the modification of the dynamics of the generated angular momentum by means of the two integral actions (tuned with κ_i, κ_{ii}) to verify our claims.

A simplified diagram summarizing the above discussion and focusing only on the yaw moment compensation task is shown in Fig. 1. For a diagram focusing on the motion solver and the low level control see [16].

IV. Simulation Results

A. Simulation Environment

To test the yaw moment compensation scheme, let us consider a simple humanoid robot with a two legs, of 6 DoF each, and two arms, with 7 DoF each, as shown in Fig. 2a.

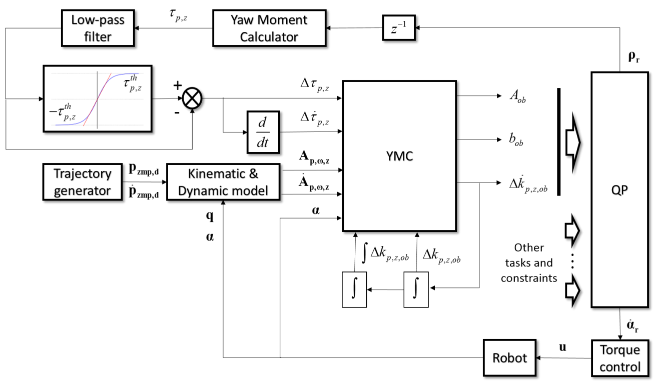

Fig. 1: Simplified diagram focusing on the yaw moment compensation task.

The height of this fictitious humanoid robot is 1.8 m when the legs are fully extended, and its weight is 77 kg. The mass for this robot is distributed in a realistic way on every link, even if they are modeled with approximate zero length (modeled as point masses). The tensors of inertia are calculated from the modeled shape of the link: torso, foots and hands are modeled as boxes, while other links are modeled as tubes with radius of 0.001 m, such that under these conditions the mass matrix is always non-degenerate.

This humanoid robot was modeled in Matlab Simscape Multibody$^{\text{TM}}$. The links of the robot were defined according to the above stated guidelines, whereas the joints were implemented as driven by geared servomotors considering an additional rotor inertia. The input to these servos is the joint torque, scaled by the corresponding gear ratio. Electromechanical effects were not considered.

The contact model between the foot and the ground was implemented by using the free Simscape Multibody Contact Forces Library [25]. This one considers that each vertex of the foot is attached to a tiny fictitious sphere, such that the wrench resulting from the contact of each sphere with the ground is calculated by using a penalty method for the normal component, and a Stick-Slip continuous friction model for the tangential component. This allows to simulate collision and sliding motion.

The controller is implemented in discrete time with a step size of $T = 0.005$ s by using common blocks of Simulink$^{\text{TM}}$ and the QP Solver (`quadprog`) provided by Matlab's Optimization Toolbox$^{\text{TM}}$. The feedback received by the controller is a set of signals given as an output by the Simscape model and reflecting the real ones for $\boldsymbol{q}$; that is, the controller is also fully aware of the position and orientation of the body, which in reality can only be estimated but not measured. As for $\dot{\boldsymbol{q}}_{\boldsymbol{\theta}}$, it was estimated using finite differences.

To run the simulations, Matlab$^{\text{TM}}$ was allowed to automatically choose a variable step solver, as well as most of the parameters (maximum and minimum step size and absolute tolerance). The only parameter specified was the relative tolerance, for which a value of 1E-3 was given.

TABLE I: Contact parameters between feet and ground.

Parameter	Value
Contact Stiffness	1E7 N/m
Contact Damping	1E3 N/(m/s)
Static & Kinetic Friction Coefficients	0.4 & 0.3
Velocity Threshold	0.001 m/s

B. Motion Configuration

In order to show the performance of the yaw moment compensation scheme let us simulate a kicking motion, performed in a typical slippery floor, simulated by setting the contact parameters between feet and floor as shown in Table I.

The robot is commanded first to go from an initial configuration (all joint angles at 0 deg) to a half-sitting pose (by bending its knees and moving its hands to a predetermined pose) in 0.5 s. Then, the robot performs a kicking motion by shifting the CoM over its left foot in 1 s, taking the right foot 0.35 m behind its waist and rising it 0.15 m from the floor in 0.5 s. The kicking motion starts at $t = 2$ s, by moving the foot forward to 0.35 m in front of the waist. This motion is commanded by a desired trajectory following a straight horizontal path, configured to arrive to the final position within 0.25 s (a very fast motion). At $t = 3$ s, the right foot is commanded to go back to the floor at its initial position and then, the robot is taken back to half-sitting pose. All these desired trajectories were constructed using a Linear Segments with Parabolic Blends (LSPB) profile, and used to feed the desired values of the tasks in the QP.

This motion is accomplished by using the following tasks: (a) a low-weight constant posture task (`q`) set to half-sitting with the purpose of dealing with the initial singularity and bend the knees correctly, as well as adding regulation and damping, (b) a horizontal position task for the CoM (`com`) working together with (c) a vertical position and full orientation task for the floating Base (`poseB`), (d) a pose task for the Right and Left Hands (`poseRH`, `poseLH`) used to maintain them in a given configuration while there is no yaw moment to compensate, as well as to provide the necessary damping to bring them to rest, (e) a pose task for the Right Foot (`poseRF`), active only during single support phase and used for performing the kicking motion, and (f) the yaw moment compensation (`YMC`) task described in this paper. The gains and weights used for the tasks are shown in Table II, where $W = \text{diag}(\texttt{W\_mask}) \times \texttt{W\_val}$. When a value of 0 is used within the mask, the corresponding dof is not controlled. The gains for `YMC` are not shown as they will vary on each experiment.

In order to successfully execute this motion, it is necessary to consider the following constraints too: (a) the underactuation constraint, to account for a feasible motion, (b) the torque constraint, (c) the joint range limits constraint, (d) the friction constraint, and (e) surface constraints for the Right and Left Feet Soles (`RFSole`, `LFSole`) used to hold the contacts with the ground, being the former one active only while the robot is in double support phase, and the latter active through the entire motion.

TABLE II: Default task parameter values.

Task	Parameter	Value	Task	Parameter	Value
q	active	entire motion	poseRH & poseLH	active	entire motion
	W_val	10		W_val	100
	W_mask	ones(1, n)		W_mask	[1 1 1 1 1 1]
	k_p	100		k_p	100
	k_v	20		k_v	300
com	active	entire motion	poseRF	active	single support
	W_val	300		W_val	500
	W_mask	[1 1 0]		W_mask	[1 1 1 1 1 1]
	k_p	500		k_p	100
	k_v	100		k_v	20
poseB	active	entire motion	YMC	active	entire motion
	k_p	100		W_val	500
	k_v	20		W_mask	1
	W_val	200			
	W_mask	[0 0 5 2 2 2]			

TABLE III: Default constraint parameter values.

Constraint	Parameter	Value
RFSole & LFSole	Kp	diag([0 0 0 1 1 0]) * 100
	Kv	diag([1 1 1 1 1 1]) * 200

The joint velocity limits constraint was not considered on purpose in order to achieve a faster kicking motion. As for the friction constraint, the controller assumes a coefficient of friction ($\mu = 0.6$) larger than the ones used to configure the contact in Matlab Simscape Multibody$^{\text{TM}}$ (see Table I). This is done with the purpose of preventing the friction constraint to be activated, allowing for yaw rotations to easily happen, assessing the validity of our method. The default PD gain matrices for the constraints used by both simulations are shown in Table III.

C. Experiments and Results

Let us compare the above described kicking motion in three conditions: (a) without yaw moment compensation (`noComp`), (b) with yaw moment compensation but without modifying the dynamics of the generated yaw angular momentum (`simpleComp`), and (c) with yaw moment compensation while modifying the dynamics of the generated yaw angular momentum (`fullComp`). In every case, a first-order low-pass digital filter with parameter a was used to get a clean $\tau_{p,z}$ signal. This parameter represents the level of attenuation at the digital frequency of $\frac{1}{2}\pi$. All the parameters of the `YMC` task, for every condition in which it is active, are listed in Table IV.

Some snapshots of the resulting motion when considering the `fullComp` case are shown in Fig. 2. Other cases are not displayed because the differences are subtle.

The resulting yaw moment, computed by the QP and exerted due to the motion of the robot, is shown in Fig. 3, whereas the generated yaw angular momentum is shown in Fig. 4. Notice that without compensation, the yaw moment achieves a peak value of 60 N m. On the other hand, the `YMC` task successfully constrained this yaw moment between -1 and 1 N m (in both cases), as dictated by the threshold. Notice that the `fullComp` achieves a slightly bigger yaw moment than `simpComp`. This is a direct consequence of bringing back the generated yaw angular momentum to zero, as it was the purpose of the modification of its dynamics.

TABLE IV: Yaw moment compensation task parameters.

Parameter	simpleComp	fullComp
a (LP Filter)	0.3	0.3
$\tau_{p,z}^{th}$	1 N m	1 N m
k_p	25	25
k_v	0.05	0.05
k_i	0	20
k_{ii}	0	50

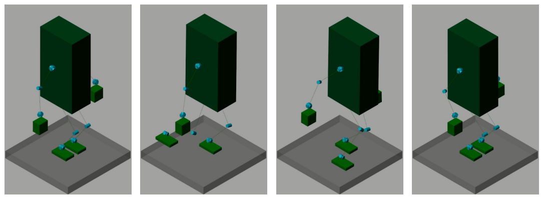

Fig. 2: Kicking motion (using `fullComp`).

The yaw angle of the support foot is shown in Fig. 5 together with its excursion, calculated as the integral of the absolute value of its derivative. Notice that while the total excursion of the yaw angle without compensation is 13 deg, `simpComp` and `fullComp` achieved values of 2 deg.

The plot for the x component of the swing foot is shown in Fig. 6. There, it can be seen that the foot could not follow the fast desired trajectory in any case, and that there is a point where there is an abrupt slow down, caused by the solution of the QP sliding on the friction constraint. Actually, at this moment the heels slightly separate from the floor, and this is because the method investigated in this paper prevents a yaw slippery rotation but not a translational slippery.

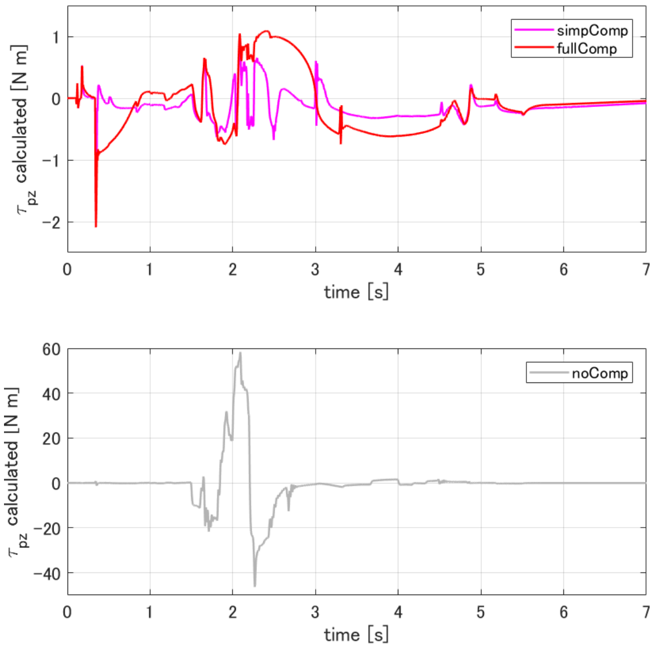

Fig. 3: Resulting yaw moment ($\tau_{p,z}$) (given by the QP).

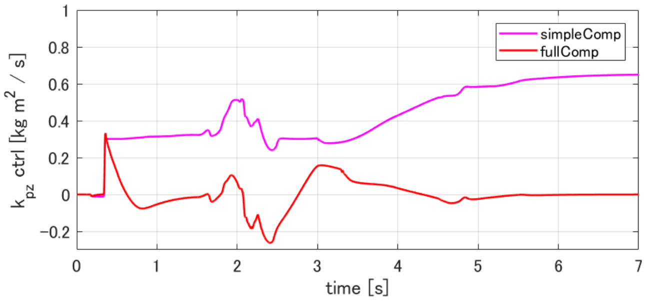

Fig. 4: Generated angular momentum ($\Delta k_{p,z,\mathrm{ob}}$) (control signal).

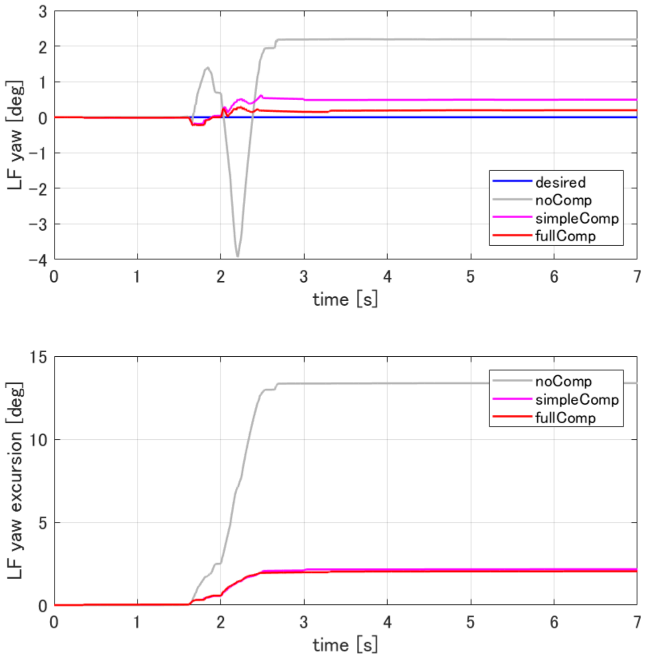

Fig. 5: Left foot yaw (ψ_{LF}) and its excursion ($\int \left|\dot{\psi}_{LF}\right| dt$) (measured).

The plot for the x component of the right hand is shown in Fig. 7. There, it can be seen that the hand also moves when the yaw moment is not compensated, but in a different way as a consequence of the CoM task. Notice that both, `simpComp` and `fullComp`, achieved no residual final motion thanks to the damping of the pose task for the hands. However, the evolution is different. In the case of `fullComp`, the residual final motion is eliminated mainly by the yaw moment compensation task, whereas in the case of `simpComp` it is done by all the other tasks and constraints.

Finally, the plot for the yaw angle of the body is shown in Fig. 8, where we can also see the effect of the compensation. Using the approach explained in [3], this compensating motion could not have been generated, as there is no yaw-axis waist joint. This implies that the trajectories of the leg joints, which cannot be "free" using that approach, were also modified.

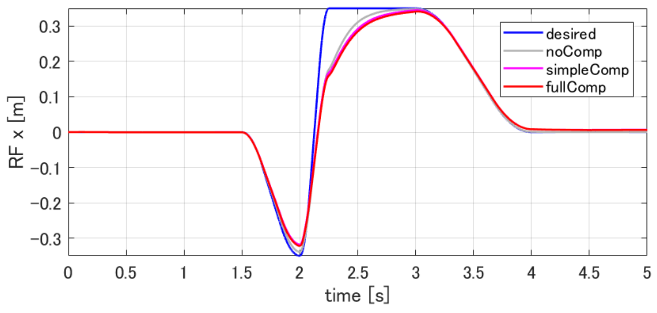

Fig. 6: Right foot x-position ($p_{RF,x}$) (measured).

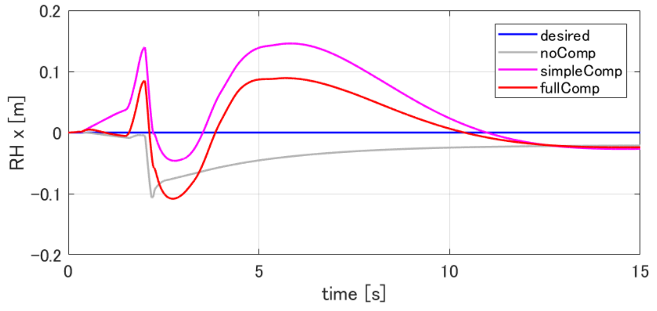

Fig. 7: Right hand x-position ($p_{RH,x}$) (measured).

V. Conclusions

This paper presents an on-line yaw moment compensation framework that uses a complete whole-body control to compensate for the current exceeding yaw moment, such that every single joint can contribute to the compensation, not only a selected dedicated set. At the same time, the controller achieved a yaw moment that was successfully bounded by the desired threshold values. This was achieved by using in the control signal not only the yaw moment exerted at the previous time step, but also its derivative, as well as by considering additional tasks and constraints.

As a future work we want to extend this framework to counteract possible falls, considering multiple contacts.

Acknowledgments

This work was partially supported by NEDO's "Development of a highly dependable humanoid robot system that can work in unstructured environments" project, and in part by the COMANOID EU project and the CNRS-AIST-Airbus Joint Research Program (JRP).

References

[1] R. Cisneros et al. Effective teleoperated manipulation for humanoid robots in partially unknown real environments: team AIST-NEDOs approach for performing the Plug Task during the DRC Finals. *Advanced Robotics*, 30(24), 2016.

[2] B. Ugurlu et al. Yaw moment compensation for bipedal robots via intrinsic angular momentum constraint. *Int. Journal of Humanoid Robotics*, 9(4), 2012.

[3] R. Cisneros et al. Partial Yaw Moment Compensation Through Whole-Body Motion. In *IEEE-RAS Int. Conf. on Humanoid Robots*, 2014.

[4] J. Park. Synthesis of natural arm swing motion in human bipedal walking. *Journal of Biomechanics*, 41(7), 2008.

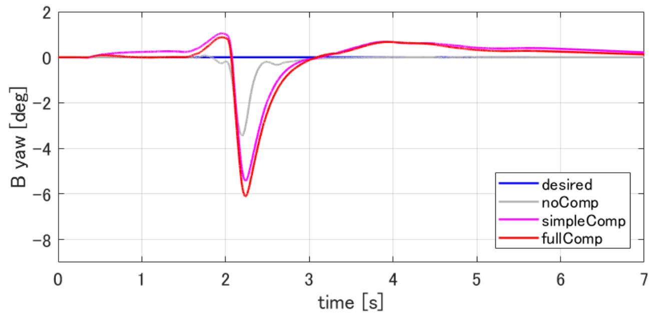

Fig. 8: Body yaw (measured).

[5] H. Pontzer et al. Control and function of arm swing in human walking and running. *The Journal of Experimental Biology*, 212, 2009.

[6] D.P. Xing et al. Arm / trunk motion generation for humanoid robot. *Science China: Information Services*, 53(8), 2010.

[7] G. Fu et al. A Yaw Moment Counteracting Method for Humanoid Robot Based on Arms Swinging. *Robot*, 34(4), 2013.

[8] S. Zhang et al. The mechanism of yaw torque compensation in the human and motion design for humanoid robots. *INTECH Int. Journal of Advanced Robotic Systems*, 10(57), 2013.

[9] L. Yang et al. Yaw Moment Compensation for Humanoid Robot via Arms Swinging. *Open Automation and Control Systems Journal*, 6(1), 2014.

[10] L. Yang et al. Energy-Efficient Yaw Moment Control for Humanoid Robot Utilizing Arms Swing. *Int. Journal of Precision Engineering and Manufacturing*, 17(9), 2016.

[11] S. Nakaoka et al. Task model of lower body motion for a biped humanoid robot to imitate human dances. In *IEEE/RSJ Int. Conf. on Intelligent Robots and Systems (IROS)*, 2005.

[12] W. Yu et al. Fastwalking pattern generation for humanoid robot using waist joint moment compensation. *Robot*, 32(2), 2010.

[13] T. Otani et al. Angular Momentum Compensation in Yaw Direction using Upper Body based on Human Running. In *IEEE Int. Conf. on Robotics and Automation (ICRA)*, 2017.

[14] T. Hirabayashi et al. Yaw Moment Compensation of Biped Fast Walking Using 3D Inverted Pendulum. In *Int. Workshop on Advanced Motion Control (AMC)*, 2008.

[15] R. Cisneros et al. Yaw Moment Compensation by Using Full Body Motion. In *IEEE Int. Conf. on Mechatronics and Automation (ICMA)*, 2014.

[16] R. Cisneros et al. Robust humanoid control using a QP solver with integral gains. In *IEEE/RSJ Int. Conf. on Intelligent Robots and Systems (IROS)*, 2018.

[17] M.A. Hopkins et al. Compliant Locomotion Using Whole-Body Control and Divergent Component of Motion Tracking. In *IEEE Int. Conf. on Robotics and Automation (ICRA)*, 2015.

[18] K. Bouyarmane et al. Using a Multi-Objective Controller to Synthesize Simulated Humanoid Robot Motion with Changing Contact Configurations. In *IEEE/RSJ Int. Conf. on Intelligent Robots and Systems (IROS)*, 2011.

[19] J. Vaillant et al. Multi-contact vertical ladder climbing by an HRP-2 humanoid. *Autonomous Robots*, 40(3), March 2016.

[20] S. Kajita et al. *Introduction to Humanoid Robotics*. Springer, 2005.

[21] T. Koolen et al. Design of a Momentum-Based Control Framework and Application to the Humanoid Robot Atlas. *Int. Journal of Humanoid Robotics*, 13(1), 2016.

[22] S. Kajita et al. Resolved Momentum Control: Humanoid Motion Planning based on the Linear and Angular Momentum. In *IEEE/RSJ Int. Conf. on Intelligent Robots and Systems (IROS)*, 2003.

[23] D. Orin et al. Centroidal Momentum Matrix of a Humanoid Robot Structure and Properties. In *IEEE/RSJ Int. Conf. on Intelligent Robots and Systems (IROS)*, 2008.

[24] F.P. Beer et al. *Vector Mechanics for Engineers: Statics and Dynamics*. McGraw-Hill, 10th edition, 2013.

[25] S. Miller. Simscape Multibody Contact Force Library. `https://www.mathworks.com/matlabcentral/fileexchange/47417-simscape-multibody-contact-forces-library`, 2017.