Generation of Dynamic Motion for Anthropomorphic Systems under Prioritized Equality and Inequality Constraints

L. Saab, N. Mansard, F. Keith, J-Y. Fourquet, P. Soueres

Abstract—In this paper, we propose a solution to compute full-dynamic motions for a humanoid robot, accounting for various kinds of constraint such as dynamic balance or joint limits. In a first time, we propose an unification of task-based control schemes, in inverse kinematics or inverse dynamics. Based of this unification, we then generalize the cascade of quadratic programs that where developed for inverse kinematics only. We then apply the solution to generate in simulation whole-body motions for a humanoid robot in unilateral contact with the ground, while ensuring the dynamic balance on a not horizontal surface.

I. INTRODUCTION

This paper focuses on whole-body motion generation in humanoid robotics. We are interested in tasks involving simultaneous motion of upper and lower parts of the body. These tasks are given at the operational level in terms of motion of the bodies involved in the task: hands for reaching or manipulating, feet for locomotion. This gives rise to trajectory equality constraints. Thus kinematic redundancy and interaction between tasks are central issues. On the other hand, equilibrium and unilateral contact constraints lead to consider forces and inequality constraints. Finally, the problem writes as a set of equality and inequality constraints in terms of motion and force variables.

Different kind of problems involving various possibly antagonistic tasks have already been tackled in the robotics literature. The first attempts were concerned by multiple equality constraints at kinematic level in Resolved Motion Rate Control schemes [18]. Prioritization has been adopted following the pioneering work of Nakamura [24] with fixed [1] or adjustable priority [2]. In these works, inequality constraints arising from the boundedness of joint range, workspace or velocities were not exactly taken into account: several authors applied a potential field approach projected in the null-space of equality constraints [15], [23] whereas other contributions prefer to apply a clamping at the joint limits when solving the equality constraints [4], [29]. In fact, the kinematic formulation can be written as a quadratic problem where the goal is to find a norm-minimizer under linear equality and inequality constraints in configuration space. It was recently proposed [14] a specific way to consider such quadratic problems that enables to find an approximate solution based on successive prioritization even when inequality constraints forbids an exact solution.

Whole-body motion is naturally concerned with dynamics and contact forces. A dynamical formulation is necessary and tasks dynamics writes as linear equalities whose unknowns are the generalized torques. Several approaches consider prioritization techniques within a dynamic formulation. This is particularly the case for the works by Khatib and colleagues [32], [27], [33], [31], [17]. The survey paper [26] gives a good comparison between operational dynamics approaches. In this family of methods, prioritization is fixed and inequalities are treated by projected potential techniques. Recently, various techniques have been developed for taking into account exactly unilateral constraints due for instance to joint limits or multiple contacts: [20] proposed an original scheme to compute a generic control law from a hierarchic set of both unilateral constraints and bilateral tasks and [6] solves dynamic and static quadratic problems for multi-contact.

This paper exploits the quadratic nature of the dynamic formulation including unilateral contact constraints in order to take into account equality and inequality constraints in a way similar to the one proposed at kinematic level in [14]. With first recall the classical schemes for inverse kinematics and dynamics in Section II. A generic shape is drawn from these schemes, that can be used to build an unified resolution.

In Section III, we prove the equivalence of the unified methodology with the control scheme of the literature, in the case of the free-floating humanoid robot. We then generalize in Section IV the hierarchy of quadratic programs, first used for inverse kinematics only, to the dynamic case. A set of specific tasks is then explicit in Section V and applied to the humanoid robot model in simulation VI. The experiments consist in performing a whole body task while preserving the robot balance on a not-horizontal surface.

II. NUMERICAL INVERSE RESOLUTION

In this section, the inverse algorithm for both kinematics-based and dynamics based control are developed. The similarities in term of algorithmic resolution are then drawn, that can be used to generate common solvers dedicated to motion generation.
A. Task function approach

The task-function approach [30] (or operational space approach [16], [25]) consist in designed the motion to be performed as a control law in a subspace of small dimension, and then back project this control law from the subspace to the space of the whole robot using numerical inverse. A task is defined the triplet \((e, \dot{e}^*, G)\), where \(e\) is the task space (i.e. a submanifold of the robot configuration), \(\dot{e}^*\) is the reference behavior in the task space, and \(G\) is the differential link between the task space and the robot actuators:

\[
\dot{e} + \mu = Gu
\]

where \(\mu\) is the drift of the task. The reference \(\dot{e}^*\) is in the tangent space to \(e\), and the differential link \(G\) maps the elements of the tangent subspace of the robot configuration to the tangent space to \(e\).

The interest to define the robot motion inside a task space rather than directly at the joint level is double: first the task space is arbitrarily chosen as a small, easy space where the control law \(\dot{e}^*\) is strait forward to design (typically, in visual servoing the task space is chosen as the visual space, where the link between sensor feedback and control is direct [9], [12]); second, it is also easy to prevent the interference between two task spaces and then decouple two concurrent objectives to be apply simultaneously on a same robot, using projection operator.

The differential link \(G\) gives the direct link between the actuator \(u\) and the feedback \(e\): from a given robot motion, \(G\) gives the reaction in the task space. To compute a specific robot control \(u\) that performs the reference \(\dot{e}^*\), any numerical inverse of \(G\) can be used. The generic control law is then

\[
u = G^\#(\dot{e}^* + \mu)
\]

where \(\#\) is a matrix-inversion operator.

B. Inverse kinematics

In inverse kinematics, the control input \(u\) is simply the robot joint velocity \(u = \dot{q}\). The differential link between the task space and the control input is the task Jacobian \(G = J = \frac{de}{dq}\). In that case, the drift \(\mu\) is null. For kinematics inverse, the matrix inverse operator is most of the time the pseudo inverse [3], [10]. The control law is then:

\[
\dot{q}^* = J^T\dot{e}^* + P\dot{q}2
\]

where \(P\) is the projection operator into the null space of \(J^T\), that allows to consider any secondary control objectives \(\dot{q}_2\) without disturbing the realization of main objective \(\dot{e}^*\) [18]. A typical reference behavior if \(\dot{e}^* = -\lambda e\), that regulates to 0 with a velocity tuned by the parameter \(\lambda\).

C. Projected inverse kinematics

The projector \(P\) represents the redundancy of the robot with respect to the task. If a second task \((\dot{e}_2, \dot{e}_2^*, J_2)\) as to be performed, then \(\dot{q}_2\) can be used as the new control input. In that case, the template (2) is obtained by replacing (3) into \(\dot{e}_2 = J_2\dot{q}_2\):

\[
\dot{e}_2 = J_2\dot{q}_2
\]

In that case, the differential link is the projected Jacobian \(G = J_2P\), and the drift is \(\mu = -J_2\dot{q}^*\). The control input \(\dot{q}_2\) is obtained once more by numerical inversion [35], [1]:

\[
\dot{q}_2^* = (J_2P)^+(\dot{e}_2 - J_2\dot{q}^*) + P\dot{q}_3
\]

The same scheme can be reproduced to account for a third task using the projected control input \(\dot{q}_3\), and iteratively any number of task until \(P_3\) is null (no more redundancy).

D. Inverse dynamics

In dynamics, the input of the system is the robot motor torques \(u = \tau\). The state of the robot is the pair \((q, \dot{q})\), and the task space comprehend both the position and velocity. The reference \(\dot{e}^*\) is homogeneous to an acceleration, and then denoted as an acceleration \(\ddot{e}\). Contrarily to the kinematic case, the map to the control input can not be built directly, but is obtained from two stages: first the dynamic equation of the system, typically:

\[
A\ddot{q} + b = \tau
\]

where \(A\) is the generalized inertia matrix of the system, \(\dot{q}\) is the robot configuration third derivative, and \(b\) is the dynamical drift (typically, gravity torques and Coriolis accelerations).

The acceleration \(\ddot{q}\) is link to the task space by the task Jacobian:

\[
J\ddot{q} + J\ddot{q} = \ddot{e}
\]

Multiplying (6) by \(JA^{-1}\), the differential link between \(\tau\) and \(\ddot{e}\) is obtained:

\[
\ddot{e} + J\ddot{q} + JA^{-1}b = JA^{-1}\tau
\]

This last equation corresponds to the template (2) with \(G = JA^{-1}\) and \(\mu = J\ddot{q} + JA^{-1}b\).

The dynamic-inverse control law is then directly obtained by inverting \(G\). To fit with the dynamic of the system, it has been proposed [16] to use the weighted pseudo inverse, with \(A\) as weights of the right-hand side [7]:

\[
\tau^* = (JA^{-1})^#A(\ddot{e} + J\ddot{q} + JA^{-1}b) + Pr_2
\]

This inverse gives the least norm \(||\tau||_4 = \sqrt{\tau^TA^{-1}\tau}, which corresponds to a minimization of the acceleration pseudo-energy \(\dot{q}^TA\dot{q}\) [28].

E. Projected inverse dynamics

As in the kinematic case, the projector \(P\) represents the redundancy of the system with respect to the task \(e\). The secondary torques can be used as a control input to perform a second task \(e_2\). Similarly as before, the differential link is obtained by replacing \(\tau^*\) in the robot dynamic equation. The differential link is then directly:

\[
\ddot{e}_2 + \mu_2 = G_2\tau_2
\]

with \(\mu_2 = J_2\ddot{q} + J_2A^{-1}b - J_2A^{-1}\tau^*\) and \(G_2 = J_2A^{-1}P\).

The same weighted inverse is used to inverse \(G_2\) [32]. AS before, a second projector appears during the inversion, and a third then any number of tasks can be added iteratively until the projector becomes null.
F. Inverse dynamics with rigid contacts

When the robot is in contact with the environment, the dynamic equation of the system becomes:
\[ A\ddot{q} + b + J_c^T f_c = \tau \] (11)
where \( J_c \) is the Jacobian matrix of the contact points \( J_c = \frac{\partial x_c}{\partial q} \), \( x_c \) being the contact points, and \( f_c \) are the respective forces. The rigid contact implies that there is no motion of the robot contact points:
\[ \dot{x}_c = 0 \quad \ddot{x}_c = 0 \] (12)
From this constraint, it is possible to obtain a force-free dynamic equation (see Appendix VII for the constructive proof):
\[ A\ddot{q} + b_p = P_c \tau \] (13)
with \( P_c = (I - J_c^T J_c)^{-1} = I - (J_c A^{-1})^T J_c A^{-1} \) is the projection operator of the contact, and \( b_p := P_c b + J_c A^{-1} J_c^T J_c \dot{q} \).

As previously, the differential link between a task \( e \) and the torque input is obtained using the intermediate variable \( \dot{q}_c \):
\[ e^T J A^{-1} b_p = J A^{-1} P_c \tau \] (14)

In that case, an interesting task is to control the force \( f_c \) to a reference value \( f^*_c \). Using \( e = x_c \), and setting \( \dot{e}^* = \Lambda_c f^*_c \), with \( \Lambda_c = (J_c A^{-1} J_c^T)^{-1} \) the apparent mass matrix at the contact point, it can be shown that the resulting force is exactly \( f_c = f^*_c \).

G. Generic actuators

In the generic case, the system actuation is given by:
\[ A\ddot{q} + b + J_c^T f_c = J_a^T \tau \] (15)
where \( J_a \) gives the link between the motors and the motion. The case of the robot is a simplification of this more generic case, since the motors controls the joint position. Two cases are of particular interest for us: first the case of the humanoid robot or avatar. In that case, some of the state parameters of the robot are not actuated, and \( J_a \) is a selection matrix. And the case of the cable-driven actuation (typically the human body) where \( J_a \) gives the map of the forces distribution.

The task differential link is obtained similarly as before:
\[ G = J A^{-1} P_c J_a^T \] . When the rank of \( J_a \) is smaller than the configuration of the robot, the system is said under-actuated. However, it may have no direct impact on the task, since \( G \) may stay full rank independently of \( J_a \). However, we can directly notice that a under-actuated system can not control its whole configuration by this approach: when the task space is equal to the configuration space, \( G = A P_c J_a^T \), which has at most the rank of \( J_a \), and is thus rank deficient.

III. Case of the free-floating dynamics

In this section, we show the equivalence of the generalization proposed upper with the work on humanoid robot control using inverse dynamics. The humanoid robot dynamic model can be written:
\[ A\ddot{q} + b + J_c^T f_c = S^T \tau \] (16)
where \( S = [0 \ I] \) is the matrix selecting the actuated joints\(^1\)

A. Control scheme from [31]

We first recall the control law for such a system proposed in [31]. In this work, an equivalent Jacobian \( J^* \) is derived from the supporting-contact constraint, that acts as a classical Jacobian, but respecting naturally the contact constraint. As mentioned in (12), the velocity constraint on the supporting contact implies that the robot velocities (base and joint together) have to belong to the null space of the support Jacobian. Given any velocity \( (v_b, \dot{q}) \), the closest acceptable velocity denoted \( \dot{q}^* \) is simply selected by:
\[ \dot{q}^* = S P_c^T \begin{bmatrix} v_b \\ \dot{q} \end{bmatrix} \]
Solving the above equation gives
\[ \begin{bmatrix} v_b \\ \dot{q} \end{bmatrix} = (S P_c^T)^+ q^* + (I - (S P_c^T)^+ S P_c^T) \begin{bmatrix} v_{b,0} \\ \dot{q}_0 \end{bmatrix} \] (18)
where \( (S P_c^T)^+ \) is a support consistent generalized inverse of \( S P_c^T \), \( (I - (S P_c^T)^+ S P_c^T) \) is a null Space basis associated with the matrix \( S P_c^T \), and \( v_{b,0}, \dot{q}_0 \) are arbitrary vectors of base and joint velocities operating in the null space of \( S P_c^T \). Replacing (18) in (17) leads to the following equality:
\[ \begin{bmatrix} v_b \\ \dot{q} \end{bmatrix} = (S P_c^T)^+ q^* \] (19)

The chosen generalized inverse for \( S P_c^T \) is the \( A^{-1} \)-weighted inverse written as:
\[ (S P_c^T)^+ = A^{-1} P_c S^T (S P_c^T A^{-1} P_c S^T)^{-1} \] (20)
Using (19), the task velocities can be expressed in terms of articulated joint velocities as:
\[ \dot{x} = J \begin{bmatrix} v_b^* \\ q^* \end{bmatrix} = J (S P_c^T)^+ \dot{q}^* \] (21)

In this last expression, we can recognize a classical Jacobian, relying only on the actuated parameters. This equivalent Jacobian is denoted:
\[ J^* = J (S P_c^T)^+ \]

\(^1\)Care has to be taken with \( \ddot{q} \) that can is not the second derivative of the robot configuration, but the derivative of the robot velocity \( \dot{q} \), where the free-floating velocity \( v \) is not integrable.
It is contact consistent, which means that it can now be used directly, without any care to the contact or to the under-actuation. Indeed, the torques that perform the reference task can then simply be written as a transpose of the consistent Jacobian [31]:

\[ \tau = J^*T F \]

where

\[ F = A_\ell|c|\ddot{x} + \mu t|c| \]

and

\[
\begin{cases}
A_\ell|c| = (Jc\|c|A^{-1}Jc\|c|)^{-1} \\
Jt\|c| = JPc\|c| \\
\mu t\|c| = Jc\|c| b - (A_\ell|c|\dot{c}x|c| - Jc\|c| Jc\|c| \dot{c}x|c|) \begin{bmatrix} \nu_b \\ \dot{q} \end{bmatrix}
\end{cases}
\]

B. Proof of the equivalence with the proposed generic scheme

Using (20) and considering only the task corresponding torque part, the torque of (??) can be developed to:

\[ \tau_{ref} = (SPc\|c|A^{-1}PCc\|c|T)\ddot{c}x - (JPc\|c| - Jc\|c|PCc\|c|Jc\|c|)^{-1}\ddot{c}x \]

On the other hand, the scheme proposed in Section II can be written:

\[ \tau = (JA^{-1}Pc\|c|S)\ddot{c}W_{SP}A^{-1}PCc\|c|T(Jpc\|c|A^{-1}Pc\|c|Jc\|c|))^{-1}\ddot{c}x \]

with \( W \) a user-defined weight matrix. Developing the weighted inverse [7] gives:

\[ \tau = WSPc\|c|A^{-1}Jc\|c|T(Jp\|c|A^{-1}Pp\|c|S)SPc\|c|^{-1}SPc\|c|A^{-1}PCc\|c|T(Jp\|c|A^{-1}Pp\|c|Jc\|c|))^{-1}\ddot{c}x \]

Choosing \( W = (SA^{-1}Pc\|c|ST)^{-1} = (SPc\|c|A^{-1}PCc\|c|ST)^{-1} \); since \( A^{-1}Pc = PCcA^{-1} \), we only need to demonstrate that:

\[ JA^{-1}Pc\|c|ST(A^{-1}PCc\|c|ST)^{-1}SPc\|c|Jc\|c| = (JPc\|c|A^{-1}Pc\|c|T) \]

From (20), the previous equation reduces to:

\[ J(SPc\|c|S)SPc\|c|A^{-1}T = (JPc\|c|A^{-1}Pc\|c|T) \]

It has also been demonstrated [31] that \( (SPc\|c|S)SPc\|c| = PCc\|c| \).

Finally:

\[ JPc\|c|A^{-1}T = (JPc\|c|A^{-1}Pc\|c|T) \]

because \( PCc\|c|A^{-1} = Pc\|c|A^{-1}PCc\|c| [31]. \]

IV. INEQUALITIES IN THE LOOP

In the previous sections, all the task introduced where defined by an equality constraint \( \dot{c} = \dot{c}^* \). However, many objective functions to define a motion have to be defined by inequalities. Typically, they are joint limits, obstacles, balance constraint, visibility of landmark in the field of view or behind occlusion, actuator limits, etc. A very well-known solution to handle such constraints are to define a potential function [15], whose gradient acts as a virtual force that drives the robot away from the obstacles [23], or that is used to weight the Jacobian inverse to penalize the motion toward the obstacle [4].

These solutions take advantages of the null-space of the main tasks. However, there is two main limitations. First, the potential function can only be defined at the configuration levels, which prevent for using it to cope with for example actuator limits (actuator limits can not be defined as a function of the configuration). Second, the potential functions can only be taken into account when the robot is redundant with respect to the main tasks.

When there is not enough redundancy to account for the avoidance field, solutions have been proposed to remove some parts of the main task [19] or to add properly-chosen equalities constraint at the top-priority level to prevent any further motion [31], [29]. However, such solutions are very costly when in the neighborhood of several constraints [21].

A. Quadratic programming

It has been proposed in [14] to replace the pseudo inverses used in inverse kinematics by a quadratic solver. Since quadratic solvers are able to handle indifferently equalities and inequalities, it is then possible to have both inequalities and equalities in the task definition. The reference part is then rewritten:

\[ \dot{c}^* \leq \dot{c} \leq \bar{c}^* \]  

with \( \dot{c}^* = \dot{c}^* \) in the case of equalities, and \( \dot{c}^* = -\infty \) or \( \dot{c}^* = +\infty \) to handle single-bounded constraints.

Most of the time, unilateral constraints have priority over any other constraints: typically, joint limits and avoidance would be put above a grasping task. However to handle less-common cases (like the visibility collision when performing a visually-guided grasping), the method proposed in [14] was generalized to enable a hierarchy of any number of task, with inequalities at any levels of the hierarchy. In [8], we have shown that this approach was applicable at low computation cost on full-size systems such as humanoid robot.

1) One task, equalities only: When considering a single task, the inversion (2) corresponds to the optimal solution of the problem:

\[ \min_u ||Gu - \dot{c}^* - \mu||^2 \]  

2) One task, inequalities and equalities together: It is strait-forward to introduce inequalities constraints into a quadratic program. However, this would introduce also a de-facto hierarchy between the inequalities part and the equalities. It was then propose [14], [11] to rely on slack variables. The quadratic program for both equalities and inequalities is then written:

\[ \min_{u,w} ||w||^2 \]

\[ \text{s.t. } \dot{c}^* \leq Gu - \mu + w \leq \bar{c}^* \]

with \( \dot{c}^* = \bar{c}^* \) for the equality parts of the task. The effect of the slack variable is to relax the part of the task that are not feasible, and therefore insures that the task is fulfill at the best (in the sense of the norm of the rest).
3) **Two tasks with priority**: When first task is solved, it was proposed [14] to use the optimal slack denoted \( w^* \) to formalize the hierarchy with a secondary task. After resolution of the first quadratic program, a secondary task is solved by

\[
\min_{u,w_2} \|w_2\|^2
\]

\[s.t. \quad \dot{e}^* - G u - \mu + w^* \leq \tau^* \]

\[
\ddot{e}_2^* - G_2 u - \mu_2 + w_2 \leq \tau_2^*
\]

In this secondary program, the first slack \( w^* \) is not variable anymore. Indeed, the first task is now prioritary, and should be solved as accurately as done by the first program. On the other hand, the second task is not prioritary. If the two task are not compatible, the second task will be relax, and then less accurately executed, due to its slack variable.

Similarly, the second slack can be used to introduce a third task, and iteratively, any number of task.

In [8], this cascade of quadratic program was performed using a dedicated optimization solver. It was possible to resolve a set of 4 tasks, for a total of 45 constraints on a 36-degree of freedom (DOF) humanoid robot in 3ms. In the following, we will use the generalization introduced in Section II to apply the same dedicated solver for a hierarchy of task while accounting of the full dynamic of the robot.

**B. Weighted inverse**

The previous quadratic programs give a least square solution: the solution correspond to a least norm for both the norm of the parameter \( ||u|| \) and the rest of the optimization \( ||\dot{e}^* + \mu - Gu|| \). However, as shown earlier, a norm for a specific weight is preferred for the dynamic inverse. It is easy to show that a weighting of the inverse can be equivalently obtained by adding the following constraint at the lowest priority:

\[
\sqrt{W} u = 0
\]

(30)

with \( \sqrt{W} \) any square root of the weight \( W \): for example, a Cholesky decomposition of \( W \). Indeed, the quadratic rest of such a task is \( u^T W u = ||u||^2_W \).

Therefore, the weighting can be obtained equivalently by adding this quadratic program at the last stage:

\[
\min_u u^T W u
\]

\[s.t. \quad \forall i \quad \dot{e}^*_i - G_i u - \mu_i + w^*_i \leq \tau^*_i
\]

**V. TASK-SET FOR DYNAMIC INVERSE**

For different tasks where used during the experiments. The first one, denoted \( e_q \) is a regulation of the posture of the robot. The task space is equal to the actuated-joint space, and the desired acceleration in this space is a proportional derivative to a given reference position at 0 velocity:

\[
\ddot{q}^* = -\lambda_p (q - \dot{q}^*) - \lambda_D \dot{q}
\]

(32)

In that case, the Jacobian is simply the selection matrix \( S \).

The second task \( e_{ij} \) is a regulation in position and orientation of the position of one point of the robot (for example right hand, or head). The reference acceleration is also a proportional derivative on the robot:

\[
\ddot{x}^* = -\lambda_p \begin{bmatrix} \frac{p}{u \theta} \end{bmatrix} - \lambda_D \dot{x}
\]

(33)

where \( \dot{x} \) is the velocity of the controlled point in its own frame (instantaneous velocity), while \( \frac{p}{u \theta} \) is the difference between the current position-orientation of the controlled point and its desired value.

The third task \( e_{jl} \) is the constraint of joint limits. As previously, it is defined in the actuated-joint space. The task is simply defined by a Euler integration of second order:

\[
\ddot{q} \leq q + \Delta T \dot{q} + \frac{\Delta T^2}{2} \ddot{q} \leq q
\]

(34)

where \( q \) and \( \dot{q} \) are of course known before the resolution.

Finally, the last task \( e_{bal} \) ensure an immediate balance control, by preventing the contact points to take off the ground. The task space is the space of the forces normal to the contact, and the task is to prevent them to reach 0:

\[
f_{\perp} = S_{\perp} J_e \pi^T (S^T \pi - b) \leq \epsilon
\]

(35)

where \( f_{\perp} \) are the normal components of the contact forces \( f_c \), \( S_{\perp} \) is the selection matrix of the corresponding lines of the contact Jacobian \( J_c \), and the link with \( \pi \) is defined by (38). The right-hand side parameter \( \epsilon \) is user defined, to ensure a security margin: if chosen closer to 0, the robot can perform more dynamic movements, but that are less robust in case of perturbation. We have used \( \epsilon = 10N \) in the experiments.

It is possible to show that this last constraint is equivalent to the well-known ZMP constraint [36], [13] when all the contact points are planar and horizontal. Indeed, the ZMP is defined as the barycenter of the contact points weighted by their normal forces:

\[
z = \sum_{i \in c} \pi^T_i \pi^T_i
\]

(36)

The ZMP constraint is then written as the point \( z \) has to stay inside the convex hull of the contact point. In the case of a barycenter, this is equivalent to say that all the weights have to be positive. □

**VI. EXPERIMENTS**

The experiments have been performed in simulation, since it was not possible to access to a humanoid robot with torque-feedback control. We used the dynamic simulator AMELIF [5] that computes the direct dynamics (ie the robot acceleration) from the current configuration and motor torques, resolve the collision and finally integrates the result. The control law has been integrated in the control framework SoT [22], using the dedicated inequalities solver developed for inverse kinematics [8].

We reproduce a well known experiment of physiology: the subject is ask to follow an oscillatory reference with the legs. When the oscillations frequency or amplitude increase, the required acceleration increases, until the natural contact constraint is saturated. An opposite oscillation then naturally
appears on the chest to counteract the oscillation of the legs, and preserve the constraint. When put on a force sensor, the subject ZMP was shown to present saturation at the maximum of the amplitude.

The robot is put on one leg. An oscillatory acceleration is given has a reference, that requires the whole body to remain static, except for one joint of the support leg. The amplitude of the acceleration is then increased until saturation of the support constraint. To prove the generality of the considered balance constraint (with opposition for example to the ZMP constraint), the ground is rotated to an angle of 10deg to the skyline (around the pitch axis of the robot).

The experiment is summed up on figures 1 to 3. The robot configuration at the maximum of the oscillation is shown on Fig. 1. The robot is bent on its left, with the hip roll axis moving. The balance-constraint saturation comes both from the bending (the center of mass of the robot is far from the support polygon), and from the acceleration required to inverse the velocity and come back to the rest position. Fig. 2 gives the normal forces at the four corners of the foot during the motion. The minimal acceptable force is set to 10N. Around iteration 1000, the force corresponding to the front right corner of the foot reached the saturation. A side effect of this saturation is the saturation of the right-back corner force (green), that is not due to a constraint.

VII. CONCLUSION

Based on a normalization of both inverse-kinematics and inverse-dynamics control scheme, a solution was propose to use the hierarchy of quadratic problems to generate dynamical motion of a humanoid robots. We have proved that the given solution was generic enough to include the inverse-dynamics schemes from the state of the art. The proposed schemes was then applied to control the motion of a humanoid robot while keeping its dynamic balance on a non horizontal ground. Many type of constraints and tasks can be accounted with the proposed method. Future works will focus on the integration of the most classical: obstacles, occlusion, and their integration to generate more complex motion.

VIII. APPENDIX: FORCE-FREE DYNAMIC EQUATION

We consider the in-contact dynamic equation (11). In case of rigid contact, $\ddot{x}_c = 0$. It arises then directly that $J_c\ddot{q} = -\dot{J}_c\dot{q}$. It is then possible to link the robot torques to the forces with no dependency on the acceleration, by projecting (11) into $x_c$ (ie multiplication by $J_cA^{-1}$):

$$J_cA^{-1}J_c^T f_c = J_cA^{-1}(\tau - b) - \dot{J}_c\dot{q}$$ (37)
In the basic case, $J_cA^{-1}J_c^T$ is invertible, and $f_c$ is deduced [27]:

$$ f_c = (J_c^T)^#A^{-1}(\tau - b) - J_cA^{-1}J_c^TJ_c\dot{q} $$

(38)

since $(J_cA^{-1})^#A = (J_cA^{-1}J_c^T)^{-1}J_cA^{-1}$ (this equality stands when $J_cA^{-1}J_c^T$ is not invertible, taking $A$ instead of $^{-1}$). The obtained force can reintroduced into (11), to obtain the force-free dynamic equation (13).

A more complete solution for dealing with redundant contact can be found in [34].

REFERENCES


