Hierarchical Quadratic Programming: Fast Online Humanoid-Robot Motion Generation

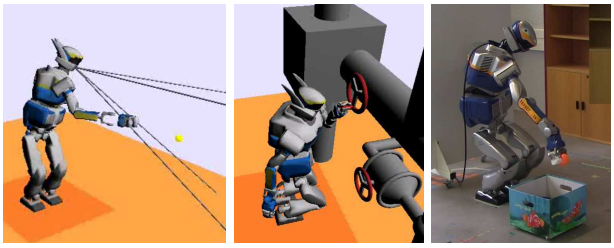
Adrien Escande
JRL-CNRS/AIST
Tsukuba, Japan

Nicolas Mansard
LAAS-CNRS, Univ. Toulouse
Toulouse, France

Pierre-Brice Wieber
INRIA Grenoble
St Ismier, France

Abstract

Hierarchical least-square optimization is often used in robotics to inverse a direct function when multiple incompatible objectives are involved. Typical examples are inverse kinematics or dynamics. The objectives can be given as equalities to be satisfied (e.g. point-to-point task) or as areas of satisfaction (e.g. the joint range). This paper proposes a complete solution to solve multiple least-square quadratic problems of both equality and inequality constraints ordered into a strict hierarchy. Our method is able to solve a hierarchy of only equalities ten times faster than the iterative-projection hierarchical solvers and can consider inequalities at any level while running at the typical control frequency on whole-body size problems. This generic solver is used to resolve the redundancy of humanoid robots while generating complex movements in constrained environment.



(a) (b) (c)

Fig. 1: *Various situations of inequality and equality constraints. (a) reaching a distant object while keeping balance. The visibility and postural tasks are satisfied only if possible. (b) Obstacle avoidance, joint limits and support polygon. (c) Grasping an object on the floor. Sequence of subtasks naturally appears from the use of the hierarchy.*

1 Introduction

1.1 Context

Least squares are a mean to satisfy *at best* a set of constraints that may not be feasible. When the constraints are linear, the least squares are written as a quadratic program, whose solution is given for example by the pseudo-inverse. Linear least squares have been widely used in robot control in the frame of instantaneous task resolution [De Schutter and Van Brussel, 1988], inverse kinematics (IK) [Whitney, 1972] or operational-space inverse dynamics (ID) [Khatib, 1987]. These approaches describe the robot objectives using a function depending on the robot configuration, named the task function [Samson et al., 1991]. Time derivatives of this function depend linearly on the robot velocity or acceleration, which gives a set of linear constraints, to be satisfied at best in the least-square sense.

When the constraint does not require the use of all the robot degrees of freedom (DOF), the remaining DOF can be used to perform a secondary objective. This redundancy was first emphasized in [Liégeois, 1977]. Least squares can be used again to execute at best a secondary objective

using the redundant DOF [Hanafusa et al., 1981], and by recurrence, any number of constraints can be handled [Siciliano and Slotine, 1991]. Here again, the pseudo-inverse is used to compute the least-square optimum. The same technique has been widely used for redundant manipulator [Chiaverini et al., 2008], mobile or underwater [Antonelli and Chiaverini, 1998] manipulator, multi manipulator [Khatib et al., 1996], platoon of mobile robots [Antonelli and Chiaverini, 2006],visual servoing [Mansard and Chaumette, 2007], medical robots [Li et al., 2012], planning under constraints [Berenson et al., 2011], control of the dynamics [Park and Khatib, 2006], etc. Such a hierarchy is now nearly systematically used in humanoid animation [Baerlocher and Boulic, 2004] and robotics [Sian et al., 2005, Mansard et al., 2007, Khatib et al., 2008].

Very often, the task objectives are defined as equality constraints. On the opposite, some objectives would naturally be written as inequality constraints, such as joints limits [Liégeois, 1977, Chaumette and Marchand, 2001], collision avoidance [Marchand and Hager, 1998, Stasse et al., 2008], singularities avoidance [Yoshikawa, 1985] or visibility [Garcia-Aracil et al., 2005, Remazeilles et al., 2006] (see Fig. 1). A first solution to account for these tasks is to embed the corresponding inequality constraint into a potential function [Khatib, 1986] such as a log barrier function, whose gradient is projected into the redundant DOF let free by the first objective [Liégeois, 1977, Gienger et al., 2006]. The potential function simultaneously prevents the robot to enter into a forbidden region and pushes it away from the obstacles when it is coming closer. The potential function in fact transforms the inequality into an equality constraint, which is always applied even far from the obstacle. However, this last property prevents the application of this solution for top-priority constraints.

To ensure that avoidance is realized whatever the situation (and not only when there is enough DOF), several solutions have been proposed, that try to specify inequality objectives as higher-priority tasks. In [Nelson and Khosla, 1995], a trade-off between the avoidance and the motion objectives was performed. In [Chang and Dubey, 1995], the joint limit avoidance was used as a damping factor to stop the motion of a joint close to the limit. Both solutions behave improperly when the motion objective and the avoidance criteria become incompatible. In [Mansard and Chaumette, 2007], the constraints having priority were relaxed to improve the execution of the avoidance constraint, set at a lower priority level. However, this solution is only valid far from the target point, and the obstacle may finally collide at the convergence of the motion task. An improvement is done by temporarily relaxing the most distant DOF in [Mansard and Chaumette, 2009], but that cannot solve the main problem. In the cases where damping is not sufficient, clamping was proposed [Raunhardt and Boulic, 2007]. It was applied for example to avoid the joint limits of a humanoid [Sentis, 2007]. However, this solution requires several iterations and might be costly. Moreover, it is difficult to relax a DOF that was clamped. In [Mansard et al., 2009], it was proposed to realize an homotopy between the control law with and without avoidance. A proper balance of the homotopy factors ensures that the objectives having priority are respected. However, the cost of this solution in complex cases is prohibitive. A reduction was proposed in [Lee et al., 2012] in the case of joint limits but involve very specific study for each new type of task.

Alternatively, the inequality constraint can be included in the least-square program as it is [Nenchev, 1989, Sung et al., 1996, Decré et al., 2009]. Such a solution is very popular for controling the dynamic of the simulated system [Collette et al., 2007, Salini et al., 2009, Bouyarmane and Kheddar, 2011]. These papers can consider inequality constraints but lose the possibility of enforcing priorities among them. The earliest and most obvious approach to solve hierarchical optimization problems is to solve single-objective optimization problems successively, in *cascade* [Behringer, 1977]. A dedicated simplex solver was designed in [Isermann, 1982] for linear problem only. Cascade was then applied to quadratic problems in [Kanoun et al., 2011], which gives the

first definition of a hierarchical quadratic program along with a simple yet limited resolution. A simplified version was proposed in [De Lasa et al., 2010], that improves the computation cost but prevents the inclusion of inequality except at the top priority. Specific work on the solver linear decomposition was used in [Escande et al., 2010] to avoid repetitive computations, reducing the resolution cost while keeping a generic formulation.

Before defining the objectives and specificities of our approach, we rewrite briefly the main resolution schemes for hierarchy of quadratic problems (with and without inequalities) in the next sections.

1.2 From inverse kinematics to least square

The task-function approach [Samson et al., 1991] is an elegant solution to express the objectives to be performed by the robot and deduce from this expression the control to be applied at the joint level. Consider a robot defined by its configuration vector q and whose control input is the joint velocity $\dot{q}$. A task function is any derivable function e of q. The evolution in the image space (or task space) with respect to the robot input is given by $\dot{e} = J\dot{q}$, with $J = \frac{\partial e}{\partial q}$ the task Jacobian.

The objective to be accomplished by the robot can then be expressed in the task space by giving a reference task velocity $\dot{e}^*$. Computing the robot input boils down to solving the following quadratic least-square program (QP):

$$\text{Find } \dot{q}^* \in \underset{\dot{q}}{\text{Arg min}} \, \| J\dot{q} - \dot{e}^* \| \tag{1}$$

More generally, many robotic schemes rely on solving linear equalities. For example, such a QP formulation can also be encountered to inverse the system dynamics in the operational space [Khatib, 1987, Collette et al., 2007], compute a walking pattern [Herdt et al., 2010] or optimize a linearized trajectory of the robot whole body [Pham and Nakamura, 2012]. The variables are for example the joint accelerations, joint torques and contact forces in the first case, the third derivative of the center of mass for the walking pattern, the coefficients of an affine transformation in the last case, and many types of equalities can be devised. Hierarchy of tasks has been widely used in inverse dynamics [Khatib et al., 2008, Mansard et al., 2009, Wensing and Orin, 2013]. In that case, the solver would typically search for the system acceleration and forces under the constraint of satisfying the dynamic consistency. We can foresee some interesting applications in optimal control too.

Consequently, we abstract in this paper these multiple contexts by considering a set of linear equality constraints $Ax = b$. In case this set of constraints is not feasible, it has to be satisfied at best in the least-square sense:

$$\text{Find } x^* \in \underset{x}{\text{Arg min}} \, \| Ax - b \| \tag{2}$$

Among the possible x^*, the solution that minimizes the norm of x^* is given by the pseudo-inverse:

$$x^* = A^+ b \tag{3}$$

where A^+ is the (Moore-Penrose) pseudo-inverse of A.

The set of all the solutions to (2) is given by [Liégeois, 1977]:

$$x^* = A^+ b + P\tilde{x}_2 \tag{4}$$

where P is a projector on the null space of A (*i.e.* such that $AP = 0$ and $PP = P$), and $\tilde{x}_2$ is any arbitrary vector of the parameter space, that can be used as a secondary input to satisfy a second objective.

1.3 Hierarchy of equality constraints [Siciliano and Slotine, 1991]

In this paper, we consider the case where p linear constraints (A_1, b_1) ... (A_p, b_p) have to be satisfied at best simultaneously. If the constraints do not conflict, then the solution is directly obtained by stacking them into a single constraint:

$$\underline{A}_p = \begin{bmatrix} A_1 \\ \vdots \\ A_p \end{bmatrix}, \quad \underline{b}_p = \begin{bmatrix} b_1 \\ \vdots \\ b_p \end{bmatrix} \tag{5}$$

The resulting QP can be solved as before. If the constraints are conflicting, a weighting matrix Q is often used to give more importance to some constraints with respect to others or to artificially create a balance between objectives of various physical dimensions:

$$\min_x \ (\underline{A}_p x - \underline{b}_p)^T Q (\underline{A}_p x - \underline{b}_p) \tag{6}$$

Rather than selecting a-priori values of Q, it was proposed in [Siciliano and Slotine, 1991] to impose a strict hierarchy between the constraints. The first constraint (with highest priority) (A_1, b_1) will be solved at best in the least-square sense using (3). Then the second constraint (A_2, b_2) is solved in the null space of the first constraint without modifying the obtained minimum of the first constraint. Introducing (4) in $A_2 x = b_2$, a QP in $\tilde{x}_2$ is obtained:

$$\min_{\tilde{x}_2} \ \|A_2 P_1 \tilde{x}_2 - (b_2 - A_2 A_1^+ b_1)\| \tag{7}$$

The generic solution to this QP is:

$$\tilde{x}_2^* = (A_2 P_1)^+ (b_2 - A_2 A_1^+ b_1) + \tilde{P}_2 \tilde{x}_3 \tag{8}$$

with $\tilde{P}_2$ the projector into the null space of $(A_2 P_1)^+$ and $\tilde{x}_3$ any vector of the parameter space that can be used to fulfill a third objective. The complete solution solving (A_1, b_1) at best and (A_2, b_2) if possible is:

$$x_2^* = A_1^+ b_1 + (A_2 P_1)^+ (b_2 - A_2 A_1^+ b_1) + P_2 \tilde{x}_3 \tag{9}$$

where x_2^* denotes the solution for the hierarchy composed of the two first levels, and $P_2 = P_1 \tilde{P}_2$ is the projector over $\underline{A}_2$.

This solution can be extended recursively to solve the p levels of the hierarchy [Siciliano and Slotine, 1991]:

$$x_p^* = \sum_{k=1}^{p} (A_k P_{k-1})^+ (b_k - A_k x_{k-1}^*) + P_p \tilde{x}_{p+1} \tag{10}$$

with $P_0 = I$, $x_0 = 0$ and $P_k = P_{k-1} \tilde{P}_k$ the projector into the null space of $\underline{A}_k$ [Baerlocher and Boulic, 2004]. $\tilde{x}_{p+1}$ is any vector of the parameter space that denotes the free space remaining after the resolution of the whole hierarchy.

1.4 Projection versus basis multiplication [Escande et al., 2010]

Given a basis Z_1 of the null space of A_1 (*i.e.* $A_1 Z_1 = 0$ and $Z_1^T Z_1 = I$), the projector in the null space of A_1 can be written $P_1 = Z_1 Z_1^T$. In that case, it is easy to show that

$$(A_2 P_1)^+ = (A_2 Z_1 Z_1^T)^+ = Z_1 (A_2 Z_1)^+ \tag{11}$$

The last writing is more efficient to compute than the first one due to the corresponding matrix sizes. Then, (10) can be rewritten equivalently under a more efficient form:

$$x_p^* = \sum_{k=1}^{p} Z_{k-1} (A_k Z_{k-1})^+ (b_k - A_k x_{k-1}^*) + Z_p z_{p+1} \tag{12}$$

where Z_k is a basis of the null space of $\underline{A}_k$ and z_{p+1} is a vector of the dimension of the null space of $\underline{A}_p$. This observation was exploited in [Escande et al., 2010] (which constitute a preliminary version of this work) to fasten the computation of (10).

1.5 Inequalities inside a cascade of QP [Kanoun et al., 2011]

The problem (2) is an equality-only least-square quadratic program (eQP). Searching a vector x that satisfy a set of linear inequalities is straightforward to write:

$$\text{Find } x^* \in \{x, \ s.t. \ Ax \leq b\} \tag{13}$$

If the polytope defined by $Ax \leq b$ is empty (the set of constraints is infeasible), then x^* can be searched as before as a minimizer in the least-square sense. The form (2) can be extended to inequalities by introducing an additional variable w, named the slack variable, in the parameter vector:

$$\min_{x,w} \|w\| \tag{14}$$

$$\text{subject to} \quad Ax \leq b + w \tag{15}$$

The slack variable can relax the constraint in case of infeasibility [Hofmann et al., 2009]. This QP is

named a inequality QP (iQP, by opposition to the eQP). In the remaining of the paper, we keep this reduced shape with only upper bound, since it encompasses lower bounds $Ax \geq b$, double bounds $b_- \leq Ax \leq b_+$ and equalities $Ax = b$ by setting respectively $-Ax \leq -b$, $\begin{bmatrix} -A \\ A \end{bmatrix} x \leq \begin{bmatrix} -b_- \\ b_+ \end{bmatrix}$ and $\begin{bmatrix} -A \\ A \end{bmatrix} x \leq \begin{bmatrix} -b \\ b \end{bmatrix}$. Such an iQP can be solved, for example, using an active-search method.

The work in [Siciliano and Slotine, 1991] is limited to a hierarchy of eQP. In [Kanoun et al., 2011], a complete solution to extend the hierarchy to inequality constraints was proposed. The method begins with minimizing the violation $\|w_1\|$ of the first level of constraints in a least-squares sense through (14). This gives a unique optimal value w_1^* since the cost function is strictly convex in w_1. It proceeds then in minimizing the violation of the second level of constraints in a least-squares sense:

$$\min_{x,w_2} \|w_2\| \tag{16}$$

$$\text{subject to} \quad A_1 x \leq b_1 + w_1^* \tag{17}$$

$$A_2 x \leq b_2 + w_2 \tag{18}$$

The first line of constraints (17) is expressed with respect to the fixed value w_1^* obtained from the first QP, which ensures that the new x will not affect the first level of constraints and therefore enforces a strict hierarchy. In that sense, (17) is a strict constraint while (18) is a relaxed constraint. The same process is then carried on through all p levels of priority.

1.6 Reduction of the computation cost [De Lasa et al., 2010]

The solution [Kanoun et al., 2011] makes it possible to solve hierarchies of iQP. However, it is very slow, since each constraint k is solved at the iQP of level k and all the following ones. In particular, the first constraint is solved p times.

In [De Lasa et al., 2010], a solution is proposed to lower the computation cost by reducing the generic nature of the problem studied in [Kanoun et al., 2011]: inequalities are considered only at the first level, and this level is supposed feasible. This hypothesis reduces the expressiveness of the method, forbidding the use of weak constraints such as visibility or preference area. However, this expressivity reduction enables to obtain very impressive result for walking, jumping or, as shown in [Mordatch et al., 2012], for planning contacts and manipulation.

The first iQP of the cascade does not need an explicit computation, since $w_1^* = 0$ by hypothesis. Then each level $k > 2$ is solved in the null space of the levels 2 to $k-1$:

$$\min_{z_k,w_k} \|w_k\| \tag{19}$$

$$\begin{aligned} \text{subject to} \quad A_1(x_{k-1}^* + Z_{k-1} z_k) &\leq b_1 && (20) \\ A_k(x_{k-1}^* + Z_{k-1} z_k) &= b_k + w_k && (21) \end{aligned}$$

where x_{k-1}^* is the optimal solution for the $k-1$ first levels, and Z_{k-1} is the null space of the levels 2 to $k-1$. The solution in the canonical basis after the k^{th} QP is set to $x_k^* = x_{k-1}^* + Z_{k-1} z_k^*$. The Z_k basis is computed from Z_{k-1} and a singular value decomposition (SVD) of $(A_k Z_{k-1})$.

If the first level is empty (or equivalently, if the bounds are wide enough for never being activated), this method is equivalent to (12). The global working scheme of the method [De Lasa et al., 2010] is the same as [Kanoun et al., 2011] since both rely on a cascade of QP computed successively for each level of the hierarchy. The method of [De Lasa et al., 2010] is faster since each QP is smaller than the previous one (the dimension of z_k decreases with k), while each QP of [Kanoun et al., 2011] was bigger than the previous ones (the number of constraints increases). The method of [De Lasa et al., 2010] requires an additional SVD, but this could be avoided since the SVD or an equivalent decomposition is already computed when solving the corresponding QP.

However, both methods [Kanoun et al., 2011] and [De Lasa et al., 2010] have the same intrinsic problem due to the nature of the underlying active search algorithm. Basically, it searches for the set of active constraints that holds as equality at the optimum. At each new QP of the cascade, the optimal active

set may be completely different. The active search may then activate and deactivate a constraint several times when moving along the cascade and the succession of all these iterative processes appears to be very inefficient in the end.

Typical examples of this situation are given in Section 6. Consider a humanoid robot that should keep its center of mass inside the support polygon, put its right hand in the front and its left hand far in the back: when solving the right-hand constraint, the center of mass will saturate in the front, which activates the corresponding constraint. The front constraint is then deactivated when the left-hand constraint brings the center of mass on the back, while the back of the support polygon becomes active. The back constraint may even be deactivated if a last level is added that regulates the robot posture.

1.7 Directions and objectives

By minimizing successively $\|w_1\|$, $\|w_2\|$ until $\|w_p\|$, the above approaches end up with a sequence of optimal objectives $\left\{\|w_1^*\|, \|w_2^*\|, \ldots, \|w_p^*\|\right\}$ which is itself minimal with respect to a lexicographic order: it is not possible to decrease an objective $\|w_k\|$ without increasing an objective $\|w_j\|$ with higher priority ($j < k$). Considering a hierarchy between these objectives or a lexicographic order appear to be synonyms. The above approaches can therefore be summarized as a lexicographic multi-objective least-squares quadratic problem, looking for

$$\underset{x, w_1 \ldots w_p}{\operatorname{lex\,min}} \ \{\|w_1\|, \|w_2\|, \ldots, \|w_p\|\} \qquad (22)$$

$$\text{subject to} \quad \forall k = 1 : p, \ A_k x \leq b_k + w_k$$

Using this formulation, the hierarchical quadratic program (HQP) appears more clearly as a single problem. In this paper, we propose to keep this unity in the resolution scheme, by providing a solver that considers all the priority levels at the same time, which appears to be much more efficient. Moreover, the monolithic resolution also enables us to warm-start the solver, which is very important to enables fast resolution when it is used to compute a control law. Beside fast computation, it also enables the algorithm to guarantee the convergence time (real-time answer), which is also very important in control.

Our solver is based on the active-set method, where an activation loop iteratively computes the optimum of a sub-problem considering only equalities. In Section 2, we then focus on equality-only and describe a hiearchical resolution scheme, equivalent to the very classical robotics [Siciliano and Slotine, 1991] method but up to ten times more efficient. The hierarchical active-set algorithm is then described in Section 3. In Section 4, we consider more specifically the continuation property of the hierarchical problem, which is important if it used in a control framework, and we explained how this property can be used to obtain fast computations using warm-start. The method efficiency is demonstrated by generating whole body movements on a humanoid robot using IK. The robotics setup is described in Section 5. Finally, the solver is compared in this context to our implementation of the concurrent methods, and is proven to be much more efficient in practice.

2 Equality hierarchical quadratic program

We propose in this section a method to solve a hierarchy of linear equality in the least-square sense. This resolution of the primal optimum of this problem equivalent to iterative solution proposed in [Siciliano and Slotine, 1991] using the task redundancy [Liégeois, 1977], upon which many robotics control schemes are based [Chiaverini, 1997, Baerlocher and Boulic, 2004, Sian et al., 2005, Khatib et al., 2008, Berenson et al., 2011], to cite a few. We propose an original decomposition that encompasses the hierarchy among the constraints. Using the best expertise of numerical mathematics, this decomposition provides a very fast computation of the primal.

We also provide an expression and an algorithm for the dual optimum, which is less used in robotics. A rough approximation of it for hierarchy of constraints was proposed in [Mansard and Chaumette, 2007].

The dual can quantify the involvement of each constraint to the primal optimum, and is very useful to predict which constraint could be relaxed, for example in the active-set algorithm. In simple case like joint limits [Raunhardt and Boulic, 2007], it was used to relax a saturated constraint.

The optimality conditions are first expressed in Sec. 2.1, upon which the decomposition dedicated to hierarchical problem is built Sec. 2.3. The primal then dual optimums are finally expressed in Sec. 2.4 and 2.5 respectively.

2.1 Optimality conditions

At first, we consider an equality-only hierarchical quadratic least-square program (eHQP). It is written as a set of p eQP: at level k, the QP to be solved is written:

$$\min_{x_k, w_k} \|w_k\| \tag{23}$$

$$\begin{aligned} \text{subject to} \quad A_k x_k &= b_k + w_k && (24)\\ \underline{A}_{k-1} x_k &= \underline{b}_{k-1} + \underline{w}^*_{k-1} && (25) \end{aligned}$$

where $\underline{A}_{k-1}$, $\underline{b}_{k-1}$ and $\underline{w}^*_{k-1}$ are the matrix and vectors composed of the stacked quantities of levels 1 to $k-1$ (by convention, they are empty matrix and vectors for $k-1=0$). $\underline{w}^*_{k-1}$ is the fixed value obtained from the previous QP. The optimality conditions of this problem are

$$\begin{aligned} w_k &= A_k x_k - b_k && (26)\\ \underline{A}_{k-1} x_k &= \underline{b}_{k-1} + \underline{w}^*_{k-1} && (27)\\ \lambda_k &= w_k && (28)\\ \underline{A}^T_{k-1} \underline{\lambda}_k &= -A^T_k w_k && (29) \end{aligned}$$

where $\underline{\lambda}_k$ and λ_k are the Lagrange multipliers corresponding respectively to (25) and (24).

The two first lines give the condition to compute the primal optimum. From (28), we see that w is indeed at the same time a primal and a dual variable. The last equation gives the condition to compute the dual optimum.

2.2 Complete orthogonal decomposition

In the first level, (25) and (27) are empty. The primal optimum is computed by minimizing w_1 in (26), that is to say by a classical pseudo-inverse of A_1. The pseudo-inverse can be computed by performing a complete rank revealing decomposition. In robotics a SVD is often chosen. Alternatively, a complete orthogonal decomposition (COD) can be used[1] [Golub and Van Loan, 1996]:

$$A_1 = \begin{bmatrix} V_1 & U_1 \end{bmatrix} \begin{bmatrix} 0 & 0 \\ L_1 & 0 \end{bmatrix} \begin{bmatrix} Y_1 & Z_1 \end{bmatrix}^T = U_1 L_1 Y_1^T \tag{30}$$

where $W_1 = \begin{bmatrix} V_1 & U_1 \end{bmatrix}$ and $\begin{bmatrix} Y_1 & Z_1 \end{bmatrix}$ are two orthonormal matrices, U_1 being a basis of the range space of A_1, Z_1 of its kernel and L_1 is a lower triangular matrix whose diagonal is strictly nonzero. If the first level (A_1, b_1) is feasible, then A_1 is full row rank and U_1 is the identity (V_1 is empty). In that case, (30) is the QR decomposition of A_1^T (or LQ decomposition of A_1).

The pseudo-inverse of A_1 now only implies the easily computable inversion of L_1:

$$A_1^+ = \begin{bmatrix} Y_1 & Z_1 \end{bmatrix} \begin{bmatrix} 0 & L_1^{-1} \\ 0 & 0 \end{bmatrix} \begin{bmatrix} V_1 & U_1 \end{bmatrix}^T = Y_1 L_1^{-1} U_1^T \tag{31}$$

The optimal solution x_1^* is obtained by

$$x_1^* = A_1^+ b_1 = Y_1 L_1^{-1} U_1^T b_1 \tag{32}$$

Rather than computing the explicit pseudo-inverse, the optimum should be computed by realizing a forward substitution of L_1 on $U_1^T b_1$.

The corresponding slack variable is:

$$w_1^* = A_1 x_1^* - b_1 = U_1 U_1^T b_1 - b_1 = -V_1 V_1^T b_1 \tag{33}$$

[1] The COD is cheaper to compute than the SVD. The algorithms to compute it involve a rather simple sequence of basic transformations and are known to be nearly as robust as the algorithms computing the SVD (and much easier to implement). As a matter of fact, the conditioning advantage of the SVD over the COD will come into play when we are so close to singularity compared to the machine precision that the situation already is problematic from a robotics point of view. It is one of the classical ways to solve rank deficient quadratic least-squares problems [Björck, 1996].

2.3 Hierarchical complete orthogonal decomposition

Consider now the second level of the hierarchy (23) ($k = 2$). As in Sec. 1.4, condition (27) can be rewritten using (32) and (33) as:

$$x_2 = x_1^* + Z_1 z_2 \tag{34}$$

where z_2 is any parameter of the null space of A_1. Condition (26) is then written:

$$w_2 = (A_2 Z_1) z_2 - (b_2 - A_2 x_1^*) \tag{35}$$

$$= \begin{bmatrix} A_2 Y_1 & A_2 Z_1 \end{bmatrix} \begin{bmatrix} Y_1^T x_1^* \\ z_2 \end{bmatrix} - b_2 \tag{36}$$

because $Y_1 Y_1^T x_1^* = x_1^*$. The matrix A_2 is in fact separated in two parts along the Y_1, Z_1 basis: the first part $A_2 Y_1$ corresponds to the coupling between the two first levels. The corresponding part of the parameter space has already been used for the level 1 and cannot be used here. The second part $A_2 Z_1$ corresponds to the free space that can be used to solve the second level.

The optimums x_2^* and w_2^* are obtained by performing the pseudo-inverse of $A_2 Z_1$ using its COD:

$$(A_2 Z_1) = \begin{bmatrix} V_2 & U_2 \end{bmatrix} \begin{bmatrix} 0 & 0 \\ L_2 & 0 \end{bmatrix} \begin{bmatrix} \tilde{Y}_2 & \tilde{Z}_2 \end{bmatrix}^T \tag{37}$$

The basis $W_2 = \begin{bmatrix} V_2 & U_2 \end{bmatrix}$ provides a decomposition of the image space of A_2 along its range space and the orthogonal to it. The basis $\begin{bmatrix} Y_2 & Z_2 \end{bmatrix} = Z_1 \begin{bmatrix} \tilde{Y}_2 & \tilde{Z}_2 \end{bmatrix}$ is in fact another basis of the null space of A_1 that also provides a separation of the kernel of A_2. In particular, Z_2 is a basis of the null space of both A_1 and A_2 that can be used to perform the third level.

The optimum x_2^* is finally:

$$x_2^* = x_1^* + Z_1 (A_2 Z_1)^+ (b_2 - A_2 x_1^*) + Z_2 z_3 \tag{38}$$

$$= x_1^* + Y_2 L_2^{-1} U_2^T (b_2 - A_2 x_1^*) + Z_2 z_3 \tag{39}$$

$$= \begin{bmatrix} Y_1 & Y_2 & Z_2 \end{bmatrix} \begin{bmatrix} Y_1^T x_1^* \\ Y_2^T \tilde{x}_2^* \\ z_3 \end{bmatrix} \tag{40}$$

where $\tilde{x}_2^* = Y_2 L_2^{-1} U_2^T (b_2 - A_2 x_1^*)$ is the contribution of the second level to the optimum and z_3 is any parameter of the null space Z_2 used to perform the following levels. The optimum w_2^* is directly obtained using (36). This can be written using the two basis V_2, U_2 and Y_1, Y_2, Z_2:

$$w_2^* = \begin{bmatrix} V_2 & U_2 \end{bmatrix} \begin{bmatrix} N_2 & 0 & 0 \\ M_2 & L_2 & 0 \end{bmatrix} \begin{bmatrix} Y_1^T x_1^* \\ Y_2^T x_2^* \\ z_3 \end{bmatrix} - b_2 \tag{41}$$

with $M_2 = U_2^T A_2 Y_1$ and $N_2 = V_2^T A_2 Y_1$ the coupled parts of A_2 corresponding respectively to its feasible space U_2 and its orthogonal, and using $Y_2^T \tilde{x}_2^* = Y_2^T x_2^*$.

In (41) a decomposition of the matrix A_2 appears, that can be written generically for any $k \geq 2$:

$$A_k = \begin{bmatrix} V_k & U_k \end{bmatrix} \begin{bmatrix} N_k & 0 & 0 \\ M_k & L_k & 0 \end{bmatrix} \begin{bmatrix} \underline{Y}_{k-1} & Y_k & Z_k \end{bmatrix}^T \tag{42}$$

$$= W_k H_k \underline{Y}_k^T \tag{43}$$

with $N_k = V_k^T A_k \underline{Y}_{k-1}$, $M_k = U_k^T A_k \underline{Y}_{k-1}$, $\underline{Y}_{k-1} = \begin{bmatrix} Y_1 & \dots & Y_{k-1} \end{bmatrix}$ and $H_k = \begin{bmatrix} N_k & 0 \\ M_k & L_k \end{bmatrix}$.

Stacking all the decompositions (42) for the k first levels, a single decomposition of $\underline{A}_k$ is recursively obtained by:

$$\begin{bmatrix} \underline{A}_{k-1} \\ A_k \end{bmatrix} = \begin{bmatrix} \underline{W}_{k-1} & 0 \\ 0 & W_k \end{bmatrix} \begin{bmatrix} \underline{H}_{k-1} & 0 & 0 \\ N_k & 0 & 0 \\ M_k & L_k & 0 \end{bmatrix} \begin{bmatrix} \underline{Y}_{k-1} & Y_k & Z_k \end{bmatrix}^T$$

$$= \underline{W}_k \underline{H}_k \underline{Y}_k^T \tag{44}$$

The complete form for the p levels is finally:

$$\begin{bmatrix} A_1 \\ \vdots \\ A_p \end{bmatrix} = \begin{bmatrix} W_1 & & \\ & \ddots & \\ & & W_p \end{bmatrix} \begin{bmatrix} 0 & 0 & 0 & 0 & 0 \\ \boxed{L_1} & 0 & 0 & 0 & 0 \\ \boxed{\begin{matrix} N_2 \\ M_2 \end{matrix}} & \boxed{L_2} & 0 & 0 & 0 \\ \vdots & \vdots & \vdots & \vdots & \\ \boxed{\begin{matrix} N_p \\ M_p \end{matrix}} & & & 0 & 0 \\ & & & \boxed{L_p} & 0 \end{bmatrix} Y^T$$

with $Y = \begin{bmatrix}\underline{Y}_p & Z_p\end{bmatrix}$.

If all the levels are feasible, all the matrices A_k and $A_k Z_{k-1}$ are full row rank and all the N_k matrices are empty. In this case, the decomposition is a COD of $\underline{A}_p$. If there is no conflict between the levels, all the N_k are zero. In this case, it is only a matter of row permutations to turn the above decomposition into a perfect COD of the matrix $\underline{A}_p$, involving an invertible lower triangular matrix. This decomposition, which has been designed to enforce a strict hierarchy between different priority levels, looks close to a classical COD and is indeed a COD in particular cases. For this reason, we propose to call this decomposition a Hierarchical Complete Orthogonal Decomposition (HCOD) of the matrix $\underline{A}_p$.

The HCOD reveals a lot of information about the structure of the problem. In particular, algorithmic singularities [Chiaverini, 1997] appear in the zeros above each L_k. The kinematic singularities corresponds to zero row in the N_k. Conflicts between tasks can be quantified by looking at large columns in the N_k: for a singular level k, a large column in N_k indicates which lower level should be relaxed to leave the singularity of level k. The redundancy of the whole problem is described by the last column of zeros and then can be used through Z_p. And, as with the SVD, small values on the diagonal of the L_k indicate the proximity of a singularity. All the indicators can be used when solving a control problem to diagnose and improve the behavior of the controller.

2.4 Primal optimum and hierarchical inverse

2.4.1 Computing x_p^*

We have seen in (39) that the primal optimum of the second level x_2^* is directly computed from H_2 and x_1^*. Using the same reasoning, the optimum of level k, given by (12), is computed using H_k:

$$x_k^* = x_{k-1}^* + Y_k L_k^{-1} U_k^T (b_k - A_k x_{k-1}^*) + Z_k z_{k+1} \quad (45)$$

The least norm solution for x_k^* is obtained for every $z_{i+1} = 0$, what we suppose from now on. As observed in this equation, the optimum of each level k is computed using the level k of the HCOD and the optimum of level $k-1$. By recurrence, x_p^* can be computed directly using the HCOD, by reformulating (45) in the following way:

$$x_p^* = \underline{A}_p^\ddagger \underline{b}_p \quad (46)$$

where $\underline{A}_p^\ddagger$ is defined by a matrix recursion:

$$\underline{A}_k^\ddagger = \begin{bmatrix}(I - Y_k L_k^{-1} U_k^T A_k)\underline{A}_{k-1}^\ddagger & Y_k L_k^{-1} U_k^T\end{bmatrix} \quad (47)$$

Or more simply, with the HCOD:

$$\underline{A}_p^\ddagger = \underline{Y}_p \, \underline{H}_p^\ddagger \, \underline{W}_p^T \quad (48)$$

with

$$\underline{H}_k^\ddagger = \begin{bmatrix}\underline{H}_{k-1}^\ddagger & 0 & 0 \\ -L_k^{-1} M_k \underline{H}_{k-1}^\ddagger & 0 & L_k^{-1}\end{bmatrix} \quad (49)$$

Matrix $\underline{A}_p^\ddagger$ respects three of the four Moore-Penrose conditions used to define the pseudo-inverse (as shown in App. B.1). It has been designed to enforce a strict hierarchy between different priority levels and looks close to a pseudo-inverse. For this reason, we propose to call this matrix the *hierarchical inverse* of the matrix $\underline{A}_p$.

2.4.2 Optimum structure

The optimum is structured by layer, following the hierarchy of problems. This structure is more evident when the optimum is computed in the Y basis. The primal optimum in the Y basis is denoted $\underline{y}_k^* = \underline{Y}_k^T x_k^*$. The contribution $x_k^* - x_{k-1}^*$ of the level k to the primal optimum is denoted $y_k^* = Y_k^T (x_k^* - x_{k-1}^*)$ $(= Y_k^T x_k^*$ since $Y_k^T x_{k-1}^* = 0)$. Then (45) can be rewritten as:

$$\underline{y}_k^* = \begin{bmatrix}\underline{y}_{k-1}^* \\ y_k^*\end{bmatrix} \quad (50)$$

where $y_k^* = L_k^{-1}(U_k^T b_k - M_k \underline{y}_{k-1}^*)$. Each component k of the optimum vector $\underline{y}_p^* = (y_1^*, \ldots, y_p^*)$ corresponds to the contribution of the level k of the hierarchy. The study of $\underline{y}_p^*$ is thus very informative to understand the obtained x_p^*: for example the hierarchy levels that induce large contributions in x_p^* directly appear in $\underline{y}_p^*$.

Algorithm 1 Primal eHQP

function eHQP_primal($\underline{A}_p$, $\underline{b}_p$)
Input: HCOD of $\underline{A}_p$, $\underline{b}_p$
Output: $x_p^*, \underline{w}_p^*$ minimizing (23)
$\underline{y}_0^* := []$
for k in 1:p **do**
  $e = U_k^T b_k - M_k \underline{y}_{k-1}^*$
  $w_k^* = V_k(N_k \underline{y}_{k-1}^* - V_k^T b_k)$
  $e := L_k^{-1} e$
  $\underline{y}_k^* := [\ \underline{y}_{k-1}^*\ ;\ e\ ]$
end for
$x_p^* = \underline{Y}_p \underline{y}_p^*$, $\underline{w}_p^* = (w_1^*, ..., w_p^*)$
return $x_p^*, \underline{w}_p^*$

As said above, the primal optimum is equivalent to the solution computed by the iterative projection in [Siciliano and Slotine, 1991]. This clearly appears in (45), with $Y_k L_k^{-1} U_k^T$ being the projected Jacobian. The similarities are less evident in (49), but the HCOD interestingly reveals a sparsity, that is used in Alg. 1 to reduce the amount of computation. The optimum in the HCOD basis $\underline{y}_p^*$ also reveals some information about the hierarchical structure of the problem: the levels with the high participation in the whole problem appear as large coefficient of $\underline{y}_p^*$. On the robot, this can be used to understand the output control and if necessary change the behavior of the robot by removing or damping some of the tasks.

2.4.3 Computing the $\underline{w}_p^*$

For each level k, the slack variable is directly obtained from x_k^* using (26). By replacing A_k by $W_k H_k \underline{Y}_k^T$ and x_k^* by (50), we obtain:

$$w_k^* = V_k N_k \underline{y}_{k-1}^* - V_k V_k^T b_k \quad (51)$$
$$= V_k V_k^T (A_k x_{k-1}^* - b_k) \quad (52)$$

The algorithm to compute x_p^* and $\underline{w}_p^*$ is summarized in Alg. 1.

2.5 Transposed hierarchical inverse and dual optimum

2.5.1 Dual expression

At level k, the dual optimum is given by (29), recalled here:

$$\underline{A}_{k-1}^T \underline{\lambda}_k = -A_k^T w_k$$

A solution (the least-square one) to this second optimality condition can be obtained with the transpose of the hierarchical inverse:

$$\underline{\lambda}_k = -\underline{A}_{k-1}^{\ddagger T} A_k^T w_k^* \quad (53)$$

A quick proof is given in App. B.1. For each level k, there is a multiplier $\underline{\lambda}_k$ that corresponds to all the level of higher priority. There is no real sense in stacking the multipliers of each level. They can be summarized under a matrix structure:

$$\Lambda_p = \begin{bmatrix} w_1^* & {}^1\underline{\lambda}_2 & {}^1\underline{\lambda}_3 & \dots & {}^1\underline{\lambda}_{p-1} & {}^1\underline{\lambda}_p \\ & w_2^* & {}^2\underline{\lambda}_3 & \dots & {}^2\underline{\lambda}_{p-1} & {}^2\underline{\lambda}_p \\ & & w_3^* & \dots & {}^3\underline{\lambda}_{p-1} & {}^3\underline{\lambda}_p \\ & & & & \vdots & \vdots \\ & & & & w_{p-1}^* & {}^{p-1}\underline{\lambda}_p \\ & & & & & w_p^* \end{bmatrix} \quad (54)$$

where ${}^j\underline{\lambda}_k$ ($j < k$) denotes the components of the multipliers $\underline{\lambda}_k$ of the level k for the constraints of level j and the empty spaces for $j > k$ express the absence of multipliers on above levels.

The matrix Λ_p gives another way to look at the interactions between tasks, thanks to a classical result of perturbation and sensitivity analysis (see [Boyd and Vandenberghe, 2004], chapter 5.6): the bigger an element of ${}^j\underline{\lambda}_k$ is (in absolute value), the more a further violation of the corresponding constraint would decrease $\|w_k\|$. Thus, big elements in Λ_p indicate strong incompatibilities between tasks. On the contrary, for compatible tasks j, k, ${}^j\underline{\lambda}_k = 0$.

2.5.2 Dual computation

There is no direct formulation to compute the whole Λ_p. Alternatively, the multipliers of each level have to be computed iteratively. The solution (53) gives the matrix formulation of the Lagrange multipliers

of level k. To compute the multipliers, it is more efficient to avoid the explicit computation of the hierarchical transpose inverse. Using (47), (53) can be rewritten:

$$\underline{\lambda}_k = \begin{bmatrix} {}^1\underline{\lambda}_k \\ \vdots \\ {}^{k-1}\underline{\lambda}_k \end{bmatrix} = \begin{bmatrix} -\underline{A}_{k-2}^{\ddagger T}(A_k^T w_k^* + A_{k-1}^T {}^{k-1}\underline{\lambda}_k) \\ -U_{k-1}L_{k-1}^{-T}Y_{k-1}^T A_k^T w_k^* \end{bmatrix} \tag{55}$$

By recurrence, the components ${}^j\underline{\lambda}_k$ of the multipliers of level k can be computed starting from $j = k-1$ down to 1.

$${}^j\underline{\lambda}_k = -U_j L_j^{-T} Y_j^T \Big(\sum_{i=j+1}^{k-1} A_i^T \, {}^i\underline{\lambda}_k + A_k^T w_k^* \Big) \tag{56}$$

Using the HCOD structure of the $A_i^T \, {}^i\underline{\lambda}_k$, the sum on the right part can be computed for a reduced cost. The algorithm is given in Alg. 2. The cumulative variable ρ is used to propagate the recursion across the k levels. At the end of any iteration j, the following property is respected:

$$\underline{\rho}^{(j)} = \underline{Y}_j^T \Big(\sum_{i=j+1}^{k-1} A_i^T \, {}^i\underline{\lambda}_k + A_k^T w_k^* \Big) \tag{57}$$

In line #7, $\underline{\rho}^{(j+1)}$ is separated in two parts following the separation of $\underline{A}_{j+1} = \begin{bmatrix} \underline{A}_j \\ A_{j+1} \end{bmatrix}$. The first part of the vector is used to satisfy (57) while the second part gives ${}^j\underline{\lambda}_k$ using (56).

While the primal algorithm 1 computes the primal optima x and w for all the levels at the same time, the dual algorithm 2 can only achieve the computation of w and $\underline{\lambda}$ for one level at a time. Both algorithms can compute w naturally. If both are used, then a choice has to be made on where to really perform the computation of w^*.

2.6 Conclusion

We provide two algorithms to compute the primal and dual optima of a eHQP problem. Both are based on the HCOD. This decomposition is adapted to the hierarchical structure, which spare many computations that are necessary in the classical iterative methods used in robotics, such as [Siciliano and Slotine, 1991]. Based on the COD, the method does not produce any significant drawback, in particular in term of numerical robustness. Using the eHQP resolution, we now propose a dedicated hierarchical active-set algorithm.

Algorithm 2 Dual eHQP of level k

1: **function eHQP_dual**($\underline{A}_p$, $\underline{b}_p$, x^*, k)
2: **Input:** HCOD of $\underline{A}_p$, $\underline{b}_p$, Primal optimum x^*, level k
3: **Output:** w_k^* and $\underline{\lambda}_k$ satisfying (26) and (29)
4: $e = N_k y_k^* - V_k b_k$
5: $w_k^* = V_k e$
6: $\underline{\rho} = -N_k^T e$
7: **for** j=k-1 **downto** 1 **do**
8: $\quad \begin{bmatrix} \underline{\rho} \\ \rho \end{bmatrix} := \underline{\rho}$
9: $\quad r := L_j^{-T} \rho$
10: $\quad {}^j\underline{\lambda}_k := U_j r$
11: $\quad \underline{\rho} := \underline{\rho} - M_j^T r$
12: **end for**
13: **return** $w_k^*, \underline{\lambda}_k$

3 Inequality hierarchical quadratic program

In this section, we devise an algorithm for solving the hierarchical problem (22) subject to inequality constraints (iHQP). We derived this algorithm from the classical primal active set method for QP [Nocedal and Wright, 2006]. This choice for this class of methods is motivated by the need to use warm start in our robotic context of parametric problems (see next section), and is made available by the fact we have at hand, with the HCOD, a decomposition that is cheaply updatable, and thus perfectly adapted. The active sets of all the hierarchical levels are computed at the same time, so that we avoid the unnecessary iterations encountered in [Kanoun et al., 2011, De Lasa et al., 2010].

3.1 Definitions and preliminary remarks

For a given x, a constraint a, b, with a a row vector is *satisfied* if $ax \leq b$, *violated* if $ax > b$ and *saturated* if $ax = b$. Active set methods stem from the following remark: at the optimum of the optimization problem, some inequality constraints hold as equality constraints. Those are the constraints preventing to go closer to the unconstraint minimum. They are said to be *active* at the optimum. The other constraints (coined *inactive*) are irrelevant for determining the solution. Would we knew in advance the optimal active set $\mathcal{S}^*$, i.e. the indices of active constraints at the optimum, we would only need to solve the associated eQP obtained by selecting the constraints indexed by $\mathcal{S}^*$ to solve the iQP. An active set method works with a set $\mathcal{S}$ of active constraints. At each iteration, the corresponding eQP is solved, and depending on the result, the active set is modified by activating a violated constraint or deactivating one whose Lagrange multiplier is negative. See [Nocedal and Wright, 2006] for details.

Adaptating the method for iHQP is done through the following changes:

- using our eHQP solver instead of the eQP, obviously, to find the hierarchical optimum for a given active set,
- iterating on the number of levels to compute the Lagrange multipliers, since they cannot be computed for all the levels at the same time,
- taking advantage of the specific role of the w_i variables to simplify the computations.

For the two first points, we trivially adapt Alg. 1 and Alg. 2 to work only with the active constraints and denote them by $eHQP\_primal(\underline{A}_p, \underline{b}_p, \mathcal{S})$ and $eHQP\_dual(\underline{A}_p, \underline{b}_p, \mathcal{S})$. With the use of the eHQP solver we can process all the levels together and therefore only need one active-set, to the contrary of methods based on a sequence of QP [Kanoun et al., 2011, De Lasa et al., 2010] which use an active set for each QP.

The third point requires more explanations and is based on two observations: (i) at the optimum, the components of the w_k corresponding to the inactive constraints (we name them *inactive slack variables* or *inactive slacks* for short) are equal to 0, and (ii) when solving the eHQP for a given active set, only the value of the active slacks are relevant. At an iteration i of our active search, we can thus take $w_{k,r}^{(i)} = A_{k,r}x^{(i)} - b_{k,r}$ if the r^{th} constraint of level k is active, where $X_{k,r}$ refers to the r^{th} row of X_k, and $w_{k,r}^{(i)} = 0$ otherwise. At each iteration, the w_k are thus completely determined by $x^{(i)}$ and the current active set, therefore we do not need to keep track of them. In particular, we can compute the step lengths by taking only x into accounts, and perform the steps only on x.

With this choice of w_k, some constraints might be inactive and violated at first. Our algorithm readily detects these constraints and activates them.

3.2 Hierarchical active search

Our hierarchical active search algorithm, *HQP*, is summarized in Alg. 3 and detailed below.

3.2.1 Algorithm organization

The proposed algorithm is composed of two loops: an inner loop that first enforces then maintains the property that *all the constraints should be activated or satisfied*. And an outer loop that explores all the levels in ascending order to search for the corresponding optimal active set. The inner loop itself is composed of two sets of instructions: the first one (lines #7 to #16) concerns the activation of needed constraints, while the second one (lines #17 to #26) deals with the deactivation to obtain the optimal active set.

3.2.2 Initialization

The algorithm starts with an initial guess $\mathcal{S}^{(0)}$ of the active set of all the levels. It does not need an initial parameter $x^{(0)}$ since none of the levels (even the first one) is guaranteed to be feasible. It then starts with the arbitrary value $x^{(0)} = 0$.

3.2.3 Step length and activation

The active-search then maintains a value of the parameter $x^{(i)}$ and the active set $\mathcal{S}^{(i)}$. At each inner iteration, the algorithm first computes the optimum $x^{(*i)}$ of eHQP associated to $\mathcal{S}^{(i)}$. The current parameter is then moved toward $x^{(*i)}$:

$$x^{(i+1)} = x^{(i)} + \tau(x^{(*i)} - x^{(i)}) \tag{58}$$

where τ is the biggest fraction of the step that can be taken without violating any satisfied constraints. It is the minimum of the step allowed by each individual constraint:

$$\tau = \min\left(\min_{k,r} \tau_{k,r}, 1\right) \tag{59}$$

$$\text{with } \tau_{k,r} = \begin{cases} \frac{b_{k,r} - A_{k,r}x^{(i)}}{A_{k,r}(x^{(*i)} - x^{(i)})} & \text{if } A_{k,r}x^{(i)} \leq b_{k,r} \\ 1 & \text{otherwise} \end{cases}$$

The constraint with the smallest $\tau_{k,r} \leq 1$ is saturated by the step (58) and is activated. If several constraints correspond to the minimum τ, only one of them should be arbitrarily activated.

Depending on the initial guess $\mathcal{S}^{(0)}$ and the corresponding eHQP optimum, some of the inactive constraints may be violated. In that case, the activation loop will eventually make some full steps $\tau = 1$, while each time adding one violated constraint into the active set. The number of activation steps is bounded by the number of rows of $\underline{A}_p$. At the end of these steps, *all the constraints should be activated or satisfied*. This property is then maintained throughout all the following iterations.

3.2.4 Positivity of the multipliers and deactivation

Eventually, a full step ($\tau = 1$), possibly trivial ($x^{(*i)} = x^{(i)}$), will be taken without activating any constraint. The Lagrange multipliers for the current level k are then computed (it was not necessary to test them before). The active set is optimal with respect to the current level if no multiplier component is strictly negative. Otherwise, the constraint corresponding to the lowest component is deactivated, and a new inner iteration is started.

If the active set is optimal for the current level k we can distinguish between the *strongly active* constraints for which the corresponding Lagrange multipliers are strictly positive and the *weakly active* constraints with a multiplier equal to 0. As observed in [Kanoun et al., 2011], strongly active constraints cannot be deactivated at a next level. To enforce this, the strongly active constraints are *locked*, as in [Kanoun et al., 2011].

In summary, the outer loop explores each level starting from the first one. At each level, it computes the multipliers. If a constraint is strictly negative and does not correspond to a strictly positive component of the multipliers of the previous levels, it is deactivated. When no more constraint needs to be deactivated, the constraints corresponding to strictly positive components of the multipliers are stored in the set $\mathcal{F}$ of locked constraints.

3.2.5 Algorithm termination and proof of convergence

The algorithm stops after p outer iterations have been completed. Each outer iteration k finishes when there are no more constraint to activate or deactivate, which induces that $\|w_k\|$ is optimal and will not be changed anymore. Upon termination, the active set is such that all the constraints are active or satisfied, and it is optimal for each of the levels.

The termination of the whole algorithm is ensured by the fact that each outer loop k terminates, as proved in App. B.2.

3.3 Lexicographic optimization

The algorithm deactivates a constraint if there exists a level k for which the component of the multiplier corresponding to the constraint is negative, while it is zero for all the multipliers of level $j < k$. Consider the matrix Λ_p in (54). A constraint is deactivated *iff* the corresponding row of Λ_p has one strictly negative component on column k preceded by only zeros for the columns $j < k$. In other words, the row is smaller than zero in the lexicographic sense:

$$\begin{bmatrix} 0 & \dots & 0 & -\alpha & \times & \times & \dots & \times \end{bmatrix} \prec 0 \tag{60}$$

Algorithm 3 Hierarchical active search

1: **Input:** Initial guess $\mathcal{S}^{(0)}$
2: **Output:** x^* minimizing (22)
3: $x = 0$; $\mathcal{S} = \mathcal{S}^{(0)}$
4: $\mathcal{F} = \emptyset$
5: **for** $k = 1 : p$ **do**
6: **repeat**
7: −−Compute the next optimum−−
8: $x^* =$ eHQP_primal $(\underline{A}_p, \underline{b}_p, \mathcal{S})$
9: −−Compute the step length using (59)−−
10: τ,activate,cst = step_length$(\underline{A}_p, \underline{b}_p, x, x^*)$
11: $x := x + \tau(x^* - x)$
12: −−If necessary, increase the active set−−
13: **if** activate **then**
14: $\mathcal{S} := \mathcal{S} \bigcup \{cst\}$
15: **continue**
16: **end if**
17: −−If necessary, decrease the active set−−
18: $w, \lambda =$ eHQP_dual $(\underline{A}_p, \underline{b}_p, x, k, \mathcal{S})$
19: $\lambda_{\mathcal{F}} := 0$
20: $\nu, cst = \min\{\lambda, w\}$
21: **if** $\nu < 0$ **then**
22: $\mathcal{S} := \mathcal{S} \setminus \{cst\}$
23: **continue**
24: **else**
25: $\mathcal{F} := \mathcal{S} \bigcup \{cst, \lambda_{cst} > 0\} \bigcup \{cst, w_{cst} > 0\}$
26: **break**
27: **end if**
28: **until not** activate **and** $\nu > 0$
29: **end for**
30: **return** $x^* := x$

With this notation, Alg. 3 can be rewritten as a classical active set search, using a lexicographic test on Λ_p instead on lines #18-#19. If the lexicographical formulation is simpler, the more explicit Alg. 3 remains more efficient from a computational point of view since it avoids computing the whole Λ_p at each iteration.

3.4 Least-norm solution

The eHQP algorithm gives the least-square solution among all the solutions of same cost, when z_{p+1} is set to 0. However, this is not the case of the iHQP since some constraints might be uselessly active. Indeed, there is no mechanism to deactivate the weakly active constraints. To force their deactivation, an artificial last level can be added by setting $A_{p+1} = I$ and $b_{p+1} = 0$ that is to say $x = 0$. This constraint will always be rank deficient, but its satisfaction at best in the least-square sense ensures that the optimal active set is unique and the the returned optimum is the least-square one.

3.5 Implementation details and complexity

We note $\underline{A}_p(\mathcal{S})$ the matrix whose lines are the lines of $\underline{A}_p$ corresponding to active constraints. The costlier operation in the algorithm is the computation of the HCOD of $\underline{A}_p(\mathcal{S})$ in *eHQP_primal*, which is in $O(n^3)$. All other operations (computation of x^*, step (58), computation of $(\underline{\lambda}_k, w_k)$ for one level) are then performed in $O(n^2)$. Computing this decomposition from scratch at every inner loop is therefore out of question. Fortunately, the decomposition being based on the COD, rank-1 updates can be done cheaply: the addition or removal of a line at any position in $\underline{A}_p(\mathcal{S})$, corresponding to a the activation or deactivation of a constraint implies a change in the decomposition that is done in $O(n^2)$. The correct way to implement the algorithm is therefore to compute only once the HCOD, at the first iteration, then to update it at each change of the active set. Details of the update process and complexity costs are given in [Escande et al., 2013].

As any numerical scheme, the algorithm needs the settings of tolerances, two in our case, for rank determination and constraint violation. We set both to square root of the machine precision.

A complete and well-documented implementation of the active-set solver is provided as an example using the MATLAB language in the attached documents (see Appendix A). A more efficient implementation using C++ is pro-

vided in the open-source project StackOfTasks `http://github.com/stack)of)tasks/soth`.

3.6 Conclusion

Based on the hierarchical decomposition, we have proposed an active-search algorithm that solves the iHQP problem. On the contrary to the cascade of QP used in [Kanoun et al., 2011, De Lasa et al., 2010], this algorithm computes the active set of all the level of the hierarchy at the same time. It thus solves the complete hierarchical problem at once, avoiding the back-and-forth effects encountered with cascades and sparing the cost of computation.

4 Parametric optimization

The major recognized interest of hierarchical optimization in robotics is in control, for IK and ID. In these contexts, the problem definition (matrices A and vectors b) only varies slightly from one control cycle to the other. We can then use the solution computed at a given cycle to reduce the complexity of the search at the next cycle. This is known as warm-starting in optimization. For that we first need to show that the optimum continuously evolves with the problem definition, by defining a parametric hierarchical problem.

4.1 Parametric problem definition

We consider that HQP, that varies continuously with respect to a given parameter t:

$$\underset{x,w_1 \dots w_p}{\operatorname{lex\,min}} \; \{\|w_1\|, \|w_2\|, \dots, \|w_p\|\}. \tag{61}$$

$$\text{subject to} \quad \forall k = 1:p, \; A_k(t)x \leq b_k(t) + w_j$$

with $A_k(t)$ and $b_k(t)$ continuous function of a real parameter $t \in \mathbb{R}$. This problem is denoted by HQP(t). We study the continuity of the two functions:

$$\mathcal{O} : t \to \mathcal{O}(t) = \{x, \forall x', \underline{A}_p x \preccurlyeq \underline{A}_p x'\} \tag{62}$$

$$x^* : t \to x^*(t) = \min_{x \in \mathcal{O}(t)} \{\|x\|\} \tag{63}$$

where $b_1 \preccurlyeq b_2$ denotes the lexicographic order corresponding to the hierarchy. $\mathcal{O}(t)$ is the optimal set of the HQP(t) and $x^*(t)$ is the least-norm element of $\mathcal{O}(t)$ (unique since $\mathcal{O}(t)$ is convex).

4.2 Continuity of the parametric optimum

For a given t, we note $\mathcal{S}^*(t)$ the optimal active set of HQP(t) corresponding to $x^*(t)$ with no weakly active constraint. As t evolves, some constraints are added or removed from $\mathcal{S}^*(t)$. Since there is a finite number of constraints, this happens at discrete instants, between which $\mathcal{S}^*(t)$ is constant. In fact $\mathcal{S}^*(t)$ is constant almost everywhere.

On each interval where $\mathcal{S}^*(t)$ is constant, HQP(t) is equivalent to the much studied equality-constraints only hierarchical problem, and therefore its solution $x^*(t)$ is continuous everywhere but when passing through a singularity [Ben-Israel and Greville, 2003]. Note that the discontinuity only occurs at the instant the singularity appears or disappears (instants of rank change of an $A_k Z_{k-1}$). We name such an instant *singularity instant*.

We are left with the study of continuity at the instants of active-set changes. Denoting by $x^*_{\mathcal{S}}(t)$ the least-norm optimum of the eHQP associated to the active set $\mathcal{S}$, the following theorem ensures the continuity of the solution everywhere but at singularity instants:

Theorem 4.1. *At a given t_0, if $t \to x^*_{\mathcal{S}^*_{t_0}}(t)$ is continuous, then $x^*(t)$ is continuous.*

Its proof only needs to consider instants of active-set changes and is formally given in [Escande et al., 2013]. It is based on the fact that if such an instant t_0 is not also a singularity instant, a constraint activated or deactivated at t_0 is weakly active and thus does not disturb the optimum at t_0 whether it is active or not. The optimum being continuous before and after t_0, the continuity is obtained.

The only sources of discontinuity are therefore the rank change in $A_k Z_{k-1}$, as in the equality-constraint only case.

This continuity result is inherent to the problem (61) and fully independent of the way to solve it. We now show how to take advantage of it with our solver.

4.3 Warm start and real-time implementation

On-board a robot, the solver is used at discretized times t,t+1,... Thanks to the continuity, the optimum of time t nearly satisfies the feasible constraints of time $t+1$. A good initial guess for the active set of time $t+1$ is thus the optimal active set of time t. In most of the cases, this is enough to reach the optimum in a single iteration. From time to time, the active set changes: in most cases few additional activations and deactivations are sufficient. However, one cannot guarantee the number of necessary iterations to reach the optimum. In particular, a minor modification of one optimum can trigger a cascade of activations-deactivations in pathological cases.

The warm start improves the efficiency of the solver. However, we cannot yet guarantee that the solver answers in a bounded number of iterations. It is however possible to force the solver to stop after an arbitrary number N of iterations, which enforces the real-time property. Thanks to the continuity property, the obtained final point $x^{(N)}$ is at a distance to the optimum x^* proportional to the sampling of the control, which guarantee a good behavior in practice. If the optimal active set was not reached at $t+1$, the search will restart at $t+2$ from the intermediary active set obtained after the N iterations of time $t+1$.

By using the previous active set as a warm start and bounding the number of iterations, the solver will find the optimum within the bounded number of iterations in most of the case, and otherwise will respect the real-time constraint by spreading the search of the optimal active set over several iterations.

The fast and real-time properties are obtained thanks to the continuity of the optimum and to the monolithic structure of the active-set algorithm, that consider only one active set candidate for all the levels simultaneously. On the opposite, it would not possible to adapt the warm start to a cascade resolution like in [Kanoun et al., 2011] and [De Lasa et al., 2010].

5 Hierarchical inverse kinematics

We first discuss quickly a possible use of the HQP solver. Then we expose the robotics setup that has been used in the next section, implementing IK for a humanoid robot. An important aspect is to prove the stability of the controller. The theoretical result is expressed in Sec. 5.2 and the main lines of the proof are given. The IK scheme is then used in the next section to validate the interest of our approach.

As explained above, the hierarchical formulation has been used in robotics for IK and ID. IK is straight-forward, since the solver variable is the robot velocity. In ID, the main variables are the joint torques but the robot acceleration and the external contact forces have also to be considered [Collette et al., 2007]. On a robotics point of view, IK is more limited (many constraints of the robot such as actuator limits or humanoid balance) cannot be expressed properly. However, it is also more widely used and easier to understand. To limit the context of this paper, we focus on IK here. Hierarchical ID with the same solver is discussed in [Saab et al., 2013].

5.1 Robotic setup

The solver has been tested to perform an IK with the HRP-2 robot, in simulation and on the real platform. The robot has 36 DOF whose first six are not actuated. When considering only the kinematics, a classical solution is to replace the underactuation by a contact constraint of equivalent size, typically constraining the six parameters of position and orientation of one contacting foot. The underactuation is then resolved while the dynamic of the robot is negligible. All the HQP solved in the following have a parameter size $n = 36$. The robot is controlled at 200Hz.

Two types of tasks are considered. The first type aims at regulating to 0 an error depending on the

robot configuration. The task function is then given by:

$$e(q, \Omega) = s(q) - s^*(\Omega) \tag{64}$$

where the task function e depends on the robot configuration and other parameters of the universe Ω as an error between a current measurement $s(q)$ and a desired value of it s^*. The functions used in the following are the end-effectors position and orientation of the right hand e_{rh}, left hand e_{lh} and feet e_{rf}, e_{lf}. Both feet are controlled together on the ground by stacking the two last tasks $e_{feet} = (e_{rf}, e_{lf})$. The center of mass (COM) is controlled to a given point by e_{com}. The orientation of one vector attached to the robot (for example, having the hand around a stick but able to rotate around the stick axis) is controlled by regulating the cross product:

$$e_\theta = u(q) \times u^*(\Omega) \tag{65}$$

where u is the vector attached to the robot to be placed on u^*. More details about these classical task functions can be found in [Sentis, 2007, Kanoun, 2009]. The regulation of the error is realized by a proportional correction $\dot{e}^* = -\kappa e$, $\kappa > 0$. The linear constraint is finally:

$$J\dot{q} = \dot{e}^* - \frac{\partial e}{\partial \Omega}\dot{\Omega} \tag{66}$$

where $J = \frac{\partial e}{\partial q}$ is the robot task Jacobian.

The second type of tasks defines a bound on a function of the robot configuration:

$$e^l(\Omega) \leq e(q) \leq e^u(\Omega) \tag{67}$$

This constraint is homogeneous to the configuration. The linear constraint is obtained by the first-order Taylor approximation:

$$\frac{\kappa}{\Delta t}(e^l(\Omega) - e(q)) \leq J\dot{q} \leq \frac{\kappa}{\Delta t}(e^u(\Omega) - e(q)) \tag{68}$$

where κ is the time horizon (in number of control iterations) used as a gain modifier [Faverjon and Tournassoud, 1987].

Several task functions e can be used to obtain various behaviors. The joint-limit task e_{jl} bounds the robot configuration q by a set of fixed value. The task e_{supp} keeps the COM in the support polygon. The field-of-view (FOV) task e_{fov} considers the projection of an object on the image plane and bounds it by the image border. The collision avoidance is enforced by the task e_{coll} by imposing the distance between a body of the robot and an object to be positive. In that case, the pair of bodies and object to check has to be specified (no systematic collision checking was performed here, it should be considered in the future [Stasse et al., 2008]). Once more, details about these classical functions can be found in [Sentis, 2007, Kanoun, 2009].

By construction, the IK problem, with or without priority, is subject to singularity. In the neighborhood of a singular point, the diagonals of the L_k matrices become low (as explained in Sec. 2.3), which raises undesirable high values during their inversion. This is the same problem encountered in any other IK scheme and our solver is not more sensible to this. Three solutions can be considered. It is possible to regularize the problem, for example by adapting the classical Tikhonov regularization classically used as a damping term for each level. Or a coercive bound can be enforced on the robot velocity, or even better, on the acceleration. This can be seen as a regularization of the dual. Finally, during the execution, the HCOD can be used to diagnose the near-singular tasks and remove them.

5.2 Stability

More abstractly, the HQP constraints can be written by a set of p tasks:

$$\dot{e}_k = J_k\dot{q} \tag{69}$$

$$\forall k \in S_I,\ \dot{e}_k \leq b_k \tag{70}$$

$$\forall k \in S_E,\ \dot{e}_k = \dot{e}_k^* \tag{71}$$

where $S_I \bigcup S_E$ is a partition of the set $\{1 \dots p\}$ of the p first integers: S_I are the task levels that are defined by a limit b_k and S_E are the task constraints to follow a given velocity $\dot{e}_k^*$. We denote by e_E the stack of all equality constraints and by J_E the associated jacobian.

We suppose that all the equality tasks are stable (*i.e.* $e_k\dot{e}_k^* < 0$) and that 0 is an acceptable solution

for all the tasks of S_I, *i.e.* $\forall k \in S_i,\ b_k > 0$. In this section, we briefly prove that the use of a HQP keeps the same properties of control stability than in other classical IK approaches.

Theorem 5.1. *The hierarchical IK control law is stable in the sense of Lyapunov. It is asymptotically stable* iff J_E *is full row rank and none of the equality-constraint levels of the HCOD are rank deficient.*

It is possible to demonstrate that the control scheme described here is stable, based on the same kind of reasoning as with the continuity. The proof is included in [Escande et al., 2013]. The idea is the following: if the attractor region is inside the polytope defined by the inequality constraints, then the control is stable using the well-known results of the task-function approach [Samson et al., 1991]. If it is outside, then the inequality constraints limit the motion (and prevent asymptotical stability) but do not destabilize it.

6 Results: simulation and experiments

We first present in Sec. 6.1 a set of timing results obtained with random problems out of any robotics application to compare our solver with previous approaches. The IK scheme presented in the previous section is then used to generate three movements in simulation and on the real robot HRP-2. In each case, our solver is compared to the solvers proposed in [Kanoun et al., 2011] and [De Lasa et al., 2010].

The computations have been performed on a 2.3GHz Intel Core 7 CPU (same CPU specification for the simulation and for the robot computer). It was not possible to try all the movement on the robot since some of the resolution methods are too slow and cannot offer real-time guarantee. The first movements are then shown in simulation only. The last movement is performed on the real robot. An overview of the solver performances is given in Table 1.

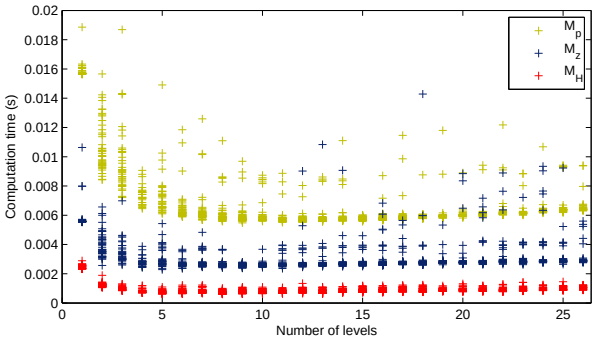


Fig. 2: *Computation time for the eHQP plotted with respect to the number of level p. The problems are randomly selected with same dimension ($n = 100$, $m = 150$, $r = 80$). The HCOD is approximately ten times faster than the classical iteration projection proposed in [Siciliano and Slotine, 1991].*

6.1 Computation time

First, we quickly compare the efficiency of our solver on generic problems. The problem definitions in this subsection (matrix $\underline{A}$ and vector $\underline{b}$) are randomly shot with the same dimension and rank. However, the problem rows are distributed into an arbitrary number of priority levels. We measure the algorithm efficiency with respect to the number of levels, first without then with inequalities.

6.1.1 Equality-only HQP

We first experimentally check the computation time needed to compute the primal optimum of a eHQP using three methods: the classical iterative-projection proposed in [Siciliano and Slotine, 1991] and recalled in (10) ($\mathcal{M}_P$ for short); the implicit projections using successive basis, proposed in [Escande et al., 2010] and recalled in (12) ($\mathcal{M}_Z$ for short); and using the HCOD proposed in this paper ($\mathcal{M}_H$ for short).

We randomly selected 1000 hierarchical problems with same dimension ($n = 100$, $m = 150$) and same total rank ($r = \sum r_i = 80$) but with number of levels and distribution among the levels varying from $p = 1$ to $p = 26$. The measured computation times are

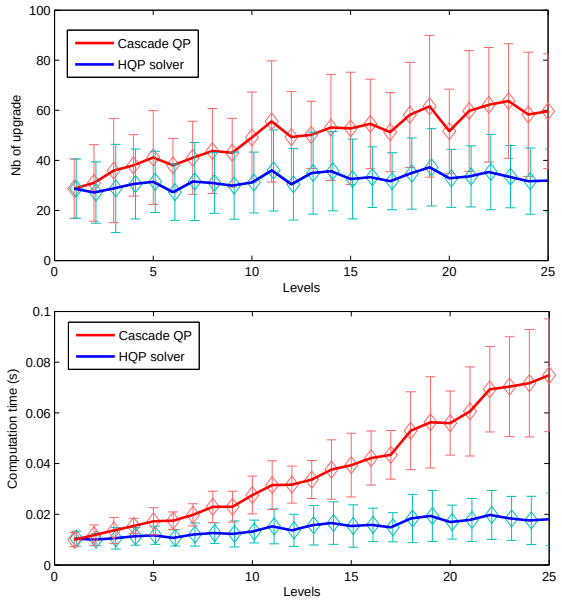


Fig. 3: *Number of active-set iterations and computation time for the iHQP resolution using a cascade of QP. Both costs in time and in number of algorithm iterations increase with p for the cascade of QP and remain constant for the HQP solver.*

presented in Fig. 2.

The HCOD implies much faster computation (approximately ten times faster than $\mathcal{M}_P$ which is the most classically used; typically less than 1ms is needed for this size of problem). It is also faster than $\mathcal{M}_Z$ since only one basis is used, which enables the processor to optimize better repetitive product operations.

Interestingly, the cost increases for small p with all the methods. This is due to the time spent in handling trade off in conflicts, that the hierarchy just avoids by definition. With more levels, the conflicts are more likely to be divided into several priority levels, which decreases the computation cost.

6.1.2 Active-set search

Similarly, the hierarchical active search Alg. 3 is compared to the cascade of QP used in [Kanoun et al., 2011][2]. The results are shown in Fig. 3. As described in Section 1.6, the number of cycle in the active-set search increases with the number p of levels when using a cascade of QP. It remains constant with the HQP. The measured computation time follows the same evolution. Typically for 6-8 levels (which is the number of levels that will be used in the following), the cascade takes twice the time needed by the HQP to converge.

6.2 Simulation A: grasping using conditional visual guidance

6.2.1 Setup

The robot has to grasp a point object while keeping its balance and respecting its joint limits. During the task, the robot tries to keep the object in its FOV and, if possible, to keep its COM into a small 2cm-wide band inside its support polygon to obtain a well-balanced posture (e_{bal}). The task order is then $e_{jl} \prec e_{feet} \prec e_{supp} \prec e_{rh} \prec e_{fov} \prec e_{bal}$. The robot motion is summarized by Figures 4 to 10. The obtained motion as well as the two motions obtained in the next two sections is recorded in the attached video (see App. A).

The ball is moved in four different places as shown in Fig. 4: first in front of the robot, in an easily reachable position; then far in the left, so that the robot has to bend to reach it; the ball is put back to the initial position before putting it to a far right position that is not reachable while keeping the COM inside the support polygon. At the first ball position, the two last tasks are respected: the ball is inside the FOV and the COM is in the central band. The second ball position is further away and requires to the robot to importantly bend to reach it. The COM cannot be kept inside the central band while satisfying all the other tasks. Relaxing the least-priority COM task enables to satisfy e_{fov} and e_{rh}. The higher-priority COM task is respected, that ensures the robot balance. The ball is then placed back in front of the robot: the COM comes back to the

[2] We cannot compare the HQP with [De Lasa et al., 2010] for this setup because this last method does not handle the problems tested here. A comparison is proposed in Sec. 6.3.

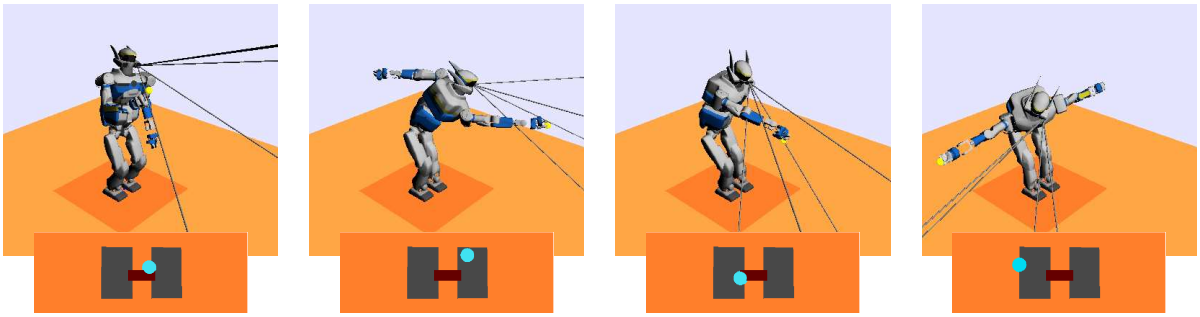


Fig. 4: *Top row: snapshots of the robot motion. Bottom row: corresponding COM projection in the support polygon (view from underside, the COM position is depicted by the small disk, each rectangle depict one of the foot, the central rectangle shows the area for e_{bal}). Each snapshot is captured at the end of a motion sequence. The FOV is displayed by the 4 lines passing by the center of the image projection.*

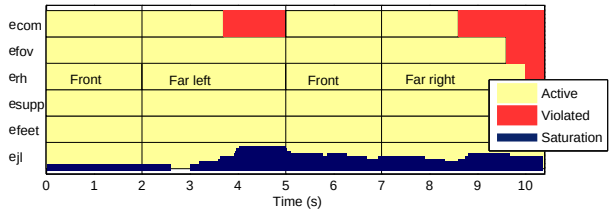


Fig. 5: *Simulation A: task sequence, listed by priority from bottom to top. The tasks are specifically marked when they become violated. The hierarchy appears through the violation order: the least-priority tasks are relaxed first in case of conflicts. The number of saturated joint limits is displayed in the e_{jl} row.*

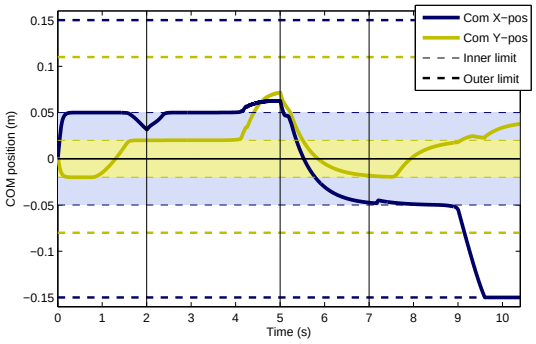


Fig. 6: *Simulation A: position of the COM wrt the inner and outer limits. The COM has to remain into the outer limits to ensure the balance of the robot, and should be kept if possible inside the central band (inner limits) to obtain a balanced robot posture.*

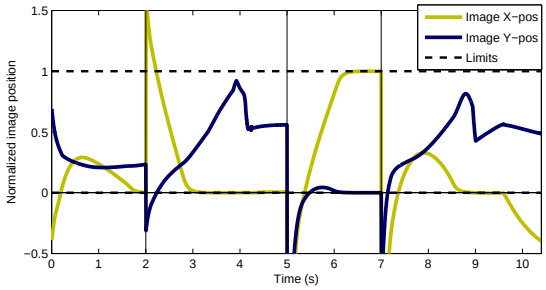


Fig. 7: *Simulation A: position of the object projection in the image plane wrt the FOV limits. When the ball is moved outside the FOV, e_{fov} brings it back into the FOV limits. At $T = 0s, 2s, 5s$ and $7s$, the ball is artificially moved out of the FOV and the robot brings it back following the task reference. The robot loses the ball at $T = 9.8s$ due to a conflict with the tasks having priority.*

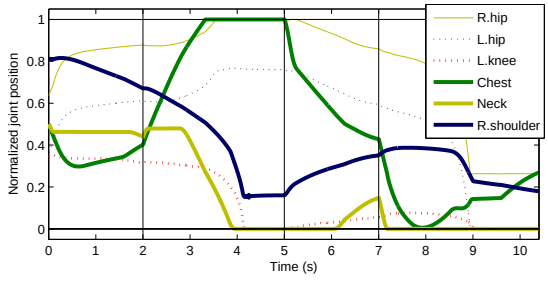


Fig. 8: *Simulation A: normalized joint positions.*

central band while all the other tasks are kept satisfied. The last position is unreachable while keeping the COM in the support polygon. All the tasks from the least-priority one are successively relaxed until the minimal distance to the ball is finally reached: at the final position, the COM is outside of the central band, on the border of the support polygon, and the ball is outside the FOV. This is a typical case of interest of the hierarchy: a proper behavior is ensured by the tasks having priority (balance, joint limits) while the optional objectives are satisfied at best.

6.2.2 Results

The task sequence is given in Fig. 5. In the beginning of each motion sequence (when the ball is just moved), the visibility constraint (67) might be violated without the FOV task (68) being violated: the control is simply bringing the ball inside the FOV boundaries according to the task definition. The task becomes violated when (68) cannot be fulfilled. Details about the COM and FOV satisfaction are given in Figures 6 and 7. At the beginning of the third motion sequence, the COM is outside of the central band. It is brought back to this zone after 0.2 seconds. Similarly, at the beginning of each sequence, the ball is outside of the FOV and is quickly brought back. At time $T = 4s$, the COM is at the central-band limit when several joints reach their limits (see Fig. 8). This reduces the available DOF for the grasp task, and then for e_{bal}, which has to be relaxed: the COM leaves the central band. Similarly, at $T = 9s$, the COM is on the border of the band. The activation of some joint limits once more drives the COM outside of the central band. At $T = 9.5s$, some DOF of the grasp task e_{rh} collapse because of a kinematic singularity. The freed DOF can then be used by the least-priority tasks and this leads first the X coordinate of the COM to leave the central band, then the ball to leave the FOV. The COM quickly escapes the central band, until it finally reaches the second COM bound imposed by e_{supp}. The limitation of the COM causes the violation of e_{rh}: the robot then stops as close as possible to the ball. Some typical trajectories of the joints are shown in Fig. 8: the limits are always respected. The number of active constraints for all levels together (*i.e.* the size of the optimal active set) is displayed in Fig. 9.

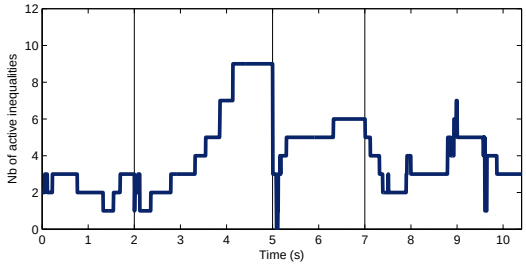


Fig. 9: *Simulation A: number of active inequalities at each control cycle.*

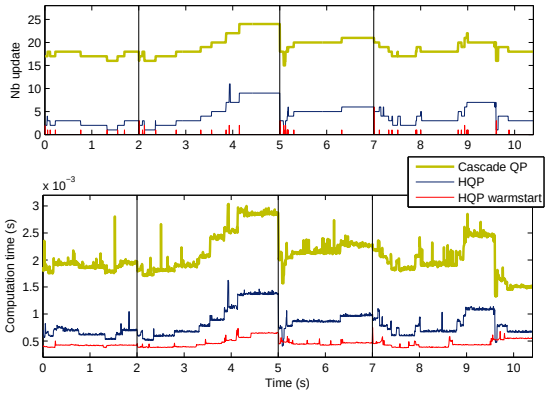


Fig. 10: *Simulation A: Number of algorithm iterations and computation time when using a cascade of QP [Kanoun et al., 2011] and using the HQP without and with warm start.*

6.2.3 Computation cost

We compare in Fig. 10 the cascade resolution proposed in [Kanoun et al., 2011] with our method using an empty initial guess and using a warm start as proposed in Sec. 4.3. First, the number of iterations in the active search loop is much higher with a cascade of QP than using the proposed HQP solver. The number of iterations in the active search is very similar to the number of active constraints at the op-

timum, as can be seen by comparing Fig. 9 to Fig. 10-(top). Our solver barely needs any deactivation to find the optimal active set. This is not the case when using a cascade of QP, which needs more than twice as many iterations to reach the optimal active set. As expected, the number of iterations is even lower using a proper warm start: in that case, the active search only iterates when a new boundary is reached. The maximal number of iterations is 6 (at the first iteration after the change of the ball position at T=7), the mean number is 0.03 and in 97.6% of the case, the active search converges without any update. As shown in Sec. 3.5, the computation time depends on the number of active constraints and of active-search iterations. Since there is nearly no iteration, the time with warm start in Fig. 10 depends only on Fig. 9 and has the same shape. For the two other, the influence of the number of iterations is more important, and the time graph shape is similar to the graph of number of iterations.

Using the HQP and the warm start, an average of 0.56ms of computation is needed. Our algorithm is three times faster than a cascade of QP and is 5-6 times faster is using a warm start. Moreover, the numerical behavior is improved by limiting the number of iteration in the search loop.

6.3 Simulation B: opening a valve

The previous movement cannot be generated using the method presented in [De Lasa et al., 2010] since inequality tasks were considered without the main priority. We now consider a less complex example that this method can handle, with all the inequality constraints feasible and at the top priority. The robot has to open a valve by manipulating a wheel. The motion is composed of two parts: the robot first manipulates the wheel using one hand, then rotates the wheel using both hands with successive re-grasps. During the motion, the robot has to avoid a block located on its left, and to keep its COM inside the support polygon. When grasping the wheel, the robot has to look at it; when rotating the wheel, it has to look at a manometer located on its left. The first movement (left-arm manipulation) is summarized in Figures 11 to 14. The second movement (both-arm manipulation) is summarized in Figures 15 to 20. The comparison of the computation times for both movements is given in Fig. 21.

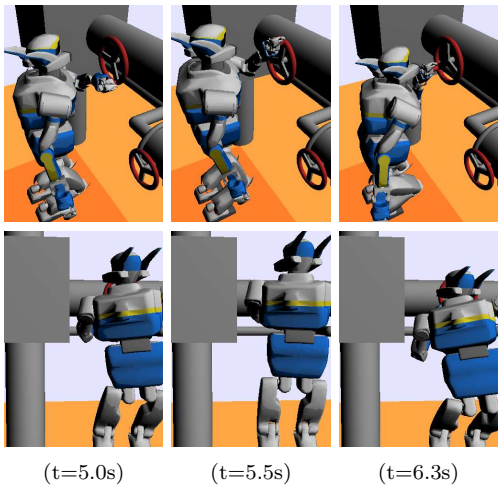


Fig. 11: *Simulation B-1: Snapshots of the first movement: the robot uses only its left hand to manipulate the wheel. The three snapshots are captured during the wheel rotation for three angles of* 0, $\frac{2\pi}{3}$ *and* $\frac{4\pi}{3}$.

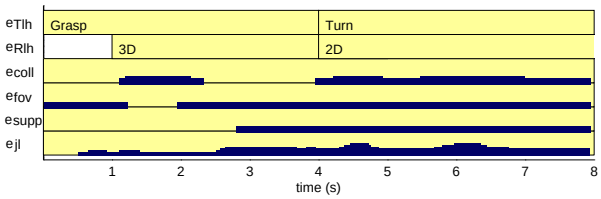


Fig. 12: *Simulation B-1: Task sequences of the first movement. The wheel is rotated of two complete loops. For the four inequality tasks, the number of active constraints during the motion is plotted for each level.*

6.3.1 First movement

Snapshots of the first motion are shown in Fig. 11. The task sequence is detailed in Fig. 12. The constraints are the joint limits, the support polygon, the FOV and the distance of the left elbow and shoulder to the left obstacle. The left-hand task is di-

vided into the translation e_{Tlh} and rotation e_{Rlh} components. During the approach, both the left-hand rotation and translation are controlled. When the robot rotates the wheel, the rotation of the hand around the wheel axis is let free and only two degrees of rotation are controlled. The hierarchy is $e_{jl} \prec e_{supp} \prec e_{fov} \prec e_{coll} \prec e_{feet} \prec e_{Tlh} \prec e_{Rlh}$. The four first tasks are compatible during all the movement and can be considered to be of equal priority (which makes the method of [De Lasa et al., 2010] possible to handle them).

The number of active constraints for the four first levels is shown in Fig. 12. The total number of active inequalities is given in Fig. 13. When the robot is on the left side of the wheel, the obstacle strongly constraints it, which raises the number of active constraints: two peaks appear for each loop of the wheel. The distances of the shoulder and elbow to the obstacle are given in Fig. 14. The arm comes close to collision when the robot approaches the wheel: the constraints are saturated to prevent it. The constraints are then deactivated when the robot goes away from the obstacle. When the robot starts to rotate the wheel, the constraints become once more active. Since the motion is not holonomic [Zanchettin and Rocco, 2012] (in particular, there is no posture task), the motion realized for each loop is different: during the second loop, the elbow constraint remains saturated.

6.3.2 Second movement

The robot then uses its second arm to ease the manipulation of the wheel. The motion is more constrained since both hands are bound to the wheel. Snapshots of the motion are given in Fig. 15. The task sequence

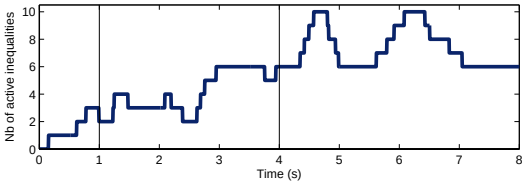


Fig. 13: *Simulation B-1: Number of active inequalities during the first movement.*

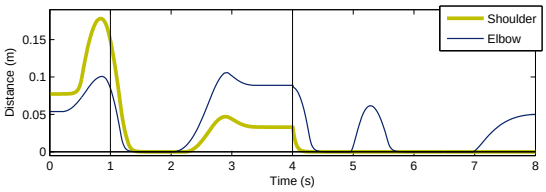


Fig. 14: *Simulation B-1: distance to the obstacle during the first movement. The obstacle is on the way between the left-arm initial position and the wheel: the two constraints of the shoulder and the elbow become active at $T = 1.1s$ and $T = 1.2s$. They are deactivated later during the grasping phase, at $T = 2s$ and $T = 2.1s$. At each loop of the wheel, the left arm comes close to the obstacle, between $T = 4s$ and $T = 5s$, and a second time between $T = 5.7s$ and $T = 7s$.*

is given in Fig. 16: each hand first reaches an arbitrary pre-grasp position before grasping the wheel. During the approach, the three rotations are controlled, while the rotation along the tangential axis to the wheel is left free during the manipulation. The distance to the obstacle is plotted in Fig. 17. The constraint becomes active at the end of the motion. The joint position with respect to the limit is shown in Fig. 18. Contrary to the previous simulation, the joints do not systematically remain on the exact limits since the robot is moving to follow the rotation of the wheel. The number of active constraints is given in Fig. 19. The number of active constraints increases with the complexity of the task. It reaches its maximum just before the robot starts to move the wheel (see Fig. 15-left, the robot is bent on the right with many apparent saturations). It then decreases when the robot stands straight (see Fig. 15-middle), and increases again at the end of the motion (see Fig. 15-right). Finally, the conditioning number of the left- and right-hand tasks is shown in Fig. 20. The conditioning number evolves both continuously with the changes in the Jacobians and discretely at each constraint activation.

6.3.3 Computation times

The computation time and corresponding number of iterations of the active set are plotted in Fig. 21. Using a warm start of the HQP, the active search loop converges without any update in 97.5% of the

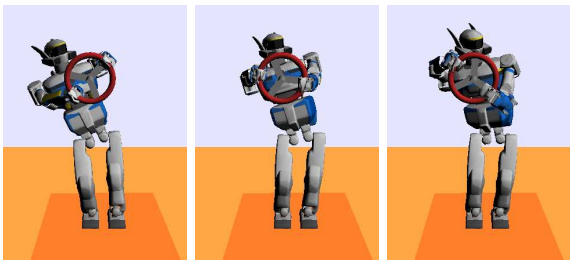


Fig. 15: *Simulation B-2: Snapshots of the second movement: the robot uses both hands to manipulate the wheel. For the sake of clarity, the environment is not displayed.*

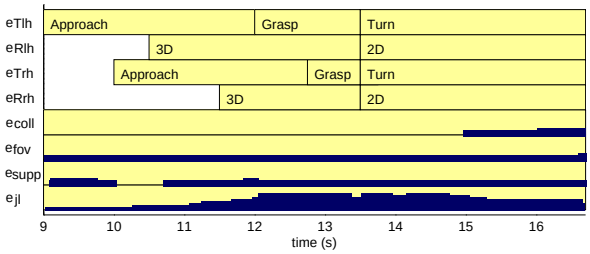


Fig. 16: *Simulation B-2: Task sequence of the second movement: the plots show the grasping phase (from $T = 9s$ to $T = 13.5s$) and a rotation of $\frac{2\pi}{3}$. For the four inequality tasks, the number of active constraints during the motion is plotted for each levels.*

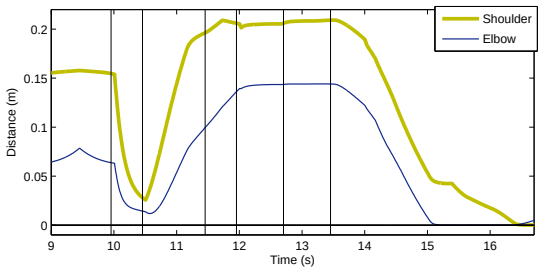


Fig. 17: *Simulation B-2: Distance to the obstacle.*

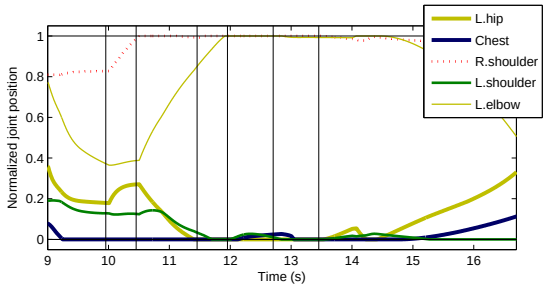


Fig. 18: *Simulation B-2: Normalized joint positions during the second movement.*

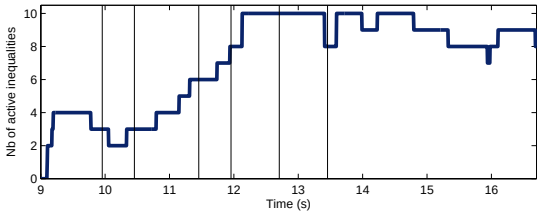


Fig. 19: *Simulation B-2: Number of active inequalities during the second movement.*

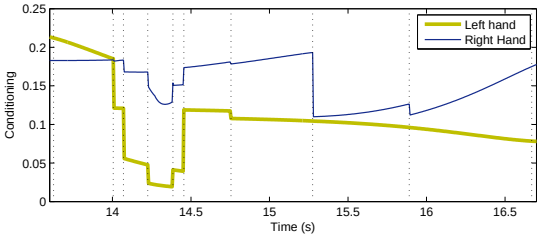


Fig. 20: *Simulation B-2: Conditioning number of the hand tasks when both hands are moving the wheel. The vertical dot lines show the inequality activations. The conditioning number evolves both continuously with the changes in the Jacobians and discretely at each constraint activation.*

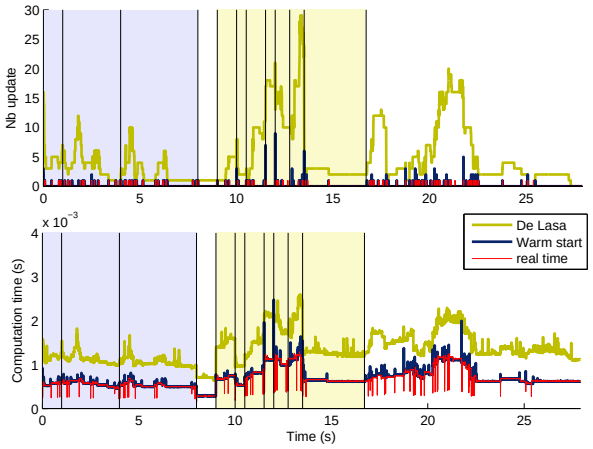


Fig. 21: *Simulation B: Number of algorithm iterations and computation time with the method proposed in [De Lasa et al., 2010], using our approach using a warm start and using a warm start and a bound on the number of active-set iteration (real-time).*

cases. The mean number of iterations is 0.035 and the maximum is 9. The mean of computation time is 0.9ms. The peaks of number of iterations correspond to peaks of computation time. When 9 iterations are needed, the algorithm takes 1.6ms to converge. This shows that the active set is not real time: it is not possible to predict how many iterations are needed to reach the optimum and, even if the mean is below 1ms, a control frequency of 1kHz is not possible.

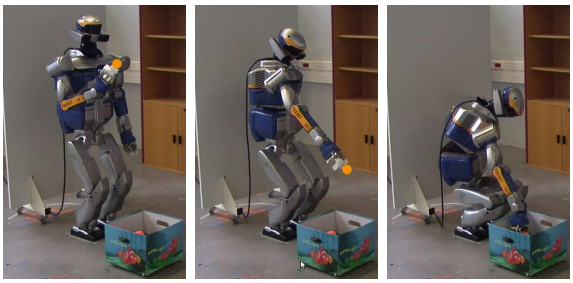
As proposed in Sec. 4.3, it is possible to arbitrarily limit the number of iterations to enforce a deterministic convergence time. The active-set was bound to maximum one update at each control cycle. When a second update is requested, the algorithm quits without even computing the dual optimum. The peaks of computation disappear. In exchange, the number of control cycles without update decreases to 96.6%. There is no perceptible changes in the robot behavior, since, due to the continuity properties, the obtained sub-optimum is very close to the solution.

Since the only inequalities are at the first levels of the hierarchy and are always compatible, the solver proposed in [De Lasa et al., 2010] can be used to generate the motion. As already noticed with the cascade of QP, the active search needs many iterations to find the optimal active set (up to 30, with a mean at 4.7). However, since each QP solved in the cascade is very small, the number of iterations only slightly impacts the computation cost. The convergence is slower than with the HQP (the mean is 1.3ms), but there is less peak than with our solver. The maximum is 2.6ms. The computation time is however always higher than for our solver.

6.4 Experiment C: grasping with posture constraints

This experiment is executed by the real HRP-2 robot. The robot has to grasp a point object while looking at it and avoiding its joint limits and the collisions with the environment. Three tasks are set at the least-priority levels to enforce the use of the upper limbs of the robot: the task e_{legs} is regulating the joint positions of the legs to the reference initial position; the task e_{chest} is regulating the orientation of the chest to keep it vertical, using (65). Finally, the last task e_{up} is blocking the upper part of the robot (chest, arms and neck). This kind of behavior using only the necessary limbs to perform an action was proposed in [Ee et al., 2007] using dedicated geometrical computations. The hierarchy is $e_{jl} \prec e_{coll} \prec e_{supp} \prec e_{rh} \prec e_{fov} \prec e_{legs} \prec e_{chest} \prec e_{up}$. The motion is summarized by Figures 22 to 27.

The ball is moved in three different positions, as shown in Fig. 22: first close in front of the robot, then at the height of the waist and finally on the ground behind a small box. The grasping task is finally removed when the last position is reached. The task sequence is shown in Fig. 23. When the ball is close enough, only the least-priority task e_{up} is violated, and the robot is grasping the ball using only its right arm. The neck and the left arm are marginally used, respectively for the FOV task and support task. When the ball is placed at the second position, it is out of the reach of the arm alone. The task e_{legs} is violated at the end of the grasping motion. When the ball is on the ground, it is not possible to grasp it with the chest being vertical. The task e_{chest} is violated at the end of the motion to reach the ball. Finally, the task e_{rh} is removed. The three tasks e_{legs}, e_{chest} and e_{up} are then feasible, and the robot goes back naturally to its initial position.



(t=2.1s) (t=4.2s) (t=7.3s)

Fig. 22: *Experiment C: Snapshots of the robot movement.*

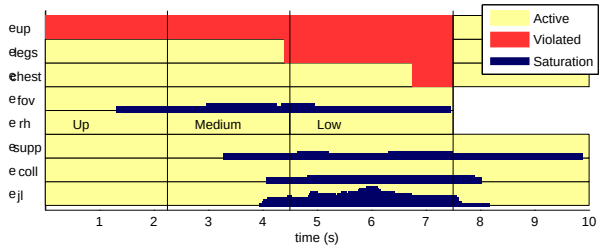


Fig. 23: *Experiment C: Task sequence.*

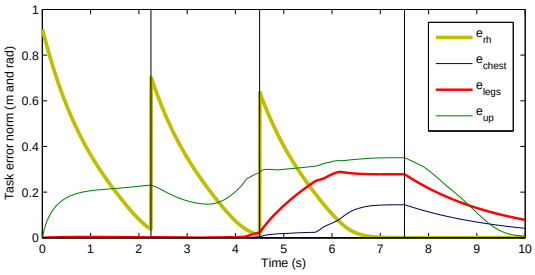


Fig. 24: *Experiment C: Norm of the task errors.*

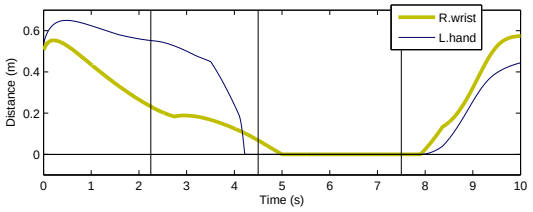


Fig. 25: *Experiment C: Distance to the obstacle.*

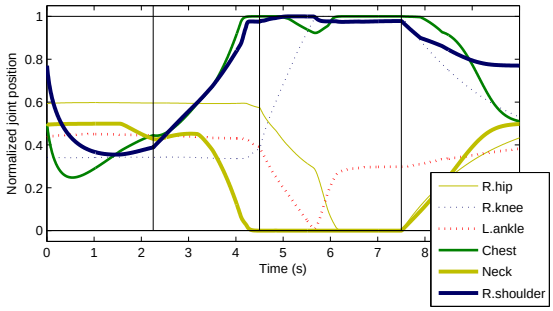


Fig. 26: *Experiment C: Normalized joint positions.*

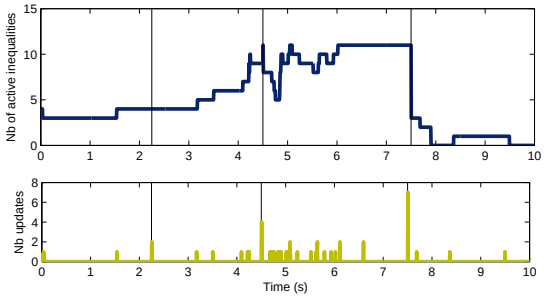


Fig. 27: *Experiment C: Number of active inequalities and updates.*

The errors of the four tasks are given in Fig. 24 and illustrate very well the hierarchical order: the task e_{rh} has priority over the three other ones, and is always accomplished: the error exponentially converges as imposed. The task e_{up} is violated first, and its error is the most important. The task e_{legs} is violated then, while the task e_{chest} is violated at the end, and keeps the lowest error value.

The activation of the limits is synthesized on Fig. 23. Some examples of activations are given in Figures 25 and 26. The left hand is moving backward to ensure the robot balance, and is quickly blocked by the task preventing the collision with the wall situated behind the robot. The right hand avoids a collision with the box when going to the third target position. Both collision constraints are deactivated when the robot moves back to the initial pose. Some joints limits of the legs are saturated when the robot goes for the second target position, and many limits are saturated when reaching the third target.

Finally, the number of active inequality constraints, and the number of iterations of the active-search loop are given in Fig. 27. As previously, the active set of the previous control cycle is used as warm start and reduces the number of iterations of the loop. The maximum number of iterations is 7 and is reached when the task e_{rh} is removed from the hierarchy. In average, 0.035 iterations and 0.6ms are needed at each control cycle, and 97.8% of the cycles are resolved without extra iteration. The timing scores are summarized on Table 1. For this last experiment, only the real-time version of the HQP was run by the physical robot, the other scores being obtained off line on a similar computer.

The Table 1 summarizes the computation performances obtained by the solvers of [Kanoun et al., 2011], [De Lasa et al., 2010] and our solver (without warm-start, with warm-start and with real-time). Our solver is much faster than the two previous ones. Moreover, the easily warm-started unified active-set loop increases the difference. We compute a solution five time faster than [Kanoun et al., 2011] in any case and two to three time faster on the examples that [De Lasa et al., 2010] is able to solve. Finally, our algorithm induces a much lower number of active-set iterations, which yields a better numerical behavior.

7 Conclusion

In this paper, we have proposed a generic solution to resolve a hierarchical least-square quadratic program defined by equality or inequality constraints. When only equalities are set, our resolution method comes back to the classical state-of-the-art solutions [Siciliano and Slotine, 1991] but is up to ten times faster. When the problem encompasses inequality constraints, a true hierarchy is solved with inequalities at any level. The resolution loop keeps a low number of iterations by using a unified active-set algorithm, in contrast to the cascades of QP used in [Kanoun et al., 2011, De Lasa et al., 2010]. Using a proper construction of the active-set search, our resolution method is five time faster than [Kanoun et al., 2011]; it is more generic than [De Lasa et al., 2010] (enabling the solver to take into account inequalities at any stage of the hierarchy) while being two times faster. The solver performs real-time resolution of humanoid whole-body problems at 200Hz on a classical personal computer and no specific hardware tuning.

This is the first time that hierarchical problems of this complexity are solved in real time on a real robot.

In the future, we will try to reduce the computation cost by predicting the future constraint activations. The solver was only applied to inverse the robot kinematics, but the dynamics could be handled as well. Only a simple obstacle-avoidance scheme was set up, and a proper link with a complete collision checker should be studied. Finally, the continuity of the control scheme is not ensured when adding or removing a task from the hierarchy, which is needed before being able to apply it as a basic solution on our robots. Openings to other kind of problems, such as those solved by a walking pattern generator, is also an important perspective.

	Kanoun	De Lasa	HQP	HQP warm-start	HQP real-time
Simu. A					
Visual grasp	2.08 (19)	NR	0.82 (4.2)	0.45 (.02)	0.43 (.02)
Simu. B					
Pipe gate	2.78 (17.6)	1.34 (4.7)	0.95 (2.6)	0.69 (.04)	0.67 (.03)
Exp. C					
Floor grasp	2.99 (18)	NR	0.92 (3.5)	0.63 (.03)	0.62 (.03)
Average	2.61 (18.2)	NR	0.89 (3.4)	0.59 (.03)	0.58 (.03)

Table 1: *Time scores for the three movements, in milliseconds. Secondary score between parenthesis is the average number of iterations after the first one. Non Relevant (NR) is indicated when the method does not apply.*

Acknowledgments

This work was supported by grants from the RobotHow.cog EU CEC project, Contract No. 288533 under the 7th Research program (`www.robohow.eu`), and French PSPC Romeo-2.

A Multimedia Extensions

Ext.	Type	Description
1	Video	The movement sequences presented in the section 6.
2	Code	Exemplary MATLAB code of the proposed HQP sovler.

B Intermediate proofs

B.1 Proof of (53)

The four Moore-Penrose conditions are used to defined the pseudo-inverse A^+ of A:

$$AA^+A = A \tag{72}$$
$$A^+AA^+ = A^+ \tag{73}$$
$$AA^+ \text{ is symmetric} \tag{74}$$
$$A^+A \text{ is symmetric} \tag{75}$$

The matrix $\underline{A}_p^\ddagger$ defined in Sec. 2.4 always respects (72), (73) and (75). If all the $N_1 ... N_k$ are zero (that is to say if no level conflicts with the above hierarchy), the fourth property (74) is also respected. In that case, $\underline{A}_p^\ddagger$ is strictly the pseudo-inverse of $\underline{A}_p$. In general, $\underline{A}_p^\ddagger$ respects only three of the four properties of Moore-Penrose. This matrix is a reflexive generalized inverse of $\underline{A}_p$.

In particular, the derivation of the last property (75) gives:

$$\underline{A}_p^\ddagger \underline{A}_p = \underline{Y}_p \underline{Y}_p^T = \sum_{k=1}^{p} Y_k Y_k^T \tag{76}$$

using $\underline{H}_p^\ddagger \underline{H}_p = I$ (by recurrence). Using this equality, we can verify that the multiplier exhibited in (53) satisfies the condition (29). Setting $\underline{\lambda}_k$ and w_k^* in (29) and using (76), we obtain:

$$\underline{A}_{k-1}^T \underline{\lambda}_k = -\underline{A}_{k-1}^T \underline{A}_{k-1}^{\ddagger T} A_k^T w_k^* = -\underline{Y}_{k-1} \underline{Y}_{k-1}^T A_k^T w_k^*$$

This last form is equal to $A_k^T w_k^*$ since, from (42) and (51), we have:

$$A_k^T w_k^* = \underline{Y}_{k-1} N_k^T (N_k \underline{y}_{k-1}^* - V_k^T b_k) \tag{77}$$

B.2 Algorithm 3 termination

We prove here that each outer loop of Algorithm 3 terminates. We note $\underline{m}$ the total number of constraints, and $w = (\|w_1\|, \cdots, \|w_p\|)$.

The first outer iteration ($k = 1$) begins with a sequence of activations (at most $\underline{m}$) until all the constraints are active or satisfied. There is no deactivation before this, and the property is always verified

after. All outer iterations have then the same behavior. Let us consider the k-th loop: the steps (58) are such that $\|w_k\|$ is non-increasing: $x^{(*i)}$ is computed so that w decreases, and $\|w_1\|, \cdots, \|w_{k-1}\|$ are constant.

Whenever an activation occurs, the constraint is activated with the corresponding slack variable equal to zero since it was feasible, so that $\|w_k\|$ does not increase.

When a constraint is deactivated at iteration i, two cases occur: if it is the r-th constraint of level k, the slack $w_{k,r}$ is strictly negative since it also the Lagrange multiplier of the constraint, then $w_{k,r}$ will be set to zero by the deactivation, yielding a strict decrease of $\|w_k\|$. Otherwise, the deactivated constraint belongs to a level $l < k$. It was selected for deactivation because it prevents $\|w_k\|$ to decrease. Then, either the next step will decrease $\|w_k\|$, or $x^{(*i+1)} = x^{(i+1)}$, meaning that another constraint needs to be immediately deactivated. This can happen only a finite number of time (bounded by the number of weakly active constraints at levels $1..k-1$) before a non-zero step is taken that strictly decreases $\|w_k\|$.

Any of the two deactivation cases occurs after the optimal $\|w_k\|$ has been reach for the current active set, and yields a strict decrease of $\|w_k\|$ inducing that the algorithm will never go back to this active set since $\|w_k\|$ will never increase. Since there are a finite number of possible active sets, the outer loop is bound terminates.

References

[Antonelli and Chiaverini, 1998] Antonelli, G. and Chiaverini, S. (1998). Task-priority redundancy resolution for underwater vehicle-manipulator systems. In *IEEE Int. Conf. on Robotics and Automation (ICRA'98)*, Leuven, Belgium.

[Antonelli and Chiaverini, 2006] Antonelli, G. and Chiaverini, S. (2006). Kinematic control of platoons of autonomous vehicles. *IEEE Trans. on Robotics*, 22(6):1285–1292.

[Baerlocher and Boulic, 2004] Baerlocher, P. and Boulic, R. (2004). An inverse kinematic architecture enforcing an arbitrary number of strict priority levels. *The Visual Computer*, 6(20):402–417.

[Behringer, 1977] Behringer, F. (1977). Lexicographic quasiconcave multiobjective programming. *Zeitschrift für Operations Research*, pages 103–116.

[Ben-Israel and Greville, 2003] Ben-Israel, A. and Greville, T. (2003). *Generalized inverses: theory and applications*. CMS Books in Mathematics. Springer, 2nd edition.

[Berenson et al., 2011] Berenson, D., Srinivasa, S., and Kuffner, J. (2011). Task space regions: A framework for pose-constrained manipulation planning. *Int. Journal of Robotics Research*, 30(12):1435–1460.

[Björck, 1996] Björck, A. (1996). *Numerical Methods for Least Squares Problems*. SIAM.

[Bouyarmane and Kheddar, 2011] Bouyarmane, K. and Kheddar, A. (2011). Multi-contact stances planning for multiple agents. In *IEEE Int. Conf. on Robotics and Automation (ICRA'11)*, Shangai, China.

[Boyd and Vandenberghe, 2004] Boyd, S. and Vandenberghe, L. (2004). *Convex Optimization*. Cambridge University Press.

[Chang and Dubey, 1995] Chang, T. and Dubey, R. (1995). A weighted least-norm solution based scheme for avoiding joints limits for redundant manipulators. *IEEE Trans. on Robotics and Automation*, 11(2):286–292.

[Chaumette and Marchand, 2001] Chaumette, F. and Marchand, E. (2001). A redundancy-based iterative scheme for avoiding joint limits: Application to visual servoing. *IEEE Trans. on Robotics and Automation*, 17(5):719–730.

[Chiaverini, 1997] Chiaverini, S. (1997). Singularity-robust task-priority redundancy resolution for real-time kinematic control of robot manipulators. *IEEE Trans. on Robotics and Automation*, 13(3):398–410.

[Chiaverini et al., 2008] Chiaverini, S., Oriolo, G., and Walker, I. (2008). Kinematically redundant manipulators. In Siciliano, B. and Khatib, O., editors, *Handbook of Robotics*, page 245268. Springer-Verlag.

[Collette et al., 2007] Collette, C., Micaelli, A., Andriot, C., and Lemerle., P. (2007). Dynamic balance control of humanoids for multiple grasps and noncoplanar frictional contacts. In *IEEE-RAS Int. Conf. on Humanoid Robots (Humanoid'07)*, Pittsburgh, USA.

[De Lasa et al., 2010] De Lasa, M., Mordatch, I., and Hertzmann, A. (2010). Feature-based locomotion controllers. In *ACM SIGGRAPH'10*.

[De Schutter and Van Brussel, 1988] De Schutter, J. and Van Brussel, H. (1988). Compliant robot motion i. a formalism for specifying compliant motion tasks. *Int. Journal of Robotics Research*, 7(4):3–17.

[Decré et al., 2009] Decré, W., Smits, R., Bruyninckx, H., and De Schutter, J. (2009). Extending itasc to support inequality constraints and non-instantaneous task specification. In *IEEE Int. Conf. on Robotics and Automation (ICRA'09)*, Kobe, Japan.

[Ee et al., 2007] Ee, N., Yokoi, K., Kajita, S., and Tanie, K. (2007). Whole-body motion generation integrating operator's intention and robot's autonomy in controlling humanoid robots. *IEEE Trans. on Robotics*, 23(4):763–775.

[Escande et al., 2010] Escande, A., Mansard, N., and Wieber, P.-B. (2010). Fast resolution of hierarchized inverse kinematics with inequality constraints. In *IEEE Int. Conf. on Robotics and Automation (ICRA'10)*, Anchorage, USA. (preliminary version of this journal paper).

[Escande et al., 2013] Escande, A., Mansard, N., and Wieber, P.-B. (2013). Hierarchical quadratic programing: companion report. Technical report, LAAS-CNRS. http://projects.laas.fr/gepetto/escande_ijrr13.

[Faverjon and Tournassoud, 1987] Faverjon, B. and Tournassoud, P. (1987). A local based approach for path planning of manipulators with a high number of degrees of freedom. In *IEEE Int. Conf. on Robotics and Automation (ICRA'87)*, volume 4, pages 1152–1159, Atlanta, USA.

[Garcia-Aracil et al., 2005] Garcia-Aracil, N., Malis, E., Aracil-Santonja, R., and Perez-Vidal, C. (2005). Continuous visual servoing despite the changes of visibility in image features. *IEEE Trans. on Robotics*, 21(6):415–421.

[Gienger et al., 2006] Gienger, M., Janben, H., and Goerick, C. (2006). Exploiting task intervals for whole body robot control. In *IEEE/RSJ Int. Conf. on Intelligent Robots and Systems (IROS'06)*, Beijin, China.

[Golub and Van Loan, 1996] Golub, G. and Van Loan, C. (1996). *Matrix computations*, chapter 5.5: The rank-deficient LS problem. John Hopkins University Press, 3rd edition.

[Hanafusa et al., 1981] Hanafusa, H., Yoshikawa, T., and Nakamura, Y. (1981). Analysis and control of articulated robot with redundancy. In *IFAC, 8th Triennal World Congress*, volume 4, pages 1927–1932, Kyoto, Japan.

[Herdt et al., 2010] Herdt, A., Diedam, H., Wieber, P., Dimitrov, D., Mombaur, K., and Diehl, M. (2010). Online walking motion generation with automatic footstep placement. *Advanced Robotics*, 24(5-6):719–737.

[Hofmann et al., 2009] Hofmann, A., Popovic, M., and Herr, H. (2009). Exploiting angular momentum to enhance bipedal center-of-mass control. In *IEEE Int. Conf. on Robotics and Automation (ICRA'09)*, Kobe, Japan.

[Isermann, 1982] Isermann, H. (1982). Linear lexicographic optimization. *Operations Research Spektrum*, 4:223–228.

[Kanoun, 2009] Kanoun, O. (2009). *Contribution à la plannification de mouvements pour robots humanoïdes (in English)*. PhD thesis, Univ. Toulouse, Toulouse, France.

[Kanoun et al., 2011] Kanoun, O., Lamiraux, F., and Wieber, P.-B. (2011). Kinematic control of redundant manipulators: generalizing the task priority framework to inequality tasks. *IEEE Trans. on Robotics*, 27(4):785–792.

[Khatib, 1986] Khatib, O. (1986). Real-time obstacle avoidance for manipulators and mobile robots. *Int. Journal of Robotics Research*, 5(1):90–98.

[Khatib, 1987] Khatib, O. (1987). A unified approach for motion and force control of robot manipulators: The operational space formulation. *International Journal of Robotics Research*, 3(1):43–53.

[Khatib et al., 2008] Khatib, O., Sentis, L., and Park, J. (2008). A unified framework for whole-body humanoid robot control with multiple constraints and contacts. In *European Robotics Symposium*, pages 303–312, Prague, Czech Republic.

[Khatib et al., 1996] Khatib, O., Yokoi, K., Chang, K., Ruspini, D., Holmberg, R., and Casal, A. (1996). Vehicle/arm coordination and multiple mobile manipulator decentralized cooperation. In *IEEE/RSJ Int. Conf. on Intelligent Robots and Systems (IROS'96)*, Osaka, Japan.

[Lee et al., 2012] Lee, J., Mansard, N., and Park, J. (2012). Intermediate desired value approach for task

transition of robots in kinematic control. *IEEE Transaction on Robotics*, 28(6):1260 – 1277.

[Li et al., 2012] Li, T., Kermorgant, O., and Krupa, A. (2012). Maintaining visibility constraints during tele-echography with ultrasound visual servoing. In *IEEE Int. Conf. on Robotics and Automation (ICRA'12)*, Saint Paul, USA.

[Liégeois, 1977] Liégeois, A. (1977). Automatic supervisory control of the configuration and behavior of multibody mechanisms. *IEEE Trans. on Systems, Man and Cybernetics*, 7(12):868–871.

[Mansard and Chaumette, 2007] Mansard, N. and Chaumette, F. (2007). Task sequencing for sensor-based control. *IEEE Trans. on Robotics*, 23(1):60–72.

[Mansard and Chaumette, 2009] Mansard, N. and Chaumette, F. (2009). Directional redundancy. *IEEE Transaction on Automatic Control*, 54(6):1179–1192.

[Mansard et al., 2009] Mansard, N., Khatib, O., and Kheddar, A. (2009). A unified approach to integrate unilateral constraints in the stack of tasks. *IEEE Transaction on Robotics*, 25(3).

[Mansard et al., 2007] Mansard, N., Stasse, O., Chaumette, F., and Yokoi, K. (2007). Visually-guided grasping while walking on a humanoid robot. In *IEEE Int. Conf. on Robotics and Automation (ICRA'07)*, Roma, Italia.

[Marchand and Hager, 1998] Marchand, E. and Hager, G. (1998). Dynamic sensor planning in visual servoing. In *IEEE/RSJ Int. Conf. on Intelligent Robots and Systems (IROS'98)*, Leuven, Belgium.

[Mordatch et al., 2012] Mordatch, I., Todorov, E., and Popović, Z. (2012). Discovery of complex behaviors through contact-invariant optimization. In *ACM SIGGRAPH'12*, Los Angeles, USA.

[Nelson and Khosla, 1995] Nelson, B. and Khosla, P. (1995). Strategies for increasing the tracking region of an eye-in-hand system by singularity and joint limits avoidance. *Int. Journal of Robotics Research*, 14(3):255–269.

[Nenchev, 1989] Nenchev, D. (1989). Redundancy resolution through local optimization: A review. *Journal of Robotic Systems*, 6(6):769–798.

[Nocedal and Wright, 2006] Nocedal, J. and Wright, S. J. (2006). *Numerical Optimization*. Springer, New York, 2nd edition.

[Park and Khatib, 2006] Park, J. and Khatib, O. (2006). Contact consistent control framework for humanoid robots. In *IEEE Int. Conf. on Robotics and Automation (ICRA'06)*, Orlando, USA.

[Pham and Nakamura, 2012] Pham, Q. and Nakamura, Y. (2012). Affine trajectory deformation for redundant manipulators. In *rss12*, Sydney, Australia. Best Paper Award.

[Raunhardt and Boulic, 2007] Raunhardt, D. and Boulic, R. (2007). Progressive clamping. In *IEEE Int. Conf. on Robotics and Automation (ICRA'07)*, Roma, Italy.

[Remazeilles et al., 2006] Remazeilles, A., Mansard, N., and Chaumette, F. (2006). Qualitative visual servoing: application to the visibility constraint. In *IEEE/RSJ Int. Conf. on Intelligent Robots and Systems (IROS'06)*, Beijing, China.

[Saab et al., 2013] Saab, L., Ramos, O., Mansard, N., Souères, P., and Fourquet, J.-Y. (2013). Dynamic whole-body motion generation under rigid contacts and other unilateral constraints. *IEEE Transaction on Robotics*, 29(2):346–362.

[Salini et al., 2009] Salini, J., Barthélemy, S., and Bidaud, P. (2009). Lqp controller design for generic whole body motion. In *IEEE Int. Conf. on Robotics and Automation (ICRA'09)*, Kobe, Japan.

[Samson et al., 1991] Samson, C., Le Borgne, M., and Espiau, B. (1991). *Robot Control: the Task Function Approach*. Clarendon Press, Oxford, United Kingdom.

[Sentis, 2007] Sentis, L. (2007). *Synthesis and Control of Whole-Body Behaviors in Humanoid Systems*. PhD thesis, Stanford University, USA.

[Sian et al., 2005] Sian, N., Yokoi, K., Kajita, S., Kanehiro, F., and Tanie, K. (2005). A switching command-based whole-body operation method for humanoid robots. *IEEE/ASME Transactions on Mechatronics*, 10(5):546–559.

[Siciliano and Slotine, 1991] Siciliano, B. and Slotine, J.-J. (1991). A general framework for managing multiple tasks in highly redundant robotic systems. In *IEEE Int. Conf. on Advanced Robotics (ICAR'91)*, Pisa, Italy.

[Stasse et al., 2008] Stasse, O., Escande, A., Mansard, N., Miossec, S., Evrard, P., and Kheddar, A. (2008). Real-time (self)-collision avoidance task on a HRP-2 humanoid robot. In *IEEE Int. Conf. on Robotics and Automation (ICRA'08)*, Pasadena, USA.

[Sung et al., 1996] Sung, Y., Cho, D., and Chung, M. (1996). A constrained optimization approach to resolving manipulator redundancy. *Journal of robotic systems*, 13(5):275–288.

[Wensing and Orin, 2013] Wensing, P. and Orin, D. (2013). Generation of dynamic humanoid behaviors through task-space control with conic optimization. In *IEEE Int. Conf. on Robotics and Automation (ICRA'13)*, Karlsruhe, Germany.

[Whitney, 1972] Whitney, D. (1972). The mathematics of coordinated control of prosthetic arms and manipulators. *Trans. ASME Journal of Dynamic System, Measures and Control*, 94:303–309.

[Yoshikawa, 1985] Yoshikawa, T. (1985). Manipulability of robotic mechanisms. *Int. Journal of Robotics Research*, 4(2):3–9.

[Zanchettin and Rocco, 2012] Zanchettin, A. and Rocco, P. (2012). A general user-oriented framework for holonomic redundancy resolution in robotic manipulators using task augmentation. *IEEE Trans. on Robotics*, 28(2):514–521.