Fast Resolution of Hierarchized Inverse Kinematics with Inequality Constraints

Adrien Escande, Nicolas Mansard, Pierre-Brice Wieber

***Abstract*— Classically, the inverse kinematics is performed by computing the singular value decomposition of the matrix to invert. This enables a very simple writing of the algorithm. However, the computation cost is high, especially when applied to complex robots and complex sets of constraints (typically around 5ms for 50 degrees of freedom – DOF). In this paper, we propose a dedicated adaptation of quadratic programming that enables fast computations of the hierarchical inverse kinematics (around 0.1ms for 50 DOF). We then extend this algorithm to deal with unilateral constraints, obtaining sufficiently high performances for reactive control.**

I. Introduction

Inverse kinematics (IK) is a very classical solution to design and control motions of robots. Instead of arbitrarily choosing the joint trajectories, it allows to design the desired motion in a well constructed *task* space [16], where the objectives or the links with sensors are easy to establish [5]. Moreover, the same inversion techniques can be applied to the dynamical equations of the system [10], [12]. Additionally, it is possible to combine by projection a secondary objective with the first inversion [15], while ensuring a complete decoupling [11]. This decoupling can be generalized to several tasks [14], and a control law can be obtained by stacking any number of tasks, ensuring a strict respect of the tasks order [17]. The hierarchy or *stack* of tasks (SOT) becomes more and more popular to build complex behaviors on redundant robots such as humanoid robots [1].

The first limitation of IK is the computation cost. IK generally relies on singular-value-decomposition (SVD) based pseudo-inverse, which is costly (typically 2 to 5 ms for a humanoid robot with about 40 DOF, while for dynamic tasks such as contact manipulation, the control loop should be 1ms). The second limitation is the formalization of inequality constraints (joint limits, obstacles, etc.) inside IK. The most widely-used solution to consider such constraints is the definition of a potential field [9], [13], whose gradient is projected at the lowest priority. In [8], [4], it was proposed to revisit the original link with optimization, by using quadratic programming (QP) *with inequalities* for IK. A cascade of QP then built a SOT with inequalities in any order of priority. Such a cascade was too expensive for real time control (typically 30ms).

A. Escande is with the CEA, LIST, Fontenay aux Roses, F-92265 France (adrien.escande@cea.fr). N. Mansard is with Université de Toulouse, LAAS/CNRS, France (nicolas.mansard@laas.fr). P-B. Wieber is with INRIA, Grenoble, France and with CNRS/AIST-JRL, Tsukuba, Japan (pierre-brice.wieber@inria.fr). This research was supported by the French Agence Nationale de la Recherche, grant reference ANR-08-JCJC-0075-01, and by the FUI Romeo project.

The contribution of this paper is twofold. **First**, only equality constraints are considered. The link between QP and IK is revisited to propose a more efficient IK algorithm. Compared to the typical 5ms performances of SVD-IK, we demonstrate 0.1ms for the same problem. Moreover, we prove that the algorithmic complexity of the algorithm does not depend on the size of the stack but only on the number of DOF of the robot. **Second**, we use this result to accelerate the algorithm proposed in [8], leading to 3ms of computation time on a HRP2 robot.

II. Hierarchized inverse kinematics

Task-based motion generation consists in defining the motion of the robot in terms of a reference in a simple task space e. This space can be for example the position of the hand, the gaze (*i.e.* the position of a point in the visual plane) or the position of the center of mass (CoM). A task k is completely defined by choosing the reference in the proper sub-space e_k, and modeling explicitly the generic link between the robot inputs and the task:

$$e_k = J_k u \tag{1}$$

with u the robot input (typically the joint velocity) and J_k the differential link to the robot input (when $u = \dot{q}$, J_k is usually the Jacobian $J_k = \frac{\partial e_k}{\partial q}$ and the reference e_k is the velocity in the task space $\dot{e}_k^*$).

When composing a set of tasks, the problem turns into solving a set of such linear equations. In the case of complex applications, some of these equations may be incompatible, at least temporarily (task singularities). One classical solution is then to introduce a hierarchy between the tasks. The hierarchy can be enforced by solving each equation (1) only in the null space of the previous ones. This solution is obtained with the recursive formula [17]

$$u_k = u_{k-1} + (J_k P_{k-1})^+ (e_k - J_k u_{k-1}) \tag{2}$$

where A^+ denotes the (Moore-Penrose) pseudo-inverse of a matrix A [2], and

$$P_k = I - \bar{J}_k^+ \bar{J}_k \tag{3}$$

is a matrix projecting on the null space of the augmented matrix:

$$\bar{J}_k = \begin{bmatrix} J_1 \\ \vdots \\ J_k \end{bmatrix}. \tag{4}$$

When beginning with $u_0 = 0$, $\bar{J}_0$ empty and therefore $P_0 = I$, this produces a sequence of solutions with minimum norm [17].

III. Equivalence with a least-squares problem

The way the pseudo-inverse is used in (2) indicates that this recursive formula solves in fact a least-squares problem with equality constraints (LSE)

$$\min_{u_k} \ \frac{1}{2}\|J_k u_k - e_k\|^2 \tag{5}$$
$$\text{subject to} \quad \bar{J}_{k-1} u_k = \bar{J}_{k-1} u_{k-1}. \tag{6}$$

When computing the solution u_k to this problem, the constraint (1) is solved in a least-squares sense by (5) while ensuring by (6) that the solution to the previous constraints $\bar{J}_{k-1}$ is left unchanged from the previous solution u_{k-1}.

Several classical solutions are available to solve this LSE, among them the null-space method [3], which begins by considering a general solution to the equality constraint (6) of the form

$$u_k = u_{k-1} + P_{k-1} z_k \tag{7}$$

where P_{k-1} is a matrix projecting on the null space of the augmented matrix $\bar{J}_{k-1}$ as in (3). Replacing this general solution in the minimization problem (5), we end up with an unconstrained least-squares problem

$$\min_{z_k} \ \frac{1}{2}\|J_k u_{k-1} + J_k P_{k-1} z_k - e_k\|^2, \tag{8}$$

The solution with minimum norm z_k is directly obtained using a pseudo-inverse. Replacing it in (7), the solution to the LSE (5)-(6) appears to be

$$u_k = u_{k-1} + P_{k-1}(J_k P_{k-1})^+ \left(e_k - J_k u_{k-1}\right) \tag{9}$$

which is strictly equivalent to (2) since $P_{k-1}(J_k P_{k-1})^+ = (J_k P_{k-1})^+$ [17].

IV. Usual matrix decompositions

The numerical method of reference to compute the pseudo-inverse in formula (2) is through a SVD:

$$J_k P_{k-1} = \begin{bmatrix} U_k & V_k \end{bmatrix} \begin{bmatrix} S_k & 0 \\ 0 & 0 \end{bmatrix} \begin{bmatrix} Y_k^T \\ Z_k^T \end{bmatrix} = U_k S_k Y_k^T \tag{10}$$

where $\begin{bmatrix} U_k & V_k \end{bmatrix}$ and $\begin{bmatrix} Y_k & Z_k \end{bmatrix}$ are orthonormal square matrices and S_k is the diagonal matrix of singular values. With this decomposition, the pseudo-inverse is obtained directly:

$$(J_k P_{k-1})^+ = \begin{bmatrix} Y_k & Z_k \end{bmatrix} \begin{bmatrix} S_k^{-1} & 0 \\ 0 & 0 \end{bmatrix} \begin{bmatrix} U_k^T \\ V_k^T \end{bmatrix} = Y_k S_k^{-1} U_k^T. \tag{11}$$

This is the most precise numerical method for obtaining a pseudo inverse [7], the most robust to the ill-conditioning of $J_k P_{k-1}$ that can happen when the robot is close to a task singularity. However, this solution is expensive to compute.

Another less frequent option is to use a Complete Orthogonal Decomposition (COD)

$$J_k P_{k-1} = \begin{bmatrix} U_k & V_k \end{bmatrix} \begin{bmatrix} 0 & 0 \\ L_k & 0 \end{bmatrix} \begin{bmatrix} Y_k^T \\ Z_k^T \end{bmatrix} = V_k L_k Y_k^T \tag{12}$$

where $\begin{bmatrix} U_k & V_k \end{bmatrix}$ and $\begin{bmatrix} Y_k & Z_k \end{bmatrix}$ are once again orthonormal square matrices but L_k is an invertible lower triangular matrix. The pseudo-inverse is also easily obtained:

$$(J_k P_{k-1})^+ = \begin{bmatrix} Y_k & Z_k \end{bmatrix} \begin{bmatrix} 0 & 0 \\ {L_k}^{-1} & 0 \end{bmatrix} \begin{bmatrix} U_k^T \\ V_k^T \end{bmatrix} = Y_k {L_k}^{-1} V_k^T. \tag{13}$$

${L_k}^{-1}$ requires $n^2/2$ computations instead of n for S^{-1}. On the other hand, this matrix decomposition is faster to compute than a SVD. It is usually considered as safe enough [7], even if more sensitive to ill-conditioning close to task singularities.

Concerning the projection matrix P_k, it is rarely computed directly from the formula (3) since an incremental formulation can be more efficient [1]:

$$P_k = P_{k-1} - (J_k P_{k-1})^+ J_k P_{k-1} \tag{14}$$

The efficiency of the recurrence is even improved by the matrix decompositions (10)-(11) or (12)-(13) which gives

$$P_k = P_{k-1} - Y_k Y_k^T. \tag{15}$$

V. The Hierarchized Complete Orthogonal Decomposition

The derivations of the previous sections correspond to the current state of the art in computing Hierarchized IK (HIK), but it seems there is still room for improvements. Let us begin by introducing an orthogonal decomposition

$$\bar{J}_{k-1} = \bar{W}_{k-1} \begin{bmatrix} \bar{L}_{k-1} & 0 \end{bmatrix} \begin{bmatrix} \bar{Y}_{k-1}^T \\ \bar{Z}_{k-1}^T \end{bmatrix} \tag{16}$$

of the augmented matrix $\bar{J}_{k-1}$, where $\bar{W}_{k-1}$ and $\begin{bmatrix} \bar{Y}_{k-1} & \bar{Z}_{k-1} \end{bmatrix}$ are orthonormal square matrices and $\bar{L}_{k-1}$ is a matrix with full column rank. This decomposition is not unique and depends on the specific structure imposed on $\bar{W}_{k-1}$ and $\bar{L}_{k-1}$. Now, having in mind that

$$\bar{J}_k = \begin{bmatrix} \bar{J}_{k-1} \\ J_k \end{bmatrix}, \tag{17}$$

we augment this decomposition in the following way,

$$\bar{J}_k = \begin{bmatrix} \bar{W}_{k-1} & 0 \\ 0 & I \end{bmatrix} \begin{bmatrix} \bar{L}_{k-1} & 0 \\ J_k \bar{Y}_{k-1} & J_k \bar{Z}_{k-1} \end{bmatrix} \begin{bmatrix} \bar{Y}_{k-1}^T \\ \bar{Z}_{k-1}^T \end{bmatrix}. \tag{18}$$

We can see in the lower right corner of the matrix in the middle a projected matrix $J_k \bar{Z}_{k-1}$. The COD of this matrix is

$$J_k \bar{Z}_{k-1} = \begin{bmatrix} U_k & V_k \end{bmatrix} \begin{bmatrix} 0 & 0 \\ L_k & 0 \end{bmatrix} \begin{bmatrix} Y_k^T \\ Z_k^T \end{bmatrix}, \tag{19}$$

Inserting it in the decomposition (18) gives:

$$\bar{J}_k = \begin{bmatrix} \bar{W}_{k-1} & 0 & 0 \\ 0 & U_k & V_k \end{bmatrix} \begin{bmatrix} \bar{L}_{k-1} & 0 & 0 \\ M_k & 0 & 0 \\ N_k & L_k & 0 \end{bmatrix} \begin{bmatrix} \bar{Y}_{k-1}^T \\ Y_k^T \\ Z_k^T \end{bmatrix} \tag{20}$$
$$= \bar{W}_k \begin{bmatrix} \bar{L}_k & 0 \end{bmatrix} \begin{bmatrix} \bar{Y}_k^T \\ \bar{Z}_k^T \end{bmatrix} \tag{21}$$

with

$$M_k = U_k^T J_k \bar{Y}_{k-1}, \ N_k = V_k^T J_k \bar{Y}_{k-1} \tag{22}$$

and

$$\bar{Y}_k = \begin{bmatrix}\bar{Y}_{k-1} & Y_k\end{bmatrix},\ \bar{Z}_k = Z_k. \tag{23}$$

Going from the decomposition (16) of the matrix $\bar{J}_{k-1}$ to the decomposition (21) of the matrix $\bar{J}_k$, it appears that this sequence of decompositions can be computed incrementally, involving a smaller COD (19) at each increment.

Doing so, the sequence of matrices $\bar{L}_k$ presents a specific structure which appears in (20): it is not a simple invertible lower triangular matrix as in a classical COD, but something similar which is going to be fundamental when dealing with inequalities in further sections. We propose to call this decomposition a *Hierarchized Complete Orthogonal Decomposition* (HCOD).

VI. A FASTER COMPUTATION SCHEME

With the help of the decomposition (16), the matrix projecting on the null space of the augmented matrix $\bar{J}_{k-1}$ appears to be directly

$$P_{k-1} = \bar{Z}_{k-1}\bar{Z}_{k-1}^T. \tag{24}$$

We can revisit then the results of Section III, introducing

$$z'_k = \bar{Z}_{k-1}^T z_k. \tag{25}$$

The general solution (7) can be written then

$$u_k = u_{k-1} + \bar{Z}_{k-1} z'_k, \tag{26}$$

leading to the alternative unconstrained least-squares problem

$$\min_{z'_k}\ \frac{1}{2}\|J_k u_{k-1} + J_k \bar{Z}_{k-1} z'_k - e_k\|^2 \tag{27}$$

The solution with minimum norm is

$$z'_k = (J_k \bar{Z}_{k-1})^+ \left(e_k - J_k u_{k-1}\right), \tag{28}$$

leading finally to an alternative formulation of the solution

$$u_k = u_{k-1} + \bar{Z}_{k-1}(J_k \bar{Z}_{k-1})^+ \left(e_k - J_k u_{k-1}\right). \tag{29}$$

This formulation can be further simplified thanks to the second relation in (23):

$$u_k = u_{k-1} + Z_{k-1}(J_k Z_{k-1})^+ \left(e_k - J_k u_{k-1}\right) \tag{30}$$
$$= u_{k-1} + Z_{k-1} Y_k {L_k}^{-1} V_k^T \left(e_k - J_k u_{k-1}\right), \tag{31}$$

involving matrices $J_k Z_{k-1}$ which are strictly equal to the matrices $J_k \bar{Z}_{k-1}$, but which are now obtained directly from the COD (19) and not from the decomposition (16). This way, it appears that only the sequence of COD of matrices $J_k Z_{k-1}$ are need, and not the whole decomposition (16).

This alternative formulation is faster to compute than the original (2) since the size of the matrices Z_{k-1} decreases when advancing in the hierarchy of tasks (when k increases), while the original matrices P_{k-1} have a constant size. Thanks to this property, the pseudo-inverses of $J_k Z_{k-1}$ are cheaper with increasing k, and much cheaper in the end than the pseudo-inverses of the original bigger matrices $J_k P_{k-1}$.

As an example, consider a robot with m DOF, having N compatible tasks of dimension n to realize (so that they all have full rank even when projected), with n significantly smaller than m. If considering the original recurrences (9) and (15) based on SVD, the cost for each task is independent from the range of the task in the SOT, and is approximately $2nm^2$ flops. If using the same scheme with a COD, the cost for each task is similar since the main computation is due to (15).

On the opposite, if using (30), the cost per task depends on the number of unconstrained DOF, which are $n_f = n - km$ for the task at range k. The cost for the k^{th} task is then $(2n+3)kn(m-kn)$ flops (split into $n(m^2 - m_f^2)$ for the projection $J_k \bar{Z}_{k-1}$, $2n^2 m_f$ for the COD (19), $\frac{3}{2}(m - m_f)^2$ for (30)).

For a robot with $m = 36$ DOF, constrained by $N = 6$ tasks of dimension $n = 6$, the SVD leads to 168Kflops, while the COD leads to 124Kflops. For the same 36DOF-robot, the total cost of our algorithm is 26.9 Kflops. As a matter of comparison, solving the M tasks at once (by inverting directly $\bar{J}_M$) would cost 1213 Kflops with the SVD and 69.5 Kflops with the COD. This is because the recursive formula (29) computes a block decomposition, what appears to be much faster.

VII. INTRODUCING INEQUALITY CONSTRAINTS

It has been proposed in [8] to consider also inequalities in the tasks, for taking into account more effectively balance, visibility, collision and self-collision avoidance, and other safety goals. Compared to earlier methods such as [11], this solution enables to consider inequalities with any priority, from the lowest to the highest, enforcing them at all time during the motion. The sequence of equations (1) becomes therefore a sequence of inequalities

$$e_k^l \leq J_k u \leq e_k^u, \tag{32}$$

where equations simply correspond to cases where $e_k^l = e_k^u$.

The algorithm proposed in [8] to solve this sequence of inequalities in a hierarchy involves a sequence of least-squares problems with inequality constraints (LSI)

$$\min_{u_k, w_k}\ \frac{1}{2}\|w_k\|^2 \tag{33}$$
$$e_k^l \leq J_k u_k - w_k \leq e_k^u \tag{34}$$
$$\bar{e}_{k-1}^l \leq \bar{J}_{k-1} u_k \leq \bar{e}_{k-1}^u \tag{35}$$

with a structure similar to the sequence of LSE introduced in (5)-(6). Here, in case of a conflict between the constraints, the *slack variable* vector w_k introduced in the constraints (34) allows to violate (34), giving therefore a priority to the constraints (35). In that case, this slack variable vector measures the amount of violation, which is minimized as much as possible by (33). Once minimized, this violation is frozen before considering the LSI at the next level of priority, propagating the bounds

$$\bar{e}_k^l = \begin{bmatrix}\bar{e}_{k-1}^l \\ e_k^l + w_k\end{bmatrix},\ \bar{e}_k^u = \begin{bmatrix}\bar{e}_{k-1}^u \\ e_k^u + w_k\end{bmatrix}, \tag{36}$$

which are equivalent to the propagation of the constraints in (6).

As proposed in [8], any state of the art QP solver can be used to solve this sequence of LSIs (33)-(35), one after the other, for $k = 1 \dots n$. However with a close inspection of the inner workings of standard QP solvers and with the help of the HCOD, we propose in the following sections a solver specifically designed for this particular sequence of LSIs in order to deliver faster computations.

VIII. A classical primal active set algorithm

At the minimum of a LSI (33)-(35), only a subset $\mathcal{A}_k$ of the inequality constraints (34)-(35) are active and hold as equality constraints. This subset is called the active set. An important observation is that considering only this subset of constraints and turning them into equalities, the minimum of the corresponding LSE is equal to the minimum of the original LSI. Solving the LSI consists then in finding the active set and solving this associated LSE.

Active set algorithms are based on this observation and try to guess iteratively the composition of the set $\mathcal{A}_k$, solving at each iteration the corresponding LSE. Many variants exist and we will consider here a classical *primal* active set algorithm. This algorithm maintains at each iteration a guess $\hat{\mathcal{A}}_k$ of the active set and a solution candidate (u_k, w_k) satisfying all the constraints (34)-(35) and lying exactly on the constraints included in $\hat{\mathcal{A}}_k$. At each iteration, it solves the corresponding LSE and modifies the guess $\hat{\mathcal{A}}_k$ if necessary, by activating or deactivating a constraint. If a constraint is violated when solving the LSE, this constraint is added to $\hat{\mathcal{A}}_k$. On the other hand, if the minimum of the LSE can be reached without violating any constraint, the algorithm checks if all the constraints in $\hat{\mathcal{A}}_k$ should really be active by checking the sign of the corresponding Lagrange multipliers.

These Lagrange multipliers can be computed with the help of the first order optimality conditions of the LSI (33)-(35), taking into account the specific guess $\hat{\mathcal{A}}_k$ of the active set:

$$\begin{bmatrix} 0 & 0 & \hat{\bar{J}}_{k-1}{}^T & \hat{J}_k^T \\ 0 & I & 0 & -I \\ \hat{\bar{J}}_{k-1} & 0 & 0 & 0 \\ \hat{J}_k & -I & 0 & 0 \end{bmatrix} \begin{bmatrix} u_k \\ w_k \\ \lambda_k \\ \mu_k \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \\ \hat{\bar{e}}_{k-1} \\ \hat{e}_k \end{bmatrix}, \tag{37}$$

where $\hat{J}_k$, $\hat{\bar{J}}_k$, $\hat{e}_k$ and $\hat{\bar{e}}_k$ correspond to the active constraints and the Lagrange multipliers μ_k and λ_k correspond respectively to the constraints (34) and (35). The second line shows that the multiplier μ_k is simply equal to the slack variable w_k:

$$\mu_k = w_k = J_k u_k - e_k \tag{38}$$

The other Lagrange multiplier is then computed from the first line:

$$\lambda_k = -\bar{J}_{k-1}^{+T} J_k^T \mu_k \tag{39}$$

IX. Some important implementation details

At each iteration of the active set algorithm, we have a new guess $\hat{\mathcal{A}}_k$ and we need to solve the corresponding LSE

$$\min_{u_k, w_k} \ \frac{1}{2}\|w_k\|^2 \tag{40}$$

$$\hat{J}_k u_k - w_k = \hat{e}_k \tag{41}$$

$$\hat{\bar{J}}_{k-1} u_k = \hat{\bar{e}}_{k-1} \tag{42}$$

which can be rewritten

$$\min_{u_k} \ \frac{1}{2}\|\hat{J}_k u_k - \hat{e}_k\|^2 \tag{43}$$

$$\hat{\bar{J}}_{k-1} u_k = \hat{\bar{e}}_{k-1} \tag{44}$$

by getting rid of the slack variable w_k.

Following the derivations of the previous sections, it appears that this LSE can be efficiently solved by (30), with the help of the orthogonal decompositions (16) and (19). However, each new guess $\hat{\mathcal{A}}_k$ introduces modifications of the active constraints $\hat{J}_k$ and $\hat{\bar{J}}_{k-1}$, adding or removing a line when a constraint is activated or deactivated: recomputing these decompositions each time would be very inefficient. Hopefully, one key property of orthogonal decompositions is that they can be efficiently updated when the initial matrices are modified in a structured way like here, what is not the case of the SVD [3].

Focusing especially on the HCOD (20), which gathers both decompositions (16) and (19) in a structured way, we have been able to design efficient and numerically robust updates corresponding to each case, when a constraint is activated or deactivated either in $\hat{J}_k$ or $\hat{\bar{J}}_{k-1}$, in a comparable way to what can be found in state of the art QP solvers [6]. This allows reducing strongly the computation cost required by each iteration of the active-set algorithm. Still, each iteration remains costly, so the total number of iterations should be reduced as much as possible. The main road to reducing this total number of iterations goes through a proper initialization of the algorithm and warm starting, what will be discussed in the next sections.

X. Initialization of the algorithm

The primal active-set algorithm presented in the previous sections requires for each LSI (33)-(35) an initial solution candidate $(\hat{u}_k, \hat{w}_k)$ which satisfies all the constraints (34)-(35). In the case of a generic LSI, finding such a point can be as hard as solving the complete LSI afterward. However two properties specific to our problem make this potential difficulty trivial to solve.

First of all, given any $\hat{u}_k$, it is straightforward to find a $\hat{w}_k$ such that the constraints (35) are satisfied. It is even very easy to find a $\hat{w}_k$ that minimizes at the same time the cost function (33), considering separately each coordinate j:

$$\hat{w}_{k_j} = \begin{cases} 0 & \text{when } e^l_{k_j} \le J_{k_j}\hat{u}_k \le e^u_{k_j}, \\ J_{k_j} u_k - e^u_{k_j} & \text{when } J_{k_j}\hat{u}_k > e^u_{k_j}, \\ J_{k_j} u_k - e^l_{k_j} & \text{when } e^l_{k_j} > J_{k_j}\hat{u}_k. \end{cases} \tag{45}$$

The only remaining point is to find a $\hat{u}_k$ that satisfies the constraints (34). Through the definition (36) of these constraints, it appears that the minimum of the LSI at level $k-1$ satisfies by definition the constraints (34) of the LSI at level k. The previous minimum constitutes therefore a perfect initial point. For the first LSI of the sequence, there are simply no such constraints. It can be initialized then with any $\hat{u}_1$ (typically 0).

Finally, we can observe that while reusing the minimum of the previous LSI, we can also reuse the corresponding active set $\mathcal{A}_{k-1}$ as an initial guess, what helps reducing drastically the number of iterations necessary to find the active set $\mathcal{A}_k$ of the following LSI. This initial guess can be easily completed with the constraints activated in (45).

XI. Warm start

Having solved an initial optimization problem, warm start consists in using this optimal solution to help for the resolution of a new problem whose data are slightly disturbed from the original. This is typically the case when considering a problem of motion generation: from one time of the control to the next, the problem shape is similar, with small modifications of the numerical values of (33)-(35).

Several different solutions can be considered as a warm start. In this article, we only considered a warm start based on the modification the active set $\mathcal{A}_k$. We said upper that the initial active set candidates $\hat{\mathcal{A}}_k$ is set to be the active set of the previous stage of the SOT. However, this straightforward initialization does not account for the constraints (34), nor for their impact on the activation of (35).

The active set at time t and for stage k is denoted $\mathcal{A}_{k,t}$. The deltas of the active set that are due to the new constraints (34) are denoted $\Delta^+\mathcal{A}_{k+1,t}$ and $\Delta^-\mathcal{A}_{k+1,t}$ so that $\mathcal{A}_{k+1,t} = (\mathcal{A}_{k,t} \setminus \Delta^-\mathcal{A}_{k+1,t}) \cup \Delta^+\mathcal{A}_{k+1,t}$. We simply propose to apply these deltas recorded at time t to compute the initial guess at time $t+1$.

It is straightforward to apply $\Delta^-\mathcal{A}_{k,t}$ at time $t+1$: simply remove from the HCOD all the active constraints of the delta. However, if applying similarly $\Delta^+\mathcal{A}_{k,t}$, there is a problem to find the initial point $(\hat{u}, \hat{w}_k)$. It is theoretically possible to compute numerically a initial point. However, we have noticed empirically that this computation is very costly since requiring several modifications of $\hat{\mathcal{A}}_k$.

To avoid these extra iterations, the delta of the variable u is also stored: $\Delta u_{k+1,t} = u_{k+1,t} - u_{k,t}$. The delta is applied at time t on the optimum of the previous stage to modify the initial point:

$$\hat{u}_{k+1,t+1} := u_{k,t+1} + \tau \Delta u_{k+1,t} \tag{46}$$

where τ is the step length used to ensure that inactive constraints are still respected. Thanks to continuity property, $\hat{u}_{k+1,t+1}$ is close to the solution of the LSI (33)-(35) at stage $k+1$ and time $t+1$. For the same reasons, all the constraints of $\Delta^+\mathcal{A}_H^{k+1,t}$ are close to be saturated at this point. From this initial point, all the constraints $\Delta^+\mathcal{A}_H^{k+1,t}$ are added to the HCOD, and the corresponding initial point is computed by a classical LSE resolution.

Experimentally, we checked first that τ is close to 1 and that activating $\Delta^+\mathcal{A}_H^{k+1,t}$ at the point $\hat{u}_{k+1,t+1}$ is strait forward (no violation of extra constraint, etc). Although perhaps not theoretically perfect, this approach proved to work in our experiments.

XII. Experiments

A. Experiment settings

The experiments have been done in simulation, to homogenize the computation times and allow constraint violations. We used a dynamical model of a humanoid robot HRP-2 (30 actuated joints plus 6 DOF on the free-floating root). The typical experiment consists in (from the top priority to the lowest one) maintaining balance (CoM control, 3DOF), keeping both feet on the ground (6 DOF), reaching a desired (6 DOF) using one hand of the robot and minimizing the velocity input (30 DOF). All these constraints are equalities: for each task, an exponential decrease of the error is imposed by setting $\dot{e}^* = -\lambda e$, with λ the (user-tuned) gain. The corresponding constraint can be written:

$$Ju = -\lambda e \tag{47}$$

The inequalities are introduced at the first level of the stack, to guarantee the joint limits:

$$q^l - q \leq u\Delta t \leq q^u - q \tag{48}$$

with q^l, q^u the lower and upper limits and Δt the period of the control.

To validate the algorithm, the object to be handed is set out of the reach of the robot. We check that the joint limits are always respected, as well as the CoM and the feet on the ground. Due to the out-of-reach desired position, the reaching task does not converge to 0 but to the lowest possible value. To validate the downgrade, the desired position is set back in the robot range: all tasks have then to converge to 0.

B. Results

A typical execution is summed up in Figures 1 to 4. A typical iteration is given on Fig. 1. Three time costs are compared: the first one is without using any warm start (this is typically what is done in [8] when classical QPs are used to solve each stack level separately); the second uses only the warm start from one stack level to the other; the last uses time-based warm start as well. The global cost using all warm starts is around 2ms to 3ms. Fig. 2 gives the fluctuation of the computation cost over the time. The solver cost is nearly constant, but on a small number of peaks. These peaks correspond to the activation of a new constraint: indeed, at these points, the new constraint will not be activated by the warm-start, but has to be found in several iterations (in general two) of the solver.

Finally, Fig. 3 gives the joint trajectories while Fig. 4 gives the evolution of the errors. We can see that each error is a perfect exponential decrease when the task is full rank. The reaching task then converges to the lowest possible value. When the desired position is set back inside the range of the robot, the task converges to regulation. All the other tasks (that have priority) are kept regulated at all time.

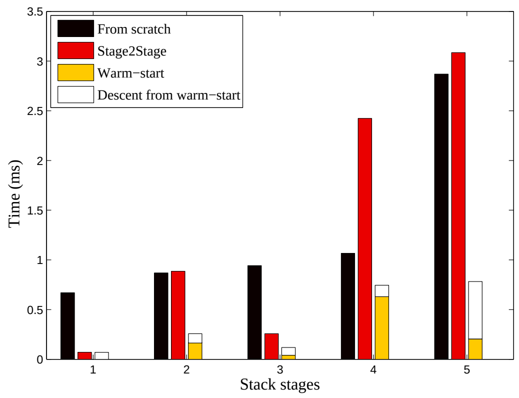

Fig. 1. Computation cost (in ms) for the set of five QPs alone (*from scratch*), the five QPs using at each stage N+1 the result of stage N for initialization (*stage2stage*), and using warm start (this last one being divided in two cost, for warm start and for the QP itself).

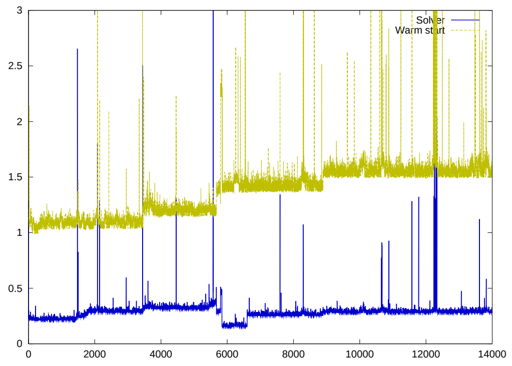

Fig. 2. Computation cost (in ms) over time (in iteration number). The warm-start cost increase with the number of saturated constraints, while the solver cost stay more constant, with peaks at constraint activation.

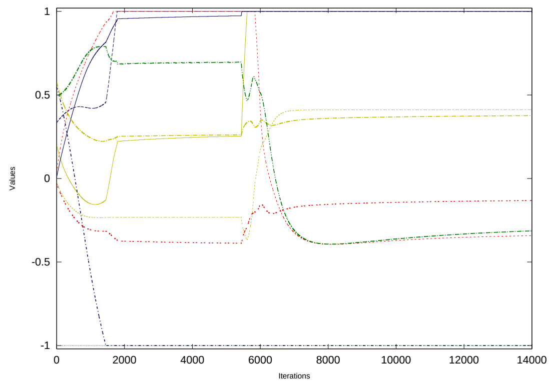

Fig. 3. Normalized joint trajectory of the hip, chest, and right arm. 1 and -1 are the two limits.

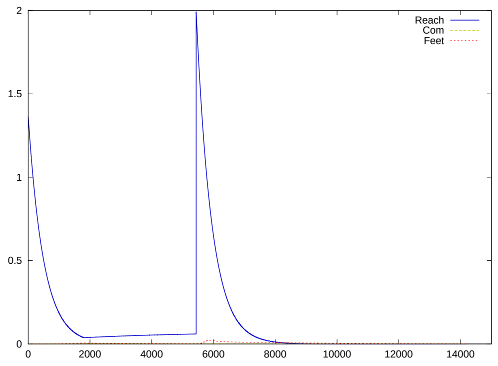

Fig. 4. Task evolution over time: the reaching task cannot converge before time 6000 since the desired position is out of reach. At time 6000, the desired position is changed, and the task converges. The other two tasks remains null since having priority.

XIII. Conclusion

The contributions of this paper are twofold: first a new QR-based algorithm was proposed to dramatically reduce the computation cost of the HIK, allowing to reduce the typical computation time from 5ms to 0.1ms. Second, this same approach was applied to write a specialized QP solver, that allows to compute a typical inverse-kinematics control law while accounting for inequalities constraints having priority in less than 5ms. Both algorithms were applied to control a humanoid-robot model in real time.

References

[1] P. Baerlocher and R. Boulic. An inverse kinematic architecture enforcing an arbitrary number of strict priority levels. *The Visual Computer*, 6(20):402–417, August 2004.
[2] A. Ben-Israel and T. Greville. *Generalized inverses: theory and applications*. CMS Books in Mathematics. Springer, 2nd edition, 2003.
[3] Å Björck. *Numerical methods for least squares problems*. SIAM, 1996.
[4] W. Decré, R. Smits, H. Bruyninckx, and J. De Schutter. Extending itasc to support inequality constraint and non-instantaneous task specification. In *IEEE Int. Conf. on Robot.& Automation (ICRA'09)*, Kobe, Japan, April 2009.
[5] B. Espiau, F. Chaumette, and P. Rives. A new approach to visual servoing in robotics. *IEEE Trans. on Robotics and Automation*, 8(3):313–326, June 1992.
[6] P.E. Gill, W. Murray, M.A. Saunders, and M.H. Wright. Procedures for optimization problems with a mixture of bounds and general linear constraints. *ACM Transactions on Mathematical Software*, 10:282–298, 1984.
[7] G. Golub and C. Van Loan. *Matrix computations*. John Hopkins University Press, 1996.
[8] O. Kanoun, F. Lamiraux, F. Kanehiro, E. Yoshida, and Laumond J-P. Prioritizing linear equality and inequality systems: application to local motion planning for redundant robots. In *IEEE Int. Conf. on Robot.& Automation (ICRA'09)*, Kobe, may 2009.
[9] O. Khatib. Real-time obstacle avoidance for manipulators and mobile robots. *Int. Journal of Robotics Research*, 5(1):90–98, Spring 1986.
[10] O. Khatib. A unified approach for motion and force control of robot manipulators: The operational space formulation. *International Journal of Robotics Research*, 3(1):43–53, February 1987.
[11] A. Liégeois. Automatic supervisory control of the configuration and behavior of multibody mechanisms. *IEEE Trans. on Systems, Man and Cybernetics*, 7(12):868–871, December 1977.
[12] N. Mansard, O. Khatib, and A. Kheddar. Integrating unilateral constraints inside the stack of tasks. *IEEE Trans. on Robotics*, 25(11):2493–2505, November 2009.
[13] E. Marchand and G. Hager. Dynamic sensor planning in visual servoing. In *IEEE/RSJ Int. Conf. on Intelligent Robots and Systems (IROS'98)*, Leuven, Belgium, May 1998.
[14] Y. Nakamura and H. Hanafusa. Inverse kinematics solutions with singularity robustness for robot manipulator control. *ASME Journ of Dyn. Sys., Measures and Control*, 108:163–171, September 1986.
[15] J. Rosen. The gradient projection method for nonlinear programmimg, part i, linear constraints. *SIAM Journal of Applied Mathematics*, 8(1):181–217, March 1960.
[16] C. Samson, M. Le Borgne, and B. Espiau. *Robot Control: the Task Function Approach*. Clarendon Press, Oxford, United Kingdom, 1991.
[17] B. Siciliano and J-J. Slotine. A general framework for managing multiple tasks in highly redundant robotic systems. In *IEEE Int. Conf. on Advanced Robot. (ICAR'91)*, Pisa, Italy, June 1991.