HAL
open science

Efficient resolution of potentially conflicting linear constraints in robotics

Dimitar Dimitrov, Alexander Sherikov, Pierre-Brice Wieber

To cite this version:

Dimitar Dimitrov, Alexander Sherikov, Pierre-Brice Wieber. Efficient resolution of potentially conflicting linear constraints in robotics. 2015. hal-01183003

HAL Id: hal-01183003

https://inria.hal.science/hal-01183003

Preprint submitted on 5 Aug 2015

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

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

Efficient resolution of potentially conflicting linear constraints in robotics

Dimitar Dimitrov, Alexander Sherikov and Pierre-Brice Wieber

***Abstract*—A classical approach to handling potentially conflicting linear equality and inequality constraints in robotics is to impose a strict prioritization between them. Ensuring that the satisfaction of constraints with lower priority does not impact the satisfaction of constraints with higher priority is routinely done by solving a hierarchical least-squares problem. Such a task prioritization is often considered to be computationally demanding and, as a result, it is often approximated using a standard weighted least-squares problem. The main contribution of this article is to address this misconception and demonstrate, both in theory and in practice, that the hierarchical problem can in fact be solved faster than its weighted counterpart. The proposed approach to efficiently solving hierarchical least-squares problems is based on a novel matrix factorization, to be referred to as "lexicographic QR", or ℓ-QR in short. We present numerical results based on three representative examples adopted from recent robotics literature which demonstrate that complex hierarchical problems can be tackled in real-time even with limited computational resources.**

I. Introduction

In the development of control and estimation schemes, the equation of motion of a robot [1]

$$H(q)\ddot{q} + h(q,\dot{q}) = \tau + J_c^T(q)f, \tag{1}$$

is rarely considered on its own. The successful execution of behaviors often relies on interactions with the environment that are commonly modeled by imposing restrictions on the contact forces f and/or on the generalized accelerations $\ddot{q}$. For example, unilateral contact and friction limits imply a constraint of the form

$$Ef \geq 0. \tag{2}$$

Moreover, the operation of robots in dynamic environments, where humans are possibly present, requires the consideration of safety constraints, such as

$$J_d(q)\ddot{q} + \dot{J}_d(q)\dot{q} \geq \ddot{d}, \tag{3}$$

where $J_d(q)$ is a configuration dependent Jacobian matrix and $\ddot{d}$ is the minimal deceleration required for the closest point to the environment to avoid collision. Explicitly accounting for limitations

$$-\tau_{max} \leq \tau \leq \tau_{max} \tag{4}$$

of the motor torques τ is also often necessary to avoid damaging them. Furthermore, additional goals ("tasks" [2]) might be considered, such as a desired dynamic behavior of an end-effector e:

$$J_e(q)\ddot{q} + \dot{J}_e(q)\dot{q} = \ddot{e}_d, \tag{5}$$

where $\ddot{e}_d$ is a desired acceleration, corresponding for example to an impedance control.

Such a typical problem in robotics can be formulated as the concatenation of P systems of linear equality and inequality constraints

$$\underbrace{\begin{bmatrix} b_1 \\ \vdots \\ b_P \end{bmatrix}}_{b} \leq \underbrace{\begin{bmatrix} C_1 \\ \vdots \\ C_P \end{bmatrix}}_{C} x \leq \underbrace{\begin{bmatrix} u_1 \\ \vdots \\ u_P \end{bmatrix}}_{u}, \tag{6}$$

where C_k is the constraint matrix of the k-th system, and $b_k \leq u_k$ are its lower and upper bounds (vector inequalities are to be interpreted componentwise). Equality constraints can be imposed by using $b_k = u_k$. In the above example, $x = (\ddot{q}, \tau, f)$.

In many problems of interest, however, there might not exist any x that satisfies all constraints due to conflicts among them. In such cases, simply concluding that the problem of interest is infeasible is often not acceptable, and the ability to handle infeasibility in a meaningful way is critical. Since it is not known in advance whether the system (6) admits a solution, a standard approach is to consider a relaxation

$$\begin{bmatrix} b_1 \\ \vdots \\ b_P \end{bmatrix} \leq \begin{bmatrix} C_1 \\ \vdots \\ C_P \end{bmatrix} x - \underbrace{\begin{bmatrix} v_1 \\ \vdots \\ v_P \end{bmatrix}}_{v} \leq \begin{bmatrix} u_1 \\ \vdots \\ u_P \end{bmatrix}, \tag{7}$$

potentially allowing constraint violations $v_1, \ldots, v_P$. Deciding how to minimize these violations, which kind of violation we are ready to accept, or prefer, appears to be a very important design choice. To this end, we can recognize that such problems usually involve groups of constraints of different nature, bearing different meanings of different importance, *e.g.*, physical constraints, safety constraints for the environment or the robot, behavior related constraints, etc. We can note that it is by observing such differences of meaning and importance that Isaac Asimov introduced the famous three laws of robotics:

1. A robot may not injure a human being or, through inaction, allow a human being to come to harm.
2. A robot must obey the orders given to it by human beings, except where such orders would conflict with the 1st law.
3. A robot must protect its own existence as long as such protection does not conflict with the 1st or 2nd law.

These laws are stated in a way that clearly imposes a strict hierarchy of importance between different goals. In the case of the typical problem described above, we could envision the following strict hierarchy:

$$(1,2) \succ (3) \succ (4) \succ (5),$$

implying that the physics (1)-(2) of the robot must naturally be considered before anything else; then, the safety of humans (3) is considered to be strictly more important than the safety of the robot (4), and the realization of additional tasks (5) may be attempted only after these safety concerns have been taken care of. In the context of robotics, such hierarchical approaches have been particularly popular in Cartesian motion control schemes (task prioritization in Inverse Kinematics, Task Function Control, Operational Space Control) [3], [4], [2], [5], [6].

This definition of a problem with a strict hierarchy suggests a guideline for approaching its solution. We can minimize first the violation v_1 of the physical constraints in a least-squares sense:

$$\begin{aligned} &\underset{x,v}{\text{minimize}} \ \ \|v_1\|^2 \\ &\text{subject to } b \leq Cx - v \leq u. \end{aligned}$$

Once this has been achieved, we can minimize the violation v_2 of the safety constraint, without interfering with the minimal violation $v_1^\star$:

$$\begin{aligned} &\underset{x,v}{\text{minimize}} \ \ \|v_2\|^2 \\ &\text{subject to } b \leq Cx - v \leq u, \\ &\qquad v_1 = v_1^\star. \end{aligned}$$

Continuing with the violations $v_3, \ldots, v_P$, a sequence of P constrained least-squares problems is solved, where objectives with lower priority are optimized as far as they do not interfere with the optimization of objectives with higher priority. Using such a sequential approach for resolving a strict hierarchy of linear inequality constraints (7) has been suggested in [7].

This sequence of constrained least-squares problems corresponds in fact to a standard lexicographic multi-objective optimization problem,

$$\begin{aligned} &\text{lex}\underset{x,v}{\text{minimize}} \ \ \left(\|v_1\|^2, \ldots, \|v_P\|^2\right) \\ &\text{subject to } b \leq Cx - v \leq u, \end{aligned} \tag{8}$$

the solution of which can be approached alternatively with a dedicated multi-objective active-set scheme, minimizing *simultaneously* all the violations $v_1, \ldots, v_P$ while preserving their hierarchy [8]. This has several advantages, especially the capacity to hot-start efficiently, what is often very useful in robotics applications. Similar to the "lexicographic simplex method" [9], this approach relies on the repeated solution of equality-constrained lexicographic least-squares problems

$$\begin{aligned} &\text{lex}\underset{x,r}{\text{minimize}} \ \ \left(\|r_1\|^2, \ldots, \|r_P\|^2\right) \\ &\text{subject to } r = Ax - y, \end{aligned} \tag{9}$$

where

$$\underbrace{\begin{bmatrix} r_1 \\ \vdots \\ r_P \end{bmatrix}}_{r} = \underbrace{\begin{bmatrix} A_1 \\ \vdots \\ A_P \end{bmatrix}}_{A} x - \underbrace{\begin{bmatrix} y_1 \\ \vdots \\ y_P \end{bmatrix}}_{y}$$

are the constraints in (7) that are assumed to be active, *i.e.*, hold as equality constraints, and r_k are residuals to be minimized. A key to efficiently solving the inequality-constrained problem (8) lies therefore in the capacity to solve efficiently the equality-constrained problem (9). This will be the focus of this paper.

Closely related to this lexicographic problem is the weighted least-squares problem

$$\begin{aligned} &\underset{x,r}{\text{minimize}} \ \ \sum_{k=1}^{P} w_k \|r_k\|^2 \\ &\text{subject to } r = Ax - y, \end{aligned} \tag{10}$$

where w_k are positive scalar weights. As demonstrated in [10] (see as well [11], p. 127), the solution of this weighted problem, provided that it is unique, converges to the solution of the lexicographic problem (9) in the limit

$$\frac{w_k}{w_{k+1}} \to \infty \text{ for } k = 1, \ldots, P-1. \tag{11}$$

As a result, this weighted problem is often used as an approximation to the lexicographic problem, even though finding appropriate weights w_k for achieving a "good" approximation is non-trivial [12]. The reason is that, in robotics, the weighted problem (10) is usually considered to be faster to solve than the problem (9) with a strict hierarchy [10], [13], [14], [15].

The main contribution of this article is to address this common misconception and demonstrate both in theory and in practice that the problem with a strict hierarchy can actually be solved faster than its weighted counterpart. Our analysis is based on the behavior of the weighted problem (10) in the limit (11), where a very particular sparse block structure is revealed. The sparsity increases with the number of hierarchical levels, hence the algorithmic complexity of the corresponding numerical scheme will decrease accordingly: the more hierarchical levels, the faster the solution will be obtained, contrary to what is usually expected in the robotics literature. This sparse block structure will be made apparent in a new matrix factorization, a "lexicographic QR", or ℓ-QR, which will be key to efficiently solving problems involving strict hierarchies.

The rest of the article is organized as follows. In Section II, we discuss alternative approaches for solving (9) from the point of view of performing a particular change of basis. In Section III, we introduce the lexicographic QR decomposition and demonstrate how it can be used to solve the lexicographic least-squares problems (9). Section IV presents both a theoretical argumentation and numerical verification that a hierarchical least squares problem can be solved faster than its weighted counterpart. Section V presents a numerical evaluation on three typical humanoid robot control problems demonstrating that, with the proposed numerical scheme, complex problems can be tackled in real-time even with limited computational resources. Section VI includes further reflections on the links between our approach and popular approaches in robotics.

II. The impact of null-space bases on linear constraints resolution

A. A standard sequence of computations

Consider the objective with highest priority in (9),

$$\underset{x}{\text{minimize}} \ \ \|A_1 x - y_1\|^2 \tag{12}$$

with a matrix $A_1 \in \mathbb{R}^{m_1 \times n}$, and introduce a change of basis, with an invertible matrix $B_1 \in \mathbb{R}^{n \times n}$, such that the vector x is partitioned as

$$x = \underbrace{\begin{bmatrix} Y_1 & Z_1 \end{bmatrix}}_{B_1} \begin{bmatrix} x_1 \\ \bar{x}_1 \end{bmatrix},$$

where the columns of $Z_1 \in \mathbb{R}^{n \times (n-p_1)}$, p_1 being the rank of A_1, form a basis of the null-space of A_1: $A_1 Z_1 = 0$. With this change of basis, the least-squares problem (12) is reformulated as:

$$\underset{x_1}{\text{minimize}} \ \|A_1 Y_1 x_1 - y_1\|^2,$$

which depends only on x_1, and has a unique solution $x_1^\star$ since the matrix $A_1 Y_1$ has full-column rank.

With this change of basis, we can consider the minimization of the next objective in (9) with the remaining variables $\bar{x}_1$, which have no impact on the first objective:

$$\underset{\bar{x}_1}{\text{minimize}} \ \|A_2 Z_1 \bar{x}_1 + A_2 Y_1 x_1^\star - y_2\|^2.$$

The solution of this second least-squares problem can be approached in a similar way, with a second change of basis B_2, partitioning $\bar{x}_1$ as

$$\bar{x}_1 = \underbrace{\begin{bmatrix} Y_2 & Z_2 \end{bmatrix}}_{B_2} \begin{bmatrix} x_2 \\ \bar{x}_2 \end{bmatrix},$$

where the columns of $Z_2 \in \mathbb{R}^{(n-p_1) \times (n-p_1-p_2)}$, p_2 being the rank of $A_2 Z_1$, form a basis of the null-space of $A_2 Z_1$: $(A_2 Z_1) Z_2 = 0$. By solving P least-squares problems in this way, one for each priority level, a minimizer of the whole lexicographic problem (9) can be obtained as

$$\begin{aligned} x^\ell &= Y_1 x_1^\star + Z_1 (Y_2 x_2^\star + Z_2 (Y_3 x_3^\star + Z_3 (\dots))) \\ &= \sum_{k=1}^{P} N_{k-1} Y_k x_k^\star, \end{aligned} \tag{13}$$

where $N_0 = I$, $N_k = N_{k-1} Z_k$ for $k = 1, \dots, P$. This sequence of computations is at the heart of many classical approaches for solving constrained least-squares problems [16], [17], the most important difference among them being the choice of basis matrices.

Apart from constructing these bases, the main computational steps in the above sequence are to form the products $A_k N_{k-1}$, and to solve the least-squares problems:

$$\underset{\bar{x}_{k-1}}{\text{minimize}} \ \|A_k N_{k-1} \bar{x}_{k-1} - \hat{y}_k\|^2 \tag{14}$$

with appropriate vectors $\hat{y}_k$. Note that, as a result of the matrix products $A_k N_{k-1}$, the least-squares problems are reformulated with respect to variables $\bar{x}_k$ of decreasing dimension: this sequence of computations can be seen as a variable reduction scheme [18].

B. Computing solutions with minimal norm

When the lexicographic problem (9) does not have a unique minimizer, it can be interesting in some cases to obtain the one with minimal Euclidean norm. This can be achieved by using orthonormal bases B_k. Since, in this case, $Y_k^T Y_k$ and $Z_k^T Z_k$ are identity matrices while $Y_k^T Z_k = 0$, one can directly conclude from (13) that (see [16], p. 190)

$$\left\|x^\ell\right\|^2 = \sum_{k=1}^{P} \|x_k^\star\|^2 .$$

This way, computing solutions $x_k^\star$ with minimal norm, leads to a minimizer x^ℓ with minimal norm. This is one of the predominant methods used in robotics. For example, the approaches proposed in [8], [19] form orthonormal bases B_k using a QR decomposition of $(A_k N_{k-1})^T$. The main drawback of this approach is the fact that orthonormal bases are generally dense (and typically stored in factored form [16], [8]), and as a result, the matrix products $A_k N_{k-1}$ end up being relatively expensive to compute.

An alternative approach is to include a final objective in the hierarchy:

$$\text{minimize} \ \|x\|^2 .$$

This way, the desired minimizer is uniquely defined, and obtained independently from the choice of bases B_k. As we will demonstrate in the sequel, being able to choose computationally attractive bases, which need not be orthonormal, leads to a much more efficient computation scheme. The same applies to cases where it is desirable to minimize a general ellipsoidal norm

$$\|x\|_M^2 = x^T M x = \left\|M^{\frac{1}{2}} x\right\|^2 ,$$

with a positive-definite symmetric matrix M, as proposed in [3], [20]: it is generally much more efficient to include explicitly a final objective

$$\text{minimize} \ \left\|M^{\frac{1}{2}} x\right\|^2 ,$$

and use computationally efficient bases B_k, instead of maintaining a minimal ellipsoidal norm at each stage of the solution process, as often proposed in the robotics literature [21], [22], [23], [24]. Furthermore, observe that if the original problem actually has a unique solution, then any choice of bases B_k would lead to the same, unique x^ℓ, and thus computationally efficient bases should be preferred. A similar observation, in a slightly different context, can be found in [25].

C. Performing variable elimination

Computationally attractive bases B_k should be easy to construct, should lead to efficient variable reduction since this involves matrix products that can be potentially very expensive to compute, and should contribute to efficiently solving the least-squares problems (14). Bases that satisfy all these requirements are presented next.

The most standard approach, both efficient and reliable, to solving (possibly rank-deficient) least-squares problems is based on the QR decomposition of the matrix $A_k N_{k-1}$:

$$\underbrace{\begin{bmatrix} Q_k^{'} & Q_k^{''} \end{bmatrix}}_{Q_k} \begin{bmatrix} R_k & S_k \\ 0 & 0 \end{bmatrix} \Pi_k^T = A_k N_{k-1}, \tag{15}$$

with an orthogonal matrix Q_k, an upper-triangular invertible matrix R_k, and a permutation matrix Π_k [26]. Let us consider then a change of basis

$$\bar{x}_{k-1} = \underbrace{\begin{bmatrix} Y_k & Z_k \end{bmatrix}}_{B_k} \begin{bmatrix} x_k \\ \bar{x}_k \end{bmatrix}$$

with

$$Y_k = \Pi_k \begin{bmatrix} R_k^{-1} \\ 0 \end{bmatrix}, \quad Z_k = \Pi_k \begin{bmatrix} -R_k^{-1} S_k \\ I \end{bmatrix}.$$

Since the matrix R_k is upper-triangular, this change of basis can be obtained with a simple backward substitution. This change of basis can be rewritten as

$$\bar{x}_{k-1} = \Pi_k \begin{bmatrix} R_k^{-1}(x_k - S_k \bar{x}_k) \\ \bar{x}_k \end{bmatrix}, \tag{16}$$

where the vector $\bar{x}_k$ appears to be a simple selection of some components of the vector $\bar{x}_{k-1}$ through the permutation matrix Π_k. For this reason, this variable reduction scheme is commonly referred to as *variable elimination* [16]. More importantly, due to this structure, the matrix products $A_k N_{k-1}$ can be performed very efficiently, what appears to be one of the keys for efficiently solving the lexicographic problem (9).

Since the orthonormal transformation Q_k^T preserves the Euclidean norm, using (15)-(16), we have

$$\|A_k N_{k-1} \bar{x}_{k-1} - \hat{y}_k\|^2 = \left\| \begin{bmatrix} x_k \\ 0 \end{bmatrix} - \begin{bmatrix} Q_k^{'T} \\ Q_k^{''T} \end{bmatrix} \hat{y}_k \right\|^2.$$

Hence, the minimum of the least-squares problem (14) is reached for $x_k^\star = Q_k^{'T} \hat{y}_k$. The corresponding solution $Y_k x_k^\star$ does not have a minimal norm. It is generally called a *basic solution*, since due to the structure of the matrix Y_k, it appears to have exactly p_k non-zero elements, p_k being the rank of the matrix $A_k N_{k-1}$. This can present advantages beyond its efficient computation, for example when using a limited number of joint velocities, contact forces, control torques, *etc.* is desirable. Note furthermore that with an appropriate column permutation strategy (see Section III-C), specific variables can be dedicated to the satisfaction of specific objectives, what can be desirable in some situations.

III. The Lexicographic QR Decomposition

Here, we introduce a factorization of the matrix A that is used to address the solution of the lexicographic problem (9). Since this factorization shares some important structural properties with standard QR decompositions, but reflects the lexicographic structure of the problem, we propose to call it "lexicographic QR" decomposition (or ℓ-QR in short):

$$\underbrace{\begin{bmatrix} Q_\ell^{'} & Q_\ell^{''} \end{bmatrix}}_{Q_\ell} \begin{bmatrix} R_\ell & T_\ell \\ 0 & 0 \end{bmatrix} \Pi^T = A, \tag{17}$$

where $\Pi \in \mathbb{R}^{n \times n}$ is a permutation matrix, $Q_\ell \in \mathbb{R}^{m \times m}$ is given by

$$Q_\ell^{'} = \begin{bmatrix} Q_1^{'} & & & \\ L_{21} & Q_2^{'} & & \\ \vdots & \vdots & \ddots & \\ L_{P1} & L_{P2} & \dots & Q_P^{'} \end{bmatrix} \in \mathbb{R}^{m \times p},$$

$$Q_\ell^{''} = diag(Q_1^{''}, \dots, Q_P^{''}) \in \mathbb{R}^{m \times (m-p)},$$

with orthogonal matrices $Q_k = \begin{bmatrix} Q_k^{'} & Q_k^{''} \end{bmatrix} \in \mathbb{R}^{m_k \times m_k}$. $R_\ell \in \mathbb{R}^{p \times p}$ is an upper-triangular invertible matrix while $T_\ell \in \mathbb{R}^{p \times (n-p)}$ is (in general) dense. The following substructure will be considered:

$$R_\ell = \begin{bmatrix} R_1 & R_1^2 & R_1^3 & \dots & R_1^P \\ & R_2 & R_2^3 & \dots & R_2^P \\ & & \ddots & & \vdots \\ & & & \ddots & \vdots \\ & & & & R_P \end{bmatrix}, \; T_\ell = \begin{bmatrix} T_1 \\ T_2 \\ \vdots \\ \vdots \\ T_P \end{bmatrix},$$

$S_k = \begin{bmatrix} R_k^{k+1} & \dots & R_k^P & T_k \end{bmatrix}$ for $k = 1, \dots, P-1$ and $S_P = T_P$.

A. Constructing the ℓ-QR decomposition

By computing a rank-revealing QR decomposition of the matrix A_1,

$$\underbrace{\begin{bmatrix} Q_1^{'} & Q_1^{''} \end{bmatrix}}_{Q_1} \begin{bmatrix} R_1 & S_1 \\ 0 & 0 \end{bmatrix} \Pi_1^T = A_1,$$

we directly obtain the first row of the decomposition, identifying $p_1 = rank(A_1)$ at the same time. Since the orthonormal transformation Q_1^T preserves the Euclidean norm, we obtain

$$\begin{aligned} \|r_1\|^2 &= \|A_1 x - y_1\|^2 \\ &= \left\| \begin{bmatrix} R_1 & S_1 \\ 0 & 0 \end{bmatrix} \Pi_1^T x - \begin{bmatrix} Q_1^{'T} \\ Q_1^{''T} \end{bmatrix} y_1 \right\|^2. \end{aligned}$$

Hence, the minimal norm of the residual r_1 is reached for any x such that

$$\begin{bmatrix} R_1 & S_1 \end{bmatrix} \Pi_1^T x = Q_1^{'T} y_1 = x_1^\star \in \mathbb{R}^{p_1},$$

with solutions that can be put in the form

$$x = \underbrace{\Pi_1 \begin{bmatrix} R_1^{-1} \\ 0 \end{bmatrix}}_{Y_1} x_1^\star + \underbrace{\Pi_1 \begin{bmatrix} -R_1^{-1} S_1 \\ I \end{bmatrix}}_{Z_1} \bar{x}_1, \tag{18}$$

where $Y_1 x_1^\star$ is a basic solution, having exactly p_1 non-zero elements that have been selected to ensure a minimal norm of the residual r_1. The remaining $n - p_1$ elements can be chosen freely with $\bar{x}_1 \in \mathbb{R}^{n-p_1}$. The columns of the matrix $Z_1 \in \mathbb{R}^{n \times (n-p_1)}$ form a basis for the null-space of A_1.

Considering solutions of the form (18) for the second hierarchical level, leads to a residual of the form

$$r_2 = A_2 \begin{bmatrix} Y_1 & Z_1 \end{bmatrix} \begin{bmatrix} x_1^\star \\ \bar{x}_1 \end{bmatrix} - y_2,$$

where only $n - p_1$ elements of the vector x remain to be chosen, through $\bar{x}_1$. We can proceed as before and consider a rank-revealing QR decomposition

$$\underbrace{\begin{bmatrix} Q_2^{'} & Q_2^{''} \end{bmatrix}}_{Q_2} \begin{bmatrix} R_2 & S_2 \\ 0 & 0 \end{bmatrix} \Pi_2^T = A_2 Z_1$$

from which $p_2 = rank(A_2 Z_1)$ is identified. Combining the permutation matrices,

$$\tilde{\Pi}_2 = \Pi_1 \begin{bmatrix} I & 0 \\ 0 & \Pi_2 \end{bmatrix} \in \mathbb{R}^{n \times n},$$

and inserting p_1 columns of zeros at the beginning, we obtain

$$\begin{aligned} Q_2 \begin{bmatrix} 0 & R_2 & S_2 \\ 0 & 0 & 0 \end{bmatrix} \tilde{\Pi}_2^T &= A_2 \begin{bmatrix} 0 & Z_1 \end{bmatrix} \Pi_1^T, \\ &= A_2 \Pi_1 \left(I - \begin{bmatrix} R_1^{-1} \\ 0 \end{bmatrix} \begin{bmatrix} R_1 & S_1 \end{bmatrix} \right) \Pi_1^T, \\ &= A_2 - A_2 Y_1 \begin{bmatrix} R_1 & S_1 \end{bmatrix} \Pi_1^T. \end{aligned}$$

With $L_{21} = A_2 Y_1$, we obtain the second row of the decomposition (17).

Since the orthonormal transformation Q_2^T preserves the Euclidean norm, we obtain

$$\begin{aligned} \|r_2\|^2 &= \|A_2 Z_1 \bar{x}_1 - (y_2 - A_2 Y_1 x_1^\star)\|^2 \\ &= \left\| \begin{bmatrix} 0 & R_2 & S_2 \\ 0 & 0 & 0 \end{bmatrix} \tilde{\Pi}_2^T x - \begin{bmatrix} Q_2^{'T} \\ Q_2^{''T} \end{bmatrix} (y_2 - L_{21} x_1^\star) \right\|^2. \end{aligned}$$

Hence, the minimal norm of the residual r_2 is reached for any x such that

$$\begin{bmatrix} 0 & R_2 & S_2 \end{bmatrix} \tilde{\Pi}_2^T x = Q_2^{'T} (y_2 - L_{21} x_1^\star) = x_2^\star,$$

with solutions that can be put in the form

$$x = Y_1 x_1^\star + Z_1 \underbrace{\Pi_2 \begin{bmatrix} R_2^{-1} \\ 0 \end{bmatrix}}_{Y_2} x_2^\star + Z_1 \underbrace{\Pi_2 \begin{bmatrix} -R_2^{-1} S_2 \\ I \end{bmatrix}}_{Z_2} \bar{x}_2,$$

where p_2 elements of the vector x have been selected to ensure a minimal norm of the residual r_2, while $n - p_1 - p_2$ elements remain to be chosen through $\bar{x}_2 \in \mathbb{R}^{n - p_1 - p_2}$.

Continuing this sequence of rank-revealing QR decompositions and variable eliminations, we can conclude that the minimal norm of the residual r_k for all $k = 1, \ldots, P$ is reached for any x such that

$$\begin{bmatrix} 0 & R_k & S_k \end{bmatrix} \tilde{\Pi}_k^T x = Q_k^{'T} \left(y_k - \sum_{j=1}^{k-1} L_{kj} x_j^\star \right) = x_k^\star, \quad (19)$$

where $L_{kj} = A_k N_j Y_j$, and the whole lexicographic QR decomposition is eventually obtained, as formalized in Algorithm 1.

B. Solving problem (9) using the decomposition (17)

Let us introduce two vectors $x^\star = (x_1^\star, \ldots, x_P^\star) \in \mathbb{R}^p$ and $\rho^\star = (\rho_1^\star, \ldots, \rho_P^\star) \in \mathbb{R}^{m-p}$ where

$$\begin{bmatrix} x_k^\star \\ \rho_k^\star \end{bmatrix} = \begin{bmatrix} Q_k^{'T} \\ Q_k^{''T} \end{bmatrix} \left(y_k - \sum_{j=1}^{k-1} L_{kj} x_j^\star \right). \quad (20)$$

One can easily verify that

$$Q_\ell \begin{bmatrix} x^\star \\ \rho^\star \end{bmatrix} = y. \quad (21)$$

Obtaining $x^\star$ and $\rho^\star$ from (20) appears to be an efficient way of solving (21) with a form of (block-wise) forward substitution, adapted to the specific structure of the matrix Q_ℓ. Note that by factorizing an extended matrix $\begin{bmatrix} A & y \end{bmatrix}$, one obtains directly $(x^\star, \rho^\star)$ in the last column.

Combining the equations (19) for all $k = 1, \ldots, P$ we can conclude that the lexicographically minimal norm of the residual r^ℓ is reached for any x such that

$$\begin{bmatrix} R_\ell & T_\ell \end{bmatrix} \Pi^T x = x^\star,$$

with solutions that can be put in the form

$$x^\ell = \underbrace{\Pi \begin{bmatrix} R_\ell^{-1} \\ 0 \end{bmatrix}}_{Y_\ell} x^\star + \underbrace{\Pi \begin{bmatrix} -R_\ell^{-1} T_\ell \\ I \end{bmatrix}}_{Z_\ell} \bar{x}$$

for some $\bar{x} \in \mathbb{R}^{n-p}$. When $p = n$, there is a unique solution. When $p < n$, a basic solution is obtained, having exactly p non-zero components, by choosing $\bar{x} = 0$. It can be directly expressed as

$$x^\ell = \underbrace{\Pi \begin{bmatrix} R_\ell^{-1} & 0 \\ 0 & 0 \end{bmatrix} Q_\ell^{-1}}_{A^\ell} y \quad (22)$$

where it is easy to verify that the matrix $A^\ell \in \mathbb{R}^{n \times m}$ is a generalized inverse[1] of A, while the columns of $Z_\ell \in \mathbb{R}^{n \times (n-p)}$ form a basis for its null-space.

Note that the matrix R_ℓ is upper-triangular, so once the decomposition (17) is available, the solution x_ℓ can be obtained very efficiently with a (block-wise) forward substitution in (20), followed by a backward substitution in (22).

Finally, the (unique) lexicographically minimal residual is simply given by

$$\begin{aligned} r^\ell = Ax^\ell - y &= \left(AA^\ell - I \right) y = Q_\ell \left(\begin{bmatrix} I & 0 \\ 0 & 0 \end{bmatrix} - I \right) Q_\ell^{-1} y \\ &= \begin{bmatrix} Q_\ell^{'} & Q_\ell^{''} \end{bmatrix} \begin{bmatrix} 0 & 0 \\ 0 & -I \end{bmatrix} \begin{bmatrix} x^\star \\ \rho^\star \end{bmatrix} = -Q_\ell^{''} \rho^\star. \end{aligned}$$

The algorithm that we have just described appears to be a generalization of the method of direct elimination commonly used for solving equality-constrained least-squares problems [16], [17].

[1] More precisely, $AA^\ell A = A$, $A^\ell A A^\ell = A^\ell$, $(AA^\ell)^T = AA^\ell$ [11].

C. Implementation

Algorithm 1 presents an approach for constructing the lexicographic QR decomposition of A "in place", that is, A is overwritten by its factors Q_ℓ, R_ℓ, T_ℓ. $A_j^{(k)}$ will be used to denote the last $n - \sum_{i=1}^{k-1} p_i$ columns of A_j. Let us partition it as

$$A_j^{(k)} = \begin{bmatrix} \bar{A}_j^{(k)} & \bar{\bar{A}}_j^{(k)} \end{bmatrix},$$

where $\bar{A}_j^{(k)} \in \mathbb{R}^{m_k \times p_k}$.

Algorithm 1: Lexicographic QR decomposition

Data: $A = (A_1, \ldots, A_P)$
Result: $\{Q_\ell, R_\ell, T_\ell, \Pi\}$
▷ initialize $\Pi \leftarrow I$
for $k = 1$ **to** P **do**

Rank-revealing QR decomposition

▷ form the QR decomposition of $A_k^{(k)}$ "in place"

$$A_k^{(k)} = Q_k \begin{bmatrix} R_k & S_k \\ 0 & 0 \end{bmatrix} \Pi_k^T,$$

that is, $A_k^{(k)} \leftarrow \begin{bmatrix} R_k & S_k \\ 0 & 0 \end{bmatrix}$, while Q_k is stored below R_k (as p_k Householder transformations)

▷ $p_k \leftarrow$ rank of R_k

▷ permute columns

$$\begin{bmatrix} A_{k+1}^{(k)} \\ \vdots \\ A_P^{(k)} \end{bmatrix} \leftarrow \begin{bmatrix} A_{k+1}^{(k)} \\ \vdots \\ A_P^{(k)} \end{bmatrix} \Pi_k$$

▷ accumulate permutations $\Pi \leftarrow \Pi\, diag(I, \Pi_k)$

Variable elimination

▷ form the Schur-complement (in two steps)

$$\begin{bmatrix} \bar{A}_{k+1}^{(k)} \\ \vdots \\ \bar{A}_P^{(k)} \end{bmatrix} \leftarrow \begin{bmatrix} \bar{A}_{k+1}^{(k)} \\ \vdots \\ \bar{A}_P^{(k)} \end{bmatrix} R_k^{-1}, \tag{23}$$

$$\begin{bmatrix} \bar{\bar{A}}_{k+1}^{(k)} \\ \vdots \\ \bar{\bar{A}}_P^{(k)} \end{bmatrix} \leftarrow \begin{bmatrix} \bar{\bar{A}}_{k+1}^{(k)} \\ \vdots \\ \bar{\bar{A}}_P^{(k)} \end{bmatrix} - \begin{bmatrix} \bar{A}_{k+1}^{(k)} \\ \vdots \\ \bar{A}_P^{(k)} \end{bmatrix} T_k. \tag{24}$$

▷ A has been overwritten with the factorization (17).

The use of column permutations in Algorithm 1 makes the QR decomposition at the k-th stage *rank-revealing*, that is, p_k is identified. Loosely speaking, p_k is the smallest number of variables that have to be dedicated to the k-th hierarchical level, so that obtaining a minimal $\|r_k\|^2$ is guaranteed for any y_k. Apart form identifying p_k, column permutations are performed with the aim of constructing a factorization in which the matrices R_k are as well conditioned as possible. Or in other words, finding the p_k variables "best suited" to the k-th hierarchical level. The reason is that the use of R_k^{-1} during the elimination step could lead to inaccurate results if R_k is ill-conditioned. We rely on a classical pivoting strategy, where the column of $A_k^{(k)}$ with largest Euclidean norm is selected to choose a pivot from [16], p. 103.

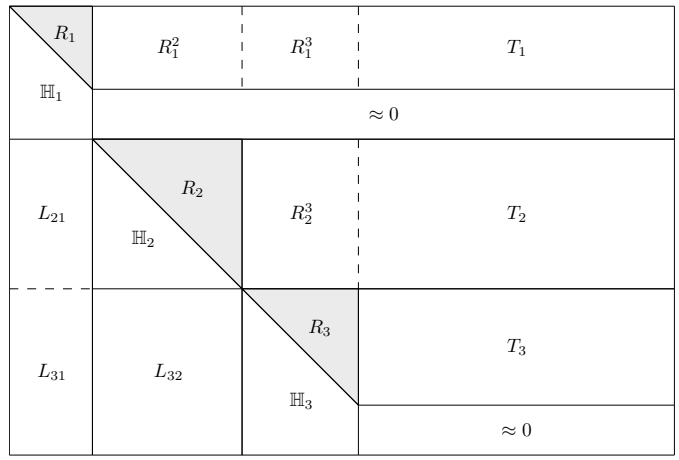


Fig. 1. Storage of the lexicographic QR decomposition (for $P = 3$ hierarchical levels).

Figure 1 depicts an example with three hierarchical levels, where $m_1 < p_1$, $m_2 = p_2$, $m_3 < p_3$. That is, A_1 is singular, A_2 is full row rank (and shares no linearly dependent equations with A_1), while A_3 is either singular or shares linearly dependent equations with A_1 and/or A_2. Note that $p < n$, *i.e.*, infinitely many solutions exist. $\mathbb{H}_k$ stores (the vector part of) the p_k Householder transformations applied during the QR decomposition at stage k.

IV. The lexicographic problem is faster to solve than the weighted problem

In robotics (as well as in other engineering fields), the lexicographic least-squares problem (9) has always been considered as expensive to solve [15]. As a result, it is not uncommon [10], [13], [14] to approximate it with the weighted problem (10), which can be reformulated as:

$$\underset{x}{\text{minimize}} \;\; \|Ax - y\|_W^2 = (Ax - y)^T W (Ax - y) \tag{25}$$

with $W = diag(w_1 I_{m_1}, \ldots, w_P I_{m_P}) \in \mathbb{R}^{m \times m}$, $m = \sum_{k=1}^{P} m_k$. Here, we demonstrate both in theory and in practice that this weighted problem is in fact more expensive to solve than the lexicographic problem (9).

A. Structure of W-invariant matrices in the limit (11)

The most common approach to solving the least-squares problem (25) relies on computing a rank-revealing QR decomposition [26]. As shown in [27], [28], the analysis of this

decomposition in the limit (11) simplifies by considering more precisely a weighted QR decomposition

$$\underbrace{\begin{bmatrix} Q_w^{'} & Q_w^{''} \end{bmatrix}}_{Q_w} \begin{bmatrix} R_w & T_w \\ 0 & 0 \end{bmatrix} \Pi^T = A, \tag{26}$$

where Q_w (and Q_w^{-1}) is an orthogonal matrix with respect to the weighted norm, and hence preserves it:

$$\|Ax - y\|_W^2 = \left\|Q_w^{-1}(Ax - y)\right\|_W^2 .$$

Such matrices can be recognized to be "W-invariant", *i.e.*, invertible and satisfying $Q_w^T W Q_w = W$ and, as shown in [27], they exhibit a very particular sparse block structure in the limit (11). In fact, by using the technique presented in [29], it is easy (even though a bit tedious) to demonstrate that in the limit (11),

$$Q_w^{'} \to Q_\ell^{'}, \quad \begin{bmatrix} R_w & T_w \end{bmatrix} \to \begin{bmatrix} R_\ell & T_\ell \end{bmatrix}. \tag{27}$$

All known approaches to construct QR decompositions [28], [30] are based on the combination of W-invariant matrices G_i, chosen such that the product

$$\underbrace{G_k^{-1} \dots G_1^{-1}}_{Q_w^{-1}} A = \begin{bmatrix} R_w & T_w \\ 0 & 0 \end{bmatrix} \Pi^T,$$

leads to an upper triangular, invertible matrix R_w. The sparse block structure of these matrices in the limit (11) can be leveraged to significantly reduce the computation time of this product, which is the most expensive operation in constructing QR decompositions. Algorithm 1 is nothing but a specific implementation of this approach. A flops-count comparison can be found in [31].

In summary, in the limit (11), the weighted QR decomposition (26) converges to the lexicographic QR decomposition (17) (in the sense of (27)), while the solution of the weighted problem (10) converges to the solution of the lexicographic problem (9) (provided that it is unique). More importantly, the weighted norm becomes singular, and as a result, the W-invariant matrices involved in computing the QR decompositions and solving the corresponding least-squares problems develop a sparse block structure, which can be leveraged to significantly reduce computation time. The lexicographic problem, with a strict hierarchy, ends up being significantly faster to solve than the weighted problem.

Let us illustrate these observations on a simple 1D example:

$$\underset{x}{\text{minimize}} \quad \left\| \begin{bmatrix} a_1 \\ a_2 \end{bmatrix} x - \begin{bmatrix} y_1 \\ y_2 \end{bmatrix} \right\|_W^2$$

with $a_1, a_2 \neq 0$ and $W = \text{diag}(w_1, w_2)$ positive-definite. The corresponding weighted QR decomposition is given by

$$Q_w = \frac{1}{\sigma} \begin{bmatrix} a_1 & -\frac{w_2}{w_1} a_2 \\ a_2 & a_1 \end{bmatrix}, \quad R_w = \begin{bmatrix} \sigma \\ 0 \end{bmatrix}, \tag{28}$$

with $\sigma = \sqrt{a_1^2 + \frac{w_2}{w_1} a_2^2}$. In the limit $\frac{w_1}{w_2} \to \infty$,

$$Q_w \to Q_\ell = \begin{bmatrix} 1 & 0 \\ \frac{a_2}{a_1} & 1 \end{bmatrix}.$$

We can observe in this specific example that in the limit, the weighted QR decomposition (28) becomes an LU decomposition [32], a form of Gaussian elimination which is generally considered to be the fastest way to solve a system of linear equations (in the non-singular case) [17], [26], significantly faster than with a QR decomposition.

Note finally that the solution of the lexicographic problem can be obtained not only more efficiently but also more reliably than the solution of the weighted problem since large gaps between the weights may lead to numerical problems [33].

Remark 1: The weighted QR decomposition (26) can also be obtained from a standard QR decomposition of the weighted matrix A:

$$\begin{bmatrix} Q^{'} & Q^{''} \end{bmatrix} \begin{bmatrix} R & T \\ 0 & 0 \end{bmatrix} \Pi^T = W^{\frac{1}{2}} A.$$

We simply have

$$Q_w^{'} = W^{-\frac{1}{2}} Q^{'} W_p^{\frac{1}{2}}, \quad \begin{bmatrix} R_w & T_w \end{bmatrix} = W_p^{-\frac{1}{2}} \begin{bmatrix} R & T \end{bmatrix},$$

where $W_p = diag(w_1 I_{p_1}, \dots, w_P I_{p_P}) \in \mathbb{R}^{p \times p}$.

B. Numerical validation

Here, we present numerical results on randomly generated problems, verifying in practice that the lexicographic problem (9) can be solved faster than the weighted problem (10). All tests are performed on an Intel Core 2 Duo CPU (2.26 GHz, P8400), similar to the CPU embedded on HRP-2 robots [34] (using `g++ 4.6.3` with `-O3` optimization under UBUNTU [35]). This is a slow CPU by current standards, as more recent mobile Intel CPUs are typically 3 times faster. This choice is intended to show that large problems can be solved very efficiently even with limited CPU resources. Our `C++` implementation is based on Eigen [36], and in the comparisons we use Eigen's QR (`ColPivHouseholderQR`) and LU (`PartialPivLU`) decompositions with partial pivoting.

First, we compare the computation time needed to solve the lexicographic problem (9) and the weighted problem (10) when A is a full-rank rectangular matrix. Figure 2 depicts a case with $n = 128$ and varying m. When $m \leq n$, a basic solution is computed. In the weighted case, the algorithmic complexity of computing a QR decomposition is approximately $O(mn^2)$ [26]. The computation time appears on this figure to grow linearly with m, as expected. In the lexicographic case, with hierarchical levels of constant size ($m_1 = \dots = m_P \in \{2, 4, 8, 16\}$), Algorithm 1 can be seen to be up to three times faster, while on problems of bigger sizes we have observed even bigger ratios. The reason for having less efficient computations when $m_1 = \dots = m_P = 2$ is due to blocking effects and will be discussed towards the end of this section.

In case the matrix A is square ($m = n$), if we know in advance that it is not singular (no conflicts among its equations), it has a unique solution, which can be obtained using an LU decomposition with partial pivoting (a form of Gaussian elimination). This is generally considered as the fastest way to solve a square and non-singular system of linear equations. We can observe on Fig. 2 that computing this

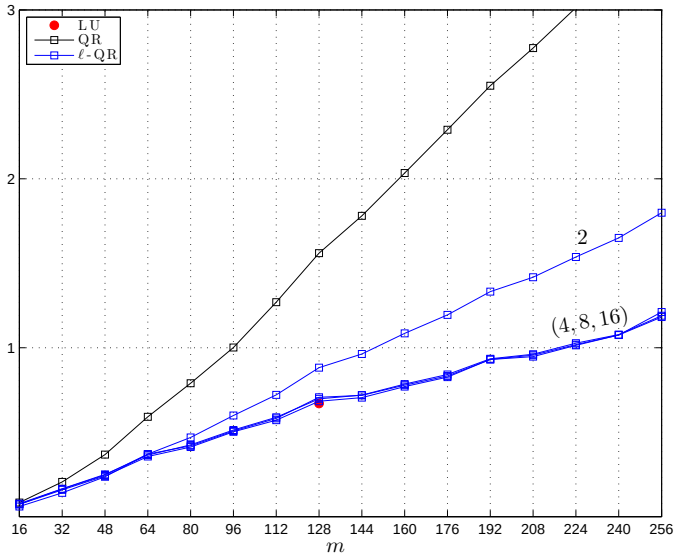


Fig. 2. Computation time in ms with $n = 128$ variables. For ℓ-QR, $m_1 = \cdots = m_P \in \{2, 4, 8, 16\}$ and P is increasing with the total number of constraints m. The red dot indicates the computation time when LU decomposition is used (which is possible only for $n = m = 128$).

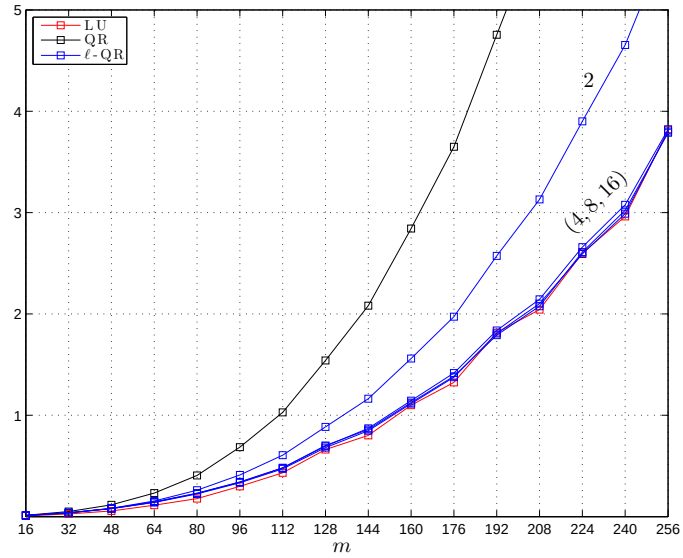


Fig. 3. Computation time in ms with $n = m$. For ℓ-QR, $m_1 = \cdots = m_P \in \{2, 4, 8, 16\}$ and P is increasing with m.

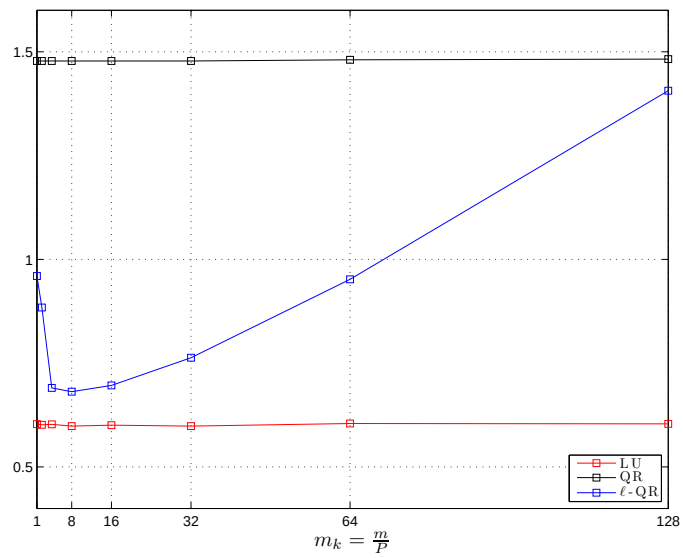


Fig. 4. Computation time in ms with $n = m = 128$. For ℓ-QR, $P = \langle 2^7, \ldots, 2^0 \rangle$, $m_1 = \cdots = m_P$.

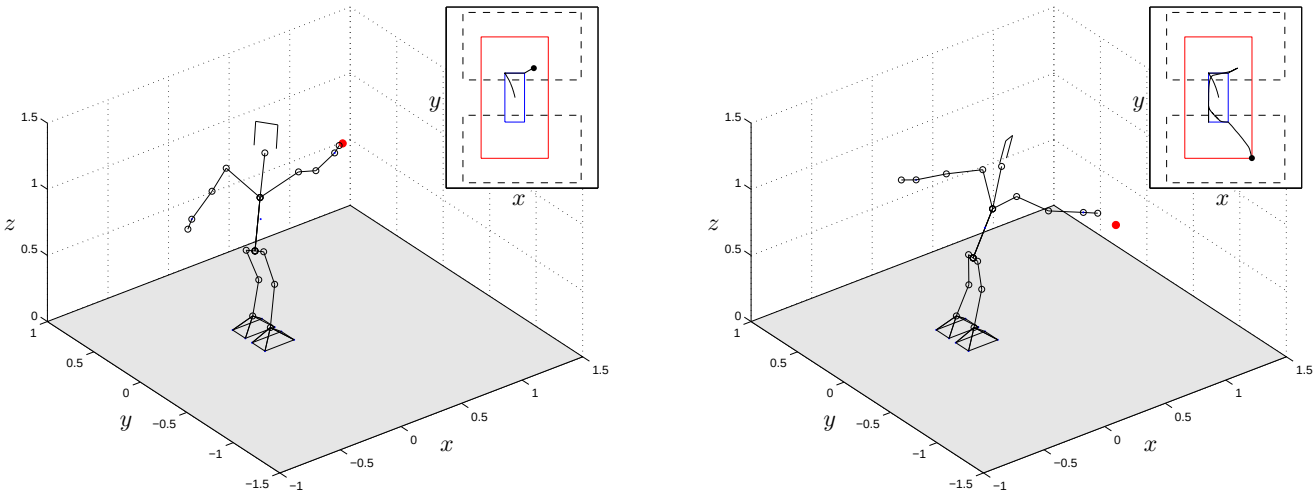
solution through the lexicographic problem (9), not relying on the system being non singular, is nearly as fast. This demonstrates that computing lexicographic solutions with the method proposed in this paper is very efficient, and that there is not much room for speed improvement. We can observe in Fig 3 that this very good performance is obtained consistently for all problem sizes, and that even with this slow CPU, problems having as many as 150 variables and constraints can be solved in under 1 ms.

Let us focus on the case $m = n = 128$ as a typical example, when A is square and non-singular (no conflicts among its equations): there is a unique solution, regardless of the presence or absence of hierarchy levels. We can analyze therefore more precisely the impact that introducing hierarchy levels can have on computation time. Let us vary the number of hierarchy levels P and adapt the size of the levels accordingly: $m_1 = \cdots = m_P = m/P$. When $P = 1$, there is no real hierarchy, and the lexicographic QR decomposition is identical to a standard QR decomposition. We can see in Fig. 4 that in this case, the computation time of the standard and lexicographic QR decompositions are comparable, as expected. When the least-squares problem is split into more and more hierarchy levels, computation time decreases, approaching that of LU decomposition, which is the fastest method applicable to the square and non-singular case.

For larger problems, *e.g.*, $n = 256$, we have observed that the lexicographic QR decomposition is as fast to compute as an LU decomposition when the size of the hierarchy levels is between 4 and 32, but slightly degrades for smaller hierarchy levels because of an incomplete implementation of blocking methods. Note that blocking is a well-known optimization technique for improving the use of memory structures. Instead of operating on a row or column of an array, blocked implementations operate on sub-matrices (or blocks) so that faster computations are achieved [37]. The effects of blocking are sensitive to the size of the blocks as well as to the stride of data accesses. That is why different performance may be observed for different matrix and block sizes. In our implementation of Algorithm 1 we leverage blocking only during the elimination steps (which is greatly facilitated by our choice of null-space basis Z_k).

The preceding discussion suggests that if a set of equations, such as the equation of motion of a robot (1), is known in advance to be non-singular (no conflicts among its equations), introducing an artificial hierarchy to limit each hierarchy levels to sizes between 4 and 32 can lead to a significant improvement in computation time.

Here, we have demonstrated experimentally that the lexicographic problem (9) can be solved significantly faster than



(a) (b)

Fig. 5. Snapshots from the simulation with hierarchy **A**. The polygons $\mathcal{P}_b$ and $\check{\mathcal{P}}_b$ are depicted in red and blue, respectively. The target object is represented as a red dot, while the profile of $c^{x,y}$ is depicted in black on the $x-y$ plane.

the weighted problem (10), on the contrary to what is usually considered in robotics. We have even reached the unexpected conclusion that introducing hierarchy levels can be a valid approach to significantly reduce computation time.

Note that in all cases discussed so far, no singularities were present. In general, having linearly dependent equations in the hierarchy leads to even slightly faster computation.

Remark 2: During the "QR step" at the k-th stage of Algorithm 1, p_k variables are dedicated to the minimization of the k-th objective. In our implementation, these variables are directly eliminated from all remaining levels $k+1,\dots,P$. This might not be the most efficient strategy in general. For example, when p_k is small, *e.g.*, $p_k = 2$, during the elimination step the peak CPU performance might not be reached. This could be addressed by performing delayed elimination, that is, the p_k variables could be eliminated only from the $k+1$-st hierarchical level, and after the "QR step" at stage $k+1$, the $p_k + p_{k+1}$ variables could be eliminated from the remaining hierarchical levels jointly (or again a delayed elimination could be performed if $p_k + p_{k+1}$ is considered to be small). This strategy resembles the Crout's method for LU decomposition [17].

V. Typical robotics applications

Here, we include three representative examples adopted from the recent robotics literature, with minor modifications. Our aim is to illustrate the key ideas presented in this article. Some details related to the examples are omitted as they can be found int the original references. In all tests we utilize the model of the HRP-2 humanoid robot [34], with 36 DoF, 6 of which are not actuated. In order to emphasize this, it is common to partition the Lagrangian dynamics of the robot as

$$\begin{bmatrix} H_1 \\ H_2 \end{bmatrix} \ddot{q} + \begin{bmatrix} h_1 \\ h_2 \end{bmatrix} = \begin{bmatrix} \tau \\ 0 \end{bmatrix} + \begin{bmatrix} J_{c_1} \\ J_{c_2} \end{bmatrix}^T f, \tag{29}$$

where the bottom 6 equations ($H_2 \in \mathbb{R}^{6\times 36}$) can be interpreted as the Newton-Euler equations for the system as a whole [38], [39]. A 200 Hz control sampling rate is used in all examples. All computations are performed on the same (slow) CPU as in the previous Section.

A. Velocity based control

Our first example is adopted from [8]. The robot is commanded to grasp a point object with its right hand while the feet are constrained to remain in contact with the flat ground. Naturally, the robot should respect joint position and velocity limits as well as preserve its balance. Balance preservation is modeled by restricting $c^{x,y} + \gamma \dot{c}^{x,y}$ to remain inside a properly chosen support polygon $\mathcal{P}_b$, where $c^{x,y}$ are the x and y coordinates of the Center of Mass (CoM) of the robot, and γ is a time constant that can be tuned by the user (following for example a *capturability*-based analysis [40]). In order to improve robustness to modeling uncertainties, it is advantageous to consider a more restrictive polygon $\check{\mathcal{P}}_b$ if possible. In addition, it is desirable to keep the target object as much as possible within the field-of-view of the robot (defined as a cone emanating from a given reference point on the head). Finally, it can be interesting to minimize joint speed. The following hierarchy is defined accordingly in [8]:

Hierarchy **A**

1) 30 joint limits: $\underline{\dot{q}} \le \dot{q} \le \bar{\dot{q}}$,
2) 12 equality constraints for feet contact: $J_c \dot{q} = 0$,
3) 2 inequality constraints for balance:
$$c^{x,y} + \gamma \dot{c}^{x,y} \in \mathcal{P}_b, \tag{30}$$
4) 3 equality constraints for the hand task, tracking the target object,
5) 1 inequality constraint for the field-of-view task,
6) 2 inequality constraints for improved balance robustness:
$$c^{x,y} + \gamma \dot{c}^{x,y} \in \check{\mathcal{P}}_b, \tag{31}$$

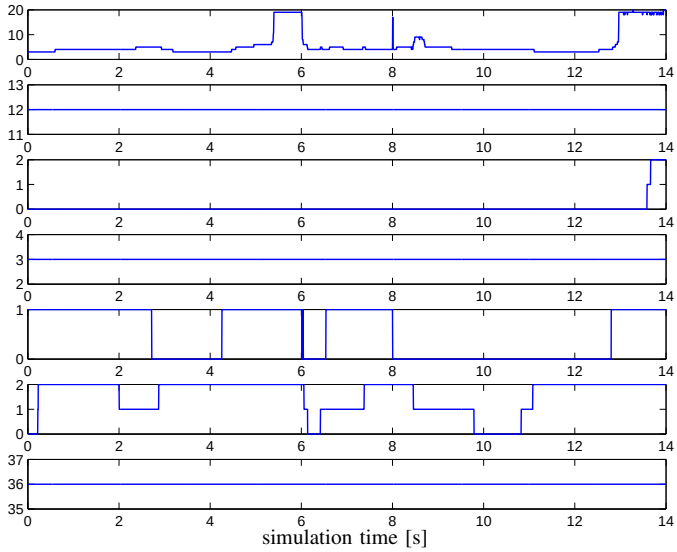


Fig. 6. Number of active constraints (ordinates) per hierarchical level at the solution of each problem in the simulation with hierarchy **A**. The topmost and lowermost subplots correspond to the levels with highest and lowest priority, respectively.

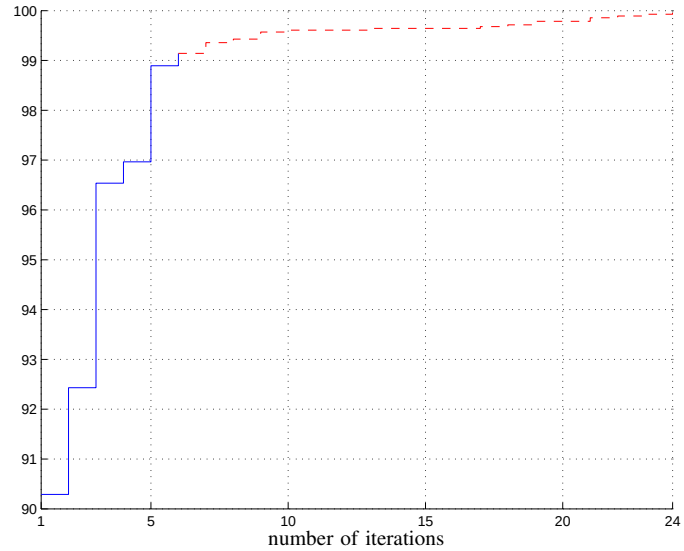


Fig. 7. Percentage of sampling periods where hierarchy **A** is solved within a given number of iterations. The number of iterations can be limited to 6 without influencing the motion of the robot in a noticeable way.

7) 36 optional equality constraints to drive joint speed to zero, either through $\dot{q} = 0$ or $H^{\frac{1}{2}}\dot{q} = 0$.

All constraints are stated in terms of the robot velocity $\dot{q} \in \mathbb{R}^{36}$, which is the decision variable in this problem. The first, third, fifth and sixth hierarchical levels involve inequality constraints. Note that each double-sided inequality is counted as one constraint only. For example, since $\mathcal{P}_b$ is a rectangular polygon, the constraint (30) can be defined as two double-bounded inequalities.

The target object moves to several locations, according to a known pattern. It starts in front of the robot, at a position that can be reached easily using the right hand. Next, it moves to the left of the robot (Fig. 5 (a)), where it cannot be grasped without violating the constraint (31). Note that the hand task has higher priority than constraint (31). The final position of the target object is depicted in Fig. 5 (b). It is unreachable without violating the constraints involved in both levels five and six. More details can be found in [8].

In order to solve this inequality-constrained hierarchy, we adopt the multi-objective active set method proposed in [8], which relies on the repeated solution of equality-constrained problems (9). This method requires computing Lagrange multipliers corresponding to all P objectives with respect to all m constraints. These can be obtained efficiently with the proposed ℓ-QR decomposition, as shown in the Appendix. Statistics regarding this algorithm can be found in Fig. 7. Note that only one equality-constrained problem needs to be solved 90 % of the time, while no more than 6 need to be solved 99 % of the time. We have observed that limiting the number of iterations of this algorithm to 6 has a negligible effect on this simulation.

In order to directly measure the influence of the choice of bases B_k (following the discussion in Section II), we extract a sequence of representative equality-constrained hierarchical problems on which the proposed ℓ-QR decomposition will be compared with the Hierarchical Complete Orthogonal Decomposition (HCOD) introduced in [8], which involves orthonormal bases and can be considered as a state-of-the-art tool in robotics for solving hierarchical problems. The number of active constraints at the solution for each hierarchical level and at each sampling time is shown in Fig. 6. We use these equality-constrained problems to perform the desired comparison.

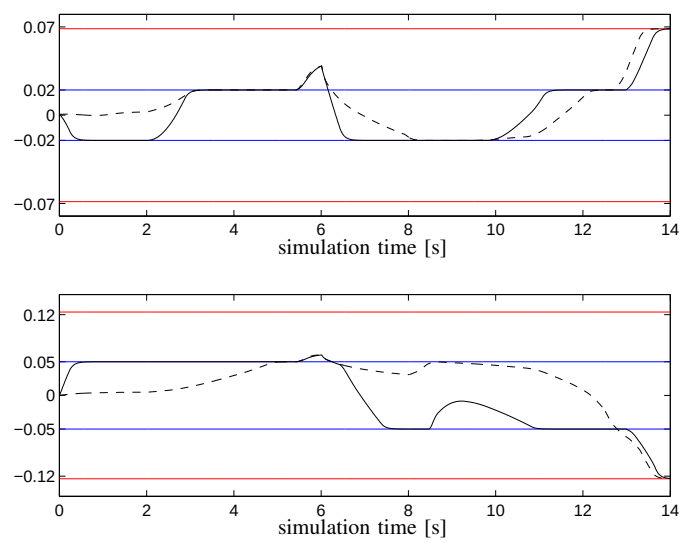
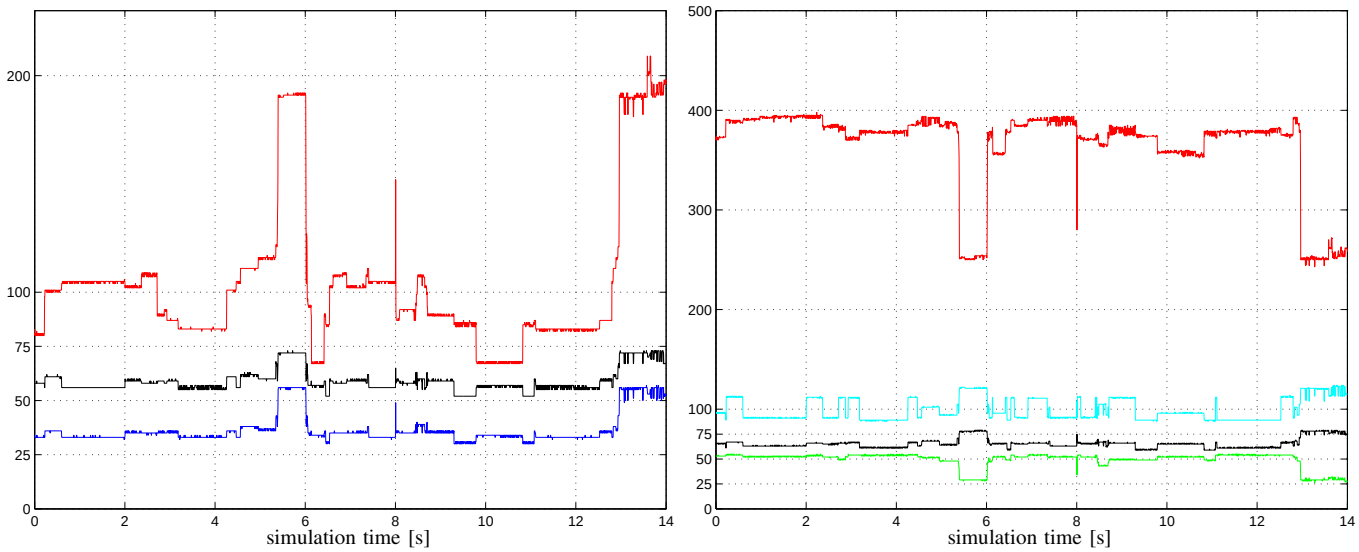


Fig. 8. Evolution of the position of the CoM c^x (top) and c^y (bottom) when using hierarchy **A** and computing solutions with minimal Euclidean norm (solid black) or with minimal kinetic energy (dashed black). The polygons $\mathcal{P}_b$ and $\tilde{\mathcal{P}}_b$ are depicted in red and blue, respectively.

Computation time will be measured in three cases. In the first case, the Euclidean norm $\|\dot{q}\|^2$ of joint speed is minimized through a final objective $\dot{q} = 0$. In the second case, the kinetic energy $\frac{1}{2}\dot{q}^T H \dot{q}$ of the robot is minimized instead,



(a) (b)

Fig. 9. Computation time in μs for equality-constrained hierarchical problems extracted from hierarchy **A**. Figure (a) depicts a comparison between HOCD (red) and ℓ-QR (black) when computing solutions with minimal Euclidean norm. The time to compute a basic solution using ℓ-QR is depicted in blue. Figure (b) depicts a comparison between HOCD (red) and ℓ-QR (black) when computing solutions minimizing the kinetic energy $\frac{1}{2}\dot{q}^T H \dot{q}$. Exploiting the structure of simple bounds with the ℓ-QR decomposition for the same problem leads to a decrease in computation time up to three times (depicted in green). The time to solve a weighted least-squares problem with the same constraints (by using a QR decomposition) is depicted in cyan.

through a final objective $H^{\frac{1}{2}}\dot{q} = 0$. We can see in Fig. 8 that this naturally leads to less abrupt motion of the CoM, since the motion of heavy parts of the robot is reduced (energy consumption is reduced by 30 % as a result). In the third case, the norm of the solution is not minimized, and a basic solution is considered.

We can see in Fig. 9 (a) that the fastest option by far (in blue) is to compute a basic solution with the proposed ℓ-QR decomposition. Minimizing the Euclidean norm with the same decomposition requires significantly more computations (in black), but still consistently less than with the HCOD (in red), even though the HCOD provides a solution with minimal Euclidean norm by construction and objective 7 can be omitted. We can see in Fig. 9 (b) that minimizing the kinetic energy instead of the Euclidean norm leads to a large increase (up to 300 %) when using the HCOD (in red), since in this case, objective 7 has to be included, to minimize explicitly $\frac{1}{2}\dot{q}^T H \dot{q}$. In contrast, the increase in computation time is small (10 %) with the proposed ℓ-QR decomposition (in black). Approaching this hierarchy of equality constraints as a single weighted least-squares problem, solved with Eigen's QR decomposition `ColPivHouseholderQR`, leads to consistently slower computations (in cyan), in accordance with the results from the previous Section. Finally, note that joint limits (at the first level of the hierarchy) can be treated explicitly as simple bounds: when active, the corresponding variables can be eliminated without the need to perform a QR decomposition. As a result, computation time with the proposed decomposition can decrease up to three times (in green). In this case, computation time appears to be consistently 7 to 8 times faster than with the HCOD.

As discussed in Section II, using orthonormal bases naturally leads to computing solutions with minimal Euclidean norm, without having to introduce a final objective such as objective 7 in Hierarchy A. When the Euclidean norm does not need to be minimized, we clearly observe in the previous results that this introduces very expensive (unnecessary) computations. But even when looking for a solution with minimal Euclidean norm, we have seen that the HCOD, using orthonormal bases, is outperformed by the approach we propose. In this regard, solution strategies relying on bases B_k that implicitly minimize a particular norm, with the aim of reducing the number of hierarchical levels, appear to have limited applications.

B. Acceleration based control

Our second example is adopted from [41]. The robot performs stepping motions in place using the following hierarchy:

Hierarchy **B**

1) 30 joint torque limits:

$$-\tau_{max} \leq H_1\ddot{q} + h_1 - J_{c_1}^T f \leq \tau_{max},$$

6 equations of motion (Newton-Euler):

$$H_2\ddot{q} + h_2 = J_{c_2}^T f, \tag{32}$$

2) $6n_c$ equality constraints for feet contact:

$$J_c\ddot{q} + \dot{J}_c\dot{q} = 0, \tag{33}$$

$4n_c$ bounds on CoP positions,
$6n_c$ bounds on friction forces,
30 joint limits,

3) 3 equality constraints for the CoM task,
3 equality constraints for the body orientation task,
$6(2 - n_c)$ equality constraints for the swing-foot task,

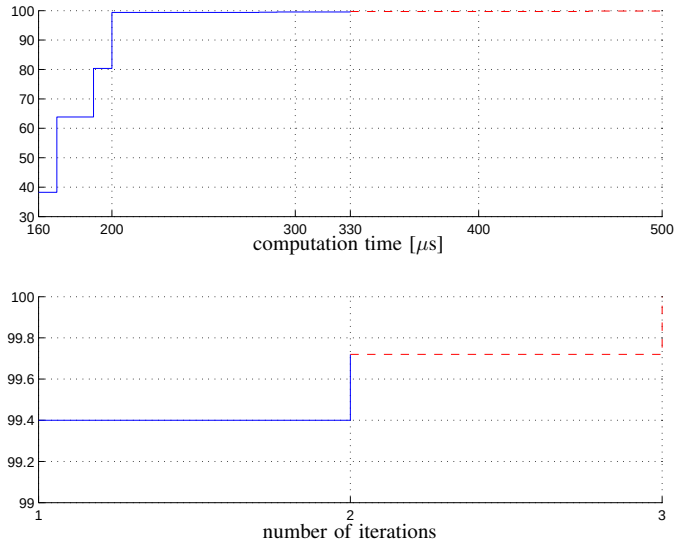


Fig. 10. Percentage of sampling periods where hierarchy **B** is solved within a given number of iterations and corresponding computation time. The number of iterations can be limited to 2 without influencing the motion of the robot in a noticeable way.

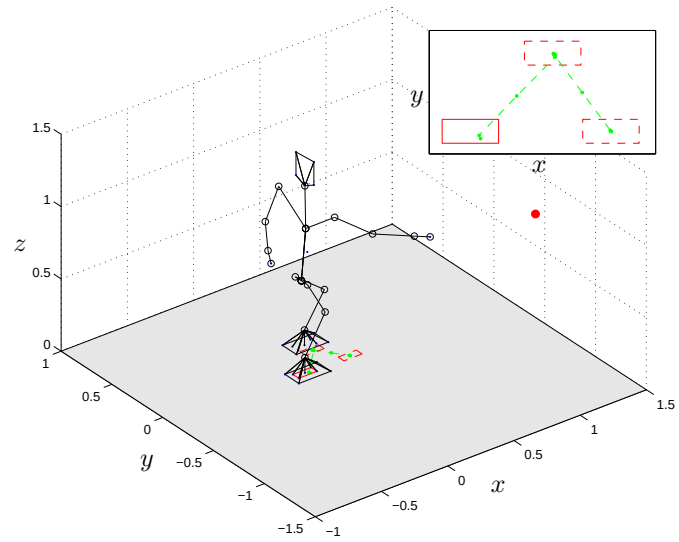


Fig. 11. A snapshot from the simulation with hierarchy **C**. The target object is represented as a red dot, while the profile of the CoP is depicted in green. The red rectangle and the red dashed rectangles represent the current and future footsteps, respectively.

4) 30 equality constraints to maintain a reference posture,
5) $6n_c$ equality constraints to minimize contact forces:

$$f = 0.$$

Note that the joint torques τ are eliminated beforehand, hence the decision variables of the above hierarchy are only $(\ddot{q}, f) \in \mathbb{R}^{36+6n_c}$. The number of feet contacting the flat ground is $n_c \in \{1, 2\}$. For more information about the Center of Pressure (CoP) condition, see [38]. There are several small differences between this hierarchy and the setting in [41]. First, we found it useful to impose in addition a bound on torsional friction ([1], Section 5.5) at the second hierarchical level. Second, the control of the trunk orientation is moved from level four to level three. Finally, the number of decision variables is slightly larger in our case since we use the model of a different robot. However, the examples are very similar overall.

Statistics on the number of iterations of the active-set method, and corresponding computation times using the proposed ℓ-QR decomposition, can be found in Fig. 10. We can observe that the active-set method requires very rarely more than one iteration when hot-start is used. Limiting the number of allowed iterations to 2 has a negligible effect, and in this case computation time is always lower than 330 μs on our slow CPU. In comparison, the computation time reported in [41] on a much more powerful CPU, and with a slightly smaller problem, is more than 2 ms. The proposed approach appears therefore to be very competitive.

C. Acceleration based control with MPC

Our third example is adopted from [42]. While instantaneous control at velocity and acceleration levels has been very popular in robotics, it is usually applied for the purpose of tracking a given reference trajectory. In many applications (*e.g.,* humanoid walking) such reference trajectory is often generated using Model Predictive Control (MPC) based on linear models [43], [44] and does not account for limitations inherent to the real robotic system. As a result of such separation between trajectory generation and tracking, the achieved performance might not be satisfactory. An approach that alleviates this is proposed in [42]. The main idea is to couple an acceleration based controller (similar to the one discussed in Section V-B) with a preview controller for a point mass (c_{mpc}) approximating the evolution of the CoM using a linear model. Such a combined formulation generates walking motions that take into account limitations due to the system dynamics and objectives/constraints related to high-level tasks, while leveraging one of the most appealing properties of MPC based schemes, namely robustness to perturbations. The following hierarchy is defined in [42]:

Hierarchy **C**

1) 30 joint acceleration limits: $\underline{\ddot{q}} \leq \ddot{q} \leq \bar{\ddot{q}}$,
   $2n_r$ bounds on previewed footsteps,
   $2n_p$ bounds on previewed CoP positions,
   $4n_c$ bounds on friction forces,
2) 6 equations of motion (32),
   $4n_c$ bounds on the current CoP position,
   $2n_c$ bounds on torsional friction,
   $6n_c$ equality constraints for feet contact (33),
   3 equality constraints for the swing-foot task,
   1 equality constraint to maintain a constant CoM height,
   2 equality constraints to couple the two models:

$$\ddot{c}^{x,y} = \ddot{c}^{x,y}_{mpc(0)},$$

3) 2 terminal equality constraints for the MPC,
4) 3 equality constraints for the body orientation task,
   3 equality constraints for the swing-foot orientation task,

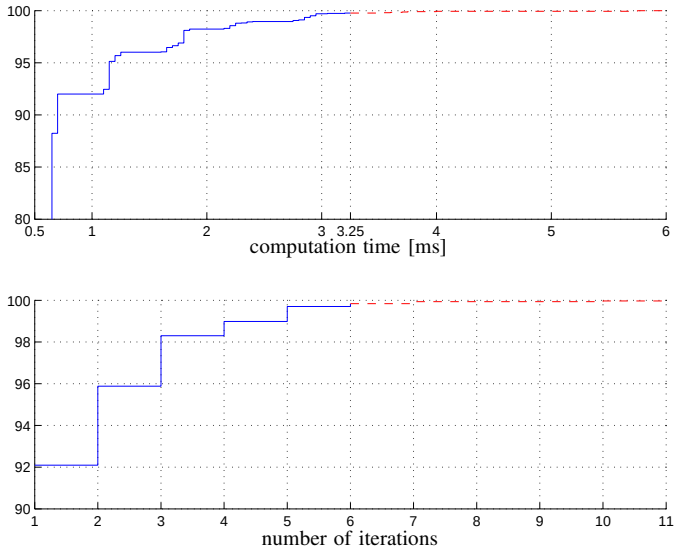


Fig. 12. Percentage of sampling periods where hierarchy **C** is solved within a given number of iterations and corresponding computation time. The number of iterations can be limited to 6 without influencing the motion of the robot in a noticeable way.

5) $4n_p$ equality constraints to center the previewed CoP,
3 equality constraints to minimize the jerk of the swing-foot,
3 equality constraints for the hand task, tracking the target object,
30 equality constraints to follow a desired joint acceleration.

The robot is commanded to grasp with its right hand a target object which follows a predefined motion profile. A snapshot of the simulation can be found in Fig. 11, where we can see the footsteps within the current preview window (dashed red boxes), the CoP profile and the robot walking towards the target object (red dot).

The decision variables in this hierarchy are: $\ddot{q} \in \mathbb{R}^{36}$, $f \in \mathbb{R}^{6n_c}$, the initial acceleration of the point mass in the mpc model $\ddot{c}^{x,y}_{mpc(0)} \in \mathbb{R}^2$, $2n_r$ previewed footsteps ($n_r \in \{0, 1, 2\}$), and $2n_p$ input variables for the MPC model ($n_p = 16$). As a result, the number of variables can grow up to 84, and the number of constraints up to 212. All inequality constraints at the first level are simple bounds and are treated as such. In this simulation, the problem appears to have a unique solution 98 % of the time, while in the remaining cases, a basic solution is obtained.

Statistics on the number of iterations of the active-set method, and corresponding computation times using the proposed method, can be found in Fig. 12. Limiting the number of allowed iterations of the active-set method to 6, which is sufficient to solve the problem 99 % of the time when using hot-start, has a negligible effect on the simulation, and in this case, computation time is always below 3.25 ms. We can observe that such a large and complex hierarchy can be tackled efficiently, even on the slow CPU considered here.

VI. Discussion

Variable elimination (also known as Schur complement method) is an efficient and reliable technique for reducing the size of linearly constrained problems [16], [17] and has been a common tool in robotics for the purpose of both analysis and numerical computations. As examples, consider the partial feedback linearization of underactuated systems in [45], the "generalized Jacobian" matrix introduced in the field of space robotics [46], the coordinate partitioning approach utilized in computational mechanics [47], [48].

In the context of hierarchical problems, variable elimination is routinely used during a pre-processing step [49], [50]. This is usually performed manually, in the sense that the variables to be eliminated and the order of elimination are predefined by the user. An example highlighting various pre-processing options is presented in [51]. However, after such reformulation, the resultant problem is typically solved using one of the classical methods in robotics involving the use of orthonormal bases [23], [52].

In contrast, the approach introduced in this article is based entirely on the concept of variable elimination: the size of a hierarchical problem is gradually reduced by means of a properly chosen sequence of elimination steps until a solution is identified. This can be seen as a generalization of the gradient projection method proposed in [53] for redundancy resolution. Note that the choice of which variables to eliminate (and in what order) is not unique, and in general has to be determined during the solution process in order to increase its reliability. We have adopted a column permutation strategy (see Section III-C) which automatically selects for each hierarchical level which variables lead to good numerical conditioning.

Manually constructing a sequence of variable elimination steps (which can be interpreted as a custom column permutation strategy) is rarely advantageous for the purposes of efficient computation. Exceptions are cases where the matrices involved in the elimination process have a very specific structure, which the solver is not aware of. The elimination of the joint torques τ is a good example. Since the leading matrix is actually the identity, there is no need for QR factorization to be performed and forming the Schur complement in (23) and (24) is typically trivial. Hence, it is usually beneficial to manually eliminate the joint torques from the decision variables (as we have done in both hierarchies **B** and **C** in Section V) [45], [25], [38], [54].

Ample research has been devoted to developing numerical methods dedicated to specific hierarchic (or constrained) problems [5], [25], [46], [49], [50], [52], [53], [55], [56], [57]. A typical example is when looking for joint accelerations consistent with Gauss' principle, minimizing an inertia-weighted norm $\ddot{q}^T H \ddot{q}$, as proposed in the Operational Space Control approach [5], [20], [21], [23], [58]. Despite significant efforts, this approach is still generally considered to be "computationally demanding, thereby making its real implementation difficult (...) explain why there have been so few real implementations" [24].

In contrast, we have proposed here a generic approach,

which can be adopted for any form of hierarchic (or constrained) problem. It is very efficient, as it appears to be as fast as an LU decomposition (see Section IV-B), which is generally considered as the fastest way to solve a system of linear equations. As a result, further efforts at improving computational performance for hierarchical problems will probably lead to very limited gain. This result would, hopefully, enable future research to focus less on computational performance and more on *design* questions, discussing what are desired behaviors and how to obtain them with a given set of objectives and constraints.

Note that having a general approach does not imply that one cannot exploit problem structure when it exists. For example, as demonstrated in Section V-A, our approach is particularly well suited to dealing with hierarchical problems involving simple bounds: a popular class of problems in robotics [55].

An important question not approached here is the case of ill-conditioning, when close to a singular situation. A traditional approach is to use Tikhonov regularization, usually called damping in the robotics literature [7], [19], [59]. In this case, the sequence of optimization problems is modified, but the hierarchic sequence of linear constraints remains unchanged. As a result, the proposed approach still applies to resolve these constraints efficiently. The difference lies in how the vectors $x^\star$ and $\rho^\star$ are obtained, with a small overhead in computation time. But questions such as when to regularize, how to regularize, and how to compute the regularized solution are all very subtle, especially in the case of a strict hierarchy, and still unanswered for the most part. A thorough analysis of all these questions is necessary, and this requires a distinct publication.

VII. Conclusion

Constraint prioritization is a powerful modeling tool that has been in the center of attention of many robotics researchers and practitioners. In this article we analyzed the structure of this prioritization, which can be seen as a hierarchical least-squares problem, and introduced a very efficient algorithm for its solution. This algorithm not only outperforms the state-of-the-art approaches in robotics, it also brings new insights about when and how priorities among linear equations should be considered. In particular, we have reached the rather unexpected conclusion that introducing hierarchy levels can be a valid approach to significantly reduce computation time, on the contrary to popular belief on this topic.

Our numerical results indicate that the proposed approach is very competitive across a variety of problems sizes. In particular, it appears to be as fast as an LU decomposition when solving a square and non-singular system of equations, which suggests that further efforts at improving computational performance for hierarchical problems will probably lead to very limited gain. We have presented a numerical evaluation of our `C++` implementation on three representative examples adopted from recent robotics literature. These results demonstrate that, even on the slow CPU embedded on the HRP-2 robot, hierarchical least-squares problems of practical interest, *e.g.,* with 80 variables and 200 equality and inequality constraints, can be solved reliably within only a few ms.

Appendix

Once a solution to the lexicographic problem (9) has been obtained, it is often desirable to obtain also the Lagrange multipliers corresponding to all P objectives with respect to all m constraints. This is required for example for the multi-objective active-set method proposed in [8], which is used to solve the inequality-constrained problems in Section V.

Let us use $\lambda_{jk} \in \mathbb{R}^{m_j}$ to denote the sensitivity of objective k with respect to the constraints involved in objective j. Due to the hierarchical nature of our problem, the optimal residual of objective k is not affected by the constraints involved in objective j when $k < j$. As a result, we have $\lambda_{jk} = 0$ when $k < j$. It can be helpful to summarize this sensitivity information in a matrix form, with the following structure:

$$\lambda = \begin{bmatrix} \lambda_{11} & \lambda_{12} & \dots & \lambda_{1P} \\ & \lambda_{22} & \dots & \lambda_{2P} \\ & & \ddots & \vdots \\ & & & \lambda_{PP} \end{bmatrix} \in \mathbb{R}^{m \times P}.$$

This matrix can be constructed sequentially, one column at a time. In order to do so, consider the minimization of the k-th objective in the lexicographic problem (9), scaled by $\frac{1}{2}$ for convenience. Using the ℓ-QR decomposition (17) of the matrix A, the optimality conditions are given by $\lambda_{kk} = r_k^\star$, and

$$A^T \lambda_k = 0 \quad \Leftrightarrow \quad Q_\ell^{'T} \lambda_k = 0,$$

where λ_k is the k-th column of λ, and we have used that $Q_\ell^{'}$ has full column rank and R_ℓ full row rank. Since the matrix $Q_\ell^{'T}$ is block-wise upper-triangular, we can easily verify that a solution to these optimality conditions is given by an efficient block-wise backward substitution

$$\forall j < k, \quad \lambda_{jk} = -Q_j^{'} \left(\sum_{i=j+1}^{k-1} L_{ij} \lambda_{ik} + L_{kj} r_k^\star \right),$$

following the same rationale as in Section III. Note that in the presence of singularities, this solution is not unique.

References

[1] R. Murray, Z. Li, and S. Sastry, *A mathematical introduction to robotic manipulation*. CRC Press, 1994.

[2] C. Samson, M. Le Borgne, and B. Espiau, *Robot control: the task function approach*. Oxford, U.K.: Clarendon Press, 1991.

[3] O. Khatib, "A unified approach for motion and force control of robot manipulators: The operational space formulation," *IEEE Transactions on Robotics and Automation*, vol. RA-3, no. 1, pp. 43–53, 1987.

[4] Y. Nakamura, H. Hanafusa, and T. Yoshikawa, "Task-priority based redundancy control of robot manipulators," *The International Journal of Robotics Research (IJRR)*, vol. 6, no. 2, pp. 3–15, 1987.

[5] O. Khatib, L. Sentis, J. Park, and J. Warren, "Whole-body dynamic behavior and control of human-like robots," *International Journal of Humanoid Robotics*, vol. 1, no. 1, pp. 29–43, 2004.

[6] J. Nakanishi, R. Cory, M. Peters, and S. Schaal, "Operational space control: a theoretical and empirical comparison," *The International Journal of Robotics Research (IJRR)*, vol. 27, no. 6, pp. 737–757, 2008.

[7] O. Kanoun, F. Lamiraux, and P.-B. Wieber, "Kinematic control of redundant manipulators: generalizing the task-priority framework to inequality task," *IEEE Transactions on Robotics*, vol. 27, no. 4, pp. 785–792, 2011.

[8] A. Escande, N. Mansard, and P.-B. Wieber, "Hierarchical quadratic programming: Fast online humanoid-robot motion generation," *The International Journal of Robotics Research (IJRR)*, vol. 33, no. 7, pp. 1006–1028, 2014.

[9] H. Isermann, "Linear lexicographic optimization," *Operations Research Spektrum*, vol. 4, no. 4, pp. 223–228, 1982.

[10] T. Kitahara and T. Takashi, "Proximity of weighted and layered least squares solutions," *SIAM Journal on Matrix Analysis and Applications*, vol. 31, no. 3, pp. 1172–1186, 2009.

[11] A. Ben-Israel and T. Greville, *Generalized inverses theory and applications*. Springer, 2003.

[12] H. Nakayama, "Aspiration level approach to interactive multi-objective programming and its applications," in *Advances in Multicriteria Analysis (Kluwer Academic Publishers)*, pp. 147–174, 1995.

[13] O. Ott, J. Artigas, and C. Preusche, "Subspace-oriented energy distribution for the time domain passivity approach," in *IEEE/RSJ International Conference on Intelligent Robots and System (IROS)*, pp. 665–671, 2011.

[14] J. Salini, V. Padois, and P. Bidaud, "Synthesis of complex humanoid whole-body behavior: a focus on sequencing and tasks transitions," in *IEEE/RSJ International Conference on Robotics and Automation (ICRA)*, pp. 1283–1290, 2011.

[15] T. Sugihara, "Robust solution of prioritized inverse kinematics based on hestenes-powell multiplier method," in *IEEE/RSJ International Conference on Intelligent Robots and System (IROS)*, pp. 510–515, 2014.

[16] Å. Björck, *Numerical methods for least squares problems*. SIAM, 1996.

[17] G. W. Stewart, *Matrix algorithms, volume I: basic decompositions*. SIAM, 1998.

[18] J. Nocedal and S. Wright, *Numerical optimization*. Springer, 1999.

[19] O. Kanoun, "Real-time prioritized kinematic control under inequality constraints for redundant manipulators," in *Proceedings of Robotics: Science and Systems*, 2011.

[20] H. Bruyninckx and O. Khatib, "Gauss' principle and the dynamics of redundant and constrained manipulators," in *IEEE/RSJ International Conference on Robotics and Automation (ICRA)*, pp. 2563–2568, 2000.

[21] L. Sentis and O. Khatib, "Synthesis of whole-body behaviors through hierarchical control of behavioral primitives," *International Journal of Humanoid Robotics*, vol. 2, no. 4, pp. 505–518, 2005.

[22] J. Park, *Control strategies for robots in contact*. PhD thesis, Stanford University, 2006.

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

[24] P. Chang and J. Jeong, "Enhanced operational space formulation for multiple tasks by using time-delay estimation," *IEEE Transactions on Robotics*, vol. 28, no. 4, pp. 773–786, 2012.

[25] L. Righetti, J. Buchli, M. Mistry, and S. S., "Inverse dynamics control of floating-base robots with external constraints: a unified view," in *IEEE/RSJ International Conference on Robotics and Automation (ICRA)*, pp. 1085–1090, 2011.

[26] G. Golub and C. Van Loan, *Matrix computations*. The Johns Hopkins University Press, 2013.

[27] M. Gulliksson and P.-Å. Wedin, "Modifying the QR-decomposition to constrained and weighted linear least squares," *SIAM Journal on Matrix Analysis and Applications*, vol. 13, no. 4, pp. 1298–1313, 1992.

[28] J. Reid, "Implicit scaling of linear least squares problems," *BIT Numerical Mathematics*, vol. 40, no. 1, pp. 146–157, 2000.

[29] G. Stewart, "On the asymptotic behavior of scaled singular value and QR decompositions," *Mathematics of Computation*, vol. 43, no. 168, pp. 483–489, 1984.

[30] Å. Björck, "Iterative refinement of linear least squares solutions II," *BIT Numerical Mathematics*, vol. 8, no. 1, pp. 8–30, 1968.

[31] A. Anda and H. Park, "Self-scaling fast rotations for stiff and equality-constrained linear least squares problems," *Linear Algebra and its Applications*, vol. 234, no. 4, pp. 137–161, 1996.

[32] A. M. Turing, "Rounding-off errors in matrix processes," *Quarterly Journal of Mechanics & App. Maths.*, vol. 1, no. 1, pp. 287 – 308, 1948.

[33] C. Van Loan, "On the method of weighting for equality constrained least squares problems," *SIAM Journal on Numerical Analysis*, vol. 22, no. 5, pp. 851–864, 1985.

[34] K. Kaneko, F. Kanehiro, S. Kajita, H. Hirukawa, T. Kawasaki, M. Hirata, K. Akachi, and T. Isozumi, "Humanoid robot HRP-2," in *IEEE/RSJ International Conference on Robotics and Automation (ICRA)*, pp. 1083–1090, 2004.

[35] "Ubuntu." https://wiki.ubuntu.com.

[36] G. Guennebaud, B. Jacob, *et al.*, "Eigen v3." http://eigen.tuxfamily.org, 2010.

[37] M. Lam, E. Rothberg, and M. Wolf, "The cache performance and optimizations of blocked algorithms," in *International conference on Architectural Support for Programming Languages and Operating Systems (ASPLOS)*, pp. 63–74, 1991.

[38] Y. Fujimoto and A. Kawamura, "Simulation of an autonomous biped walking robot including environmental force interaction," *IEEE Robotics & Automation Magazine*, vol. 5, no. 2, pp. 33–42, 1998.

[39] P.-B. Wieber, "Holonomy and nonholonomy in the dynamics of articulated motion," in *Fast Motions in Biomechanics and Robotics*, pp. 411–425, 2006.

[40] T. Koolen, T. de Boer, J. Rebula, A. Goswami, and J. Pratt, "Capturability-based analysis and control of legged locomotion, part 1: Theory and application to three simple gait models," *The International Journal of Robotics Research (IJRR)*, vol. 31, no. 9, pp. 1094–1113, 2012.

[41] A. Herzog, L. Righetti, F. Grimminger, P. Pastor, and S. Schaal, "Balancing experiments on a torque-controlled humanoid with hierarchical inverse dynamics," in *IEEE/RSJ International Conference on Intelligent Robots and System (IROS)*, pp. 981–988, 2014.

[42] A. Sherikov, D. Dimitrov, and P.-B. Wieber, "Whole body motion controller with long-term balance constraints," in *IEEE-RAS International Conference on Humanoid Robots (Humanoids)*, pp. 444–450, 2014.

[43] A. Herdt, H. Diedam, P.-B. Wieber, D. Dimitrov, K. Mombaur, and M. Diehl, "Online walking motion generation with automatic foot step placement," *Advanced Robotics*, vol. 24, no. 5–6, pp. 719–737, 2010.

[44] N. Mansard, O. Stasse, F. Chaumette, and K. Yokoi, "Visually-guided grasping while walking on a humanoid robot," in *IEEE/RSJ International Conference on Robotics and Automation (ICRA)*, pp. 3041–3047, 2007.

[45] M. W. Spong, "Partial feedback linearization of underactuated mechanical systems," in *IEEE/RSJ International Conference on Intelligent Robots and System (IROS)*, pp. 314–321, 1994.

[46] Y. Umetani and K. Yoshida, "Resolved motion rate control of space manipulators with generalized jacobian matrix," *IEEE Transactions on Robotics and Automation*, vol. 5, no. 3, pp. 303–314, 1989.

[47] A. Kurdila, J. G. Papastavridis, and M. P. Kamat, "Role of Maggi's equations in computational methods for constrained multibody systems," *Journal of Guidance, Control, and Dynamics*, vol. 13, no. 1, pp. 113–120, 1990.

[48] R. von Schwerin, *MultiBody System SIMulation: Numerical Methods, Algorithms, and Software*. Springer, 1999.

[49] N. Nenchev and K. Yoshida, "Impact analysis and post-impact motion control issues of a free-floating space robot subject to a force impulse," *IEEE Transactions on Robotics and Automation*, vol. 15, no. 3, pp. 548–557, 1999.

[50] N. Mansard, "A dedicated solver for fast operational-space inverse dynamics," in *IEEE/RSJ International Conference on Robotics and Automation (ICRA)*, pp. 4943–4949, 2012.

[51] D. Dimitrov, P.-B. Wieber, and A. Escande, "Multi-objective control of robots," *Journal of the Robotics Society of Japan*, vol. 32, no. 6, pp. 512–518, 2014.

[52] B. Siciliano and J.-J. Slotine, "A general framework for managing multiple tasks in highly redundant robotic systems," in *Fifth International Conference on Advanced Robotics (ICAR)*, pp. 1211–1216, 1991.

[53] H. Zghal, R. V. Dubey, and J. A. Euler, "Efficient gradient projection optimization for manipulators with multiple degrees of redundancy," in *IEEE/RSJ International Conference on Robotics and Automation (ICRA)*, pp. 1006–1011, 1990.

[54] P.-B. Wieber, "On the stability of walking systems," in *Proceedings of the international workshop on humanoid and human friendly robotics*, 2002.

[55] F. Flacco, A. De Luca, and O. Khatib, "Control of redundant robots under hard joint constraints: Saturation in the null space," *IEEE Transactions on Robotics*, vol. 31, no. 3, pp. 637–654, 2015.

[56] M. Mistry and L. Righetti, "Operational space control of constrained and underactuated systems," in *Proceedings of Robotics: Science and Systems*, 2011.

[57] F. Aghili, "A unified approach for inverse and direct dynamics of constrained multibody systems based on linear projection operator: applications to control and simulation," *IEEE Transactions on Robotics*, vol. 21, no. 5, pp. 834–849, 2005.

[58] L. Saab, O. Ramos, F. Keith, N. Mansard, P. Souères, and J.-Y. Fourquet, "Dynamic whole-body motion generation under rigid contacts and other unilateral constraints," *IEEE Transactions on Robotics*, vol. 29, no. 2, pp. 346–362, 2013.

[59] A. Deo and I. Walker, "Overview of damped least-squares methods for inverse kinematics of robot manipulators," *Journal of Intelligent and Robotic Systems*, vol. 14, no. 1, pp. 43–68, 1995.