Singularity resolution in equality and inequality constrained hierarchical task-space control by adaptive non-linear least-squares

Kai Pfeiffer, Adrien Escande, Abderrahmane Kheddar

Singularity resolution in equality and inequality constrained hierarchical task-space control by adaptive non-linear least-squares

Kai Pfeiffer[1,2], Adrien Escande[1] and Abderrahmane Kheddar[1,2]

***Abstract*— We propose a robust method to handle kinematic and algorithmic singularities of any kinematically redundant robot under task-space hierarchical control with ordered equalities and inequalities. Our main idea is to exploit a second order model of the non-linear kinematic function, in the sense of the Newton's method in optimization. The second order information is provided by a hierarchical BFGS algorithm omitting the heavy computation required for the true Hessian. In the absence of singularities, which is robustly detected, we use the Gauss-Newton algorithm that has quadratic convergence. In all cases we keep a least-squares formulation enabling good computation performances. Our approach is demonstrated in simulation with a simple robot and a humanoid robot, and compared to state-of-the-art algorithms.**

I. Introduction

Hierarchical (or strict priority-based) task-space schemes are a popular approach for the control of kinematically redundant robots, i.e. robots having more degrees of freedom (DoF) than required to fulfil a given task [1]. For a task-space error function that must be driven to zero, a linearized model is built at the velocity or acceleration level, and solved in a least-squares way. The null space of the linear model can be used to satisfy at best a second task in the same way without interfering with the first one. This generalizes to any number of tasks, as long as some DoF's are remaining [2].

While the approach has been long known for equality-only tasks, it is only with the seminal work of [3] that a full support of inequality tasks at any priority level has been proposed. There, the resolution of the linearized problem is done through a so-called *cascade* of constrained least-squares programs (LSP): each level of the hierarchy is solved in turn in a least-squares sense under the constraint of keeping all previous levels at their optimal amount of violation. The downside of this approach is a relatively high computation cost because each LSP is redoing some work done by the previous ones, and it is not possible to use a proper warm-start to leverage the similarities between problems in consecutive control iterations. A dedicated solver was proposed in [4] and further improved in [5], enabling to solve the same general hierarchical problem very efficiently.

The conjunction of a linear model and a hierarchical least-squares resolution is however failing when approaching a singularity: the Jacobian matrix of one of the tasks (kinematic singularity case) or its projection onto the nullspace of higher-priority tasks (algorithmic singularity case) is becoming nearly rank-deficient [1]; its inversion therefore yields very high and unwanted joint velocities or accelerations, leading to instabilities. The first kind of singularity typically arises when a target is out of reach, while the second one is a consequence of conflicts between tasks.

One way to handle singularities is to compute an approximate solution to the linear problem with reasonable values. This is most commonly achieved with *damping*, where a term minimizing the solution norm is added to the least-squares. It is introduced with a weight that can be either fixed by the user or can be automatically adapted, for example using the so-called manipulability factor [6] or some estimate of the minimum singular value [7]. Another approach prohibits infeasible end-effector targets by clamping and introducing a more refined damping considering all the Jacobian's singular values w.r.t. the robot's joints distinctively [8]. A very simple method avoids the Jacobian inversion and uses its transpose instead, with stable but worse convergence results [9].

Singularity handling has been mostly studied for equality tasks only, and is still an open problem for a general hierarchy including inequality tasks. This is why, despite their efficiency, the solvers [4][5] have never been used online on a real robot. To the best of our knowledge, the only instance of a general hierarchy tackling singularities is [10] where the authors propose an improved cascade of LSP that includes fixed damping at every level. While faster than [3], it is still suffering from the same drawback and from the fact that tuning the damping weights can be difficult: too low gains and the solution will be too large; too high gains and the convergence will be hindered.

There is another way to look at the shortcomings of the linear model with the least-squares resolution approach: rather than approximating the solution, change the model to better specify the solution we want. It can be indeed argued that the linear model is not 'good enough' in the vicinity of a singularity: the near rank deficiency of the Jacobian matrix means that we are losing information in one direction. Since the first derivative proves to be insufficient, we could incorporate second-order information.

It is interesting to draw parallels between control and non-linear least-squares optimization [11], [12]. For one task, the least-squares minimization of a linear model corresponds to the Gauss-Newton (GN-) method, while the damping approach is akin to the Levenberg-Marquardt (LM-) algorithm [13]. When the full second order information is used, we get the Newton (N-) method. In fact, the three methods have a ground commonality: they are all derived from the

This research is supported in part by the CNRS-AIST-AIRBUS Joint Research Program, and the EU H2020 COMANOID project.

[1]CNRS-AIST JRL (Joint Robotic Laboratory) UMI3218/RL, Japan

[2]CNRS-University of Montpellier, LIRMM UMR5506, Interactive Digital Human, France

second order Taylor approximation of the squared norm of a non-linear function, only differing in whether the Hessian is given analytically or by some approximation. Second order derivatives are usually costly to compute, so that several methods have been proposed to approximate them [14], [15], the most widely used being the BFGS formula [16], derived from the secant equation. Newton methods with Hessian approximation are named Quasi-Newton (QN-) methods.

Neither the N-method nor the QN-method is specifically designed to serve in multi-level hierarchies. In [17], a QN-method using BFGS updates for the second order information, weights its objectives to establish a non-strict hierarchy. LM-algorithms like [18], [1] can be included into a strict hierarchical scheme by damping and handing each objective to a hierarchical least-squares solver like in [4], [5], though missing the higher accuracy of the N-method.

In this paper we combine the (strict) hierarchy task-space control scheme with the high accuracy and computation speed of QN-methods: we propose a new singularity robust method for prioritized inverse kinematics with any number of hierarchical levels of equalities and inequalities, may they be feasible or not, while encompassing numerical stability. Our QN-approach outperforms current state of the art in singularity robust inverse kinematics methods in terms of accuracy while being computationally competitive.

II. Second order information in a hierarchy

In this paper, we consider a robot with configuration $\boldsymbol{q}$, and a set of p geometric functions $\boldsymbol{f}(\boldsymbol{q})$, one for each priority level. For each function we define a target or bound $\boldsymbol{f}_d$, defining tasks $\boldsymbol{f}(\boldsymbol{q}) = \boldsymbol{f}_d$ or $\boldsymbol{f}(\boldsymbol{q}) \leq \boldsymbol{f}_d$, and the error function $\boldsymbol{e} = \boldsymbol{f}_d - \boldsymbol{f}$. We derive a hierarchical velocity-based controller minimizing those errors. The targets $\boldsymbol{f}_d$ can be time varying, but at each control iteration, we consider them as constant (no feed-forward term).

At each control step k, we need to compute a velocity $\dot{\boldsymbol{q}}$. The next state $\boldsymbol{q}^{k+1}$ is then integrated to

$$\boldsymbol{q}^{k+1} = \boldsymbol{q}^k + \Delta\boldsymbol{q}^k. \tag{1}$$

with the increment $\Delta\boldsymbol{q} = \dot{\boldsymbol{q}}\Delta t$ ($\Delta t = 1$).

In the following we will drop the index k for a better reading (other indices $k-1$, $k+1$... are kept).

Given a single function $\boldsymbol{f}$ with Jacobian $\boldsymbol{J}$, we can look for $\Delta\boldsymbol{q}$ yielding a given error decrease $\Delta\boldsymbol{e}^*$ with the relation

$$\Delta\boldsymbol{e}^* = -\boldsymbol{J}\Delta\boldsymbol{q}. \tag{2}$$

For the sake of simplicity we take $\Delta\boldsymbol{e}^* = -\boldsymbol{e}$.

The required error decrease may not be achievable and we solve the above using a linear least-squares

$$\min_{\Delta\boldsymbol{q}} \quad \frac{1}{2}\|\boldsymbol{J}\Delta\boldsymbol{q} - \boldsymbol{e}\|_2^2 \tag{3}$$

$$= \min_{\Delta\boldsymbol{q}} \quad \frac{1}{2}\boldsymbol{e}^T\boldsymbol{e} - \Delta\boldsymbol{q}^T\boldsymbol{J}^T\boldsymbol{e} + \frac{1}{2}\Delta\boldsymbol{q}^T\boldsymbol{J}^T\boldsymbol{J}\Delta\boldsymbol{q}. \tag{4}$$

A solution is given by $\Delta\boldsymbol{q} = \boldsymbol{J}^+\boldsymbol{e}$, where $\boldsymbol{J}^+$ is the Moore-Penrose pseudo inverse. This formulation corresponds to the GN-algorithm.

As discussed above, this approach is not robust in the vicinity of singularities: $\boldsymbol{J}$ is almost loosing a rank and $\Delta\boldsymbol{q}$ becomes very large. We now discuss how second order information can be incorporated into the linear model.

Let's define a scalar target function as follows [11]:

$$\Phi(\boldsymbol{q}) = \frac{1}{2}\|\boldsymbol{f}_d - \boldsymbol{f}(\boldsymbol{q})\|_2^2 = \frac{1}{2}\|\boldsymbol{e}(\boldsymbol{q})\|_2^2 = \frac{1}{2}\boldsymbol{e}^T\boldsymbol{e}. \tag{5}$$

$\Phi(\boldsymbol{q})$ is non-linear (quadratic, trigonometric functions) and can only be handled by means of non-linear programming. Yet, we can reformulate the problem to be approximated as linear least-squares.

We approximate $\Phi(\boldsymbol{q})$ in a small neighbourhood $\Delta\boldsymbol{q}$ of point $\boldsymbol{q}$ by a second order Taylor series:

$$\Phi^{k+1}(\boldsymbol{q} + \Delta\boldsymbol{q}) \approx \Phi(\boldsymbol{q}) + \Delta\boldsymbol{q}^T\nabla\Phi + \frac{1}{2}\Delta\boldsymbol{q}^T\nabla^2\Phi\Delta\boldsymbol{q} \tag{6}$$

$$= \Phi(\boldsymbol{q}) - \Delta\boldsymbol{q}^T\boldsymbol{J}^T\boldsymbol{e} + \frac{1}{2}\Delta\boldsymbol{q}^T(\boldsymbol{J}^T\boldsymbol{J}+\boldsymbol{H})\Delta\boldsymbol{q}.$$

We have the gradient

$$\nabla\Phi(\boldsymbol{q}) = -\boldsymbol{J}^T\boldsymbol{e} \tag{7}$$

and the Hessian

$$\nabla^2\Phi(\boldsymbol{q}) = \boldsymbol{J}^T\boldsymbol{J} + \boldsymbol{H} = \boldsymbol{J}^T\boldsymbol{J} + \sum_{i=1}^{\dim(\boldsymbol{f})} \boldsymbol{e}_i\nabla^2\boldsymbol{f}_i \tag{8}$$

where $\boldsymbol{H}$ gathers the second order information (see next section). $\Delta\boldsymbol{q}$ can then be computed by

$$\min_{\Delta\boldsymbol{q}} \quad \Phi^{k+1}(\boldsymbol{q} + \Delta\boldsymbol{q}) \tag{9}$$

$$= \min_{\Delta\boldsymbol{q}} \quad \frac{1}{2}\boldsymbol{e}^T\boldsymbol{e} - \Delta\boldsymbol{q}^T\boldsymbol{J}^T\boldsymbol{e} + \frac{1}{2}\Delta\boldsymbol{q}^T(\boldsymbol{J}^T\boldsymbol{J} + \boldsymbol{H})\Delta\boldsymbol{q}.$$

This corresponds to the Newton method applied to non-linear least-squares [13]. We see that by neglecting the second order term $\boldsymbol{H}$, we get back to the GN formulation. Taking $\boldsymbol{H}$ as a multiple of the identity yields the LM approach.

If $\boldsymbol{H}$ is positive definite, or approximated by a positive definite matrix, we can take its Cholesky decomposition $\boldsymbol{H} = \boldsymbol{R}^T\boldsymbol{R}$. This leads to the LSP

$$\min_{\Delta\boldsymbol{q}} \quad \frac{1}{2}\left\|\begin{bmatrix}\boldsymbol{J}\\ \boldsymbol{R}\end{bmatrix}\Delta\boldsymbol{q} - \begin{bmatrix}\boldsymbol{e}\\ \boldsymbol{0}\end{bmatrix}\right\|_2^2 \tag{10}$$

$$= \min_{\Delta\boldsymbol{q}} \quad \frac{1}{2}\|\boldsymbol{J}\Delta\boldsymbol{q} - \boldsymbol{e}\|_2^2 + \frac{1}{2}\|\boldsymbol{R}\Delta\boldsymbol{q}\|_2^2.$$

This is the approach we propose in this paper. It can be seen as an augmentation of the GN formulation with the second order information in $\boldsymbol{R}$. $\boldsymbol{R}$ is obtained with the Bunch-Kaufman decomposition $\boldsymbol{L}\boldsymbol{D}\boldsymbol{L}^T$ [19] where the 1×1 blocks of the block diagonal matrix $\boldsymbol{D}$ have to be positive and the 2×2 blocks need to be positive definite [20]. The factorization $\boldsymbol{R}^T\boldsymbol{R}$ is then retrieved by $\boldsymbol{R} = \boldsymbol{D}^{1/2}\boldsymbol{L}^T$. The regularization ensures positive definiteness of $\boldsymbol{R}$ and therefore the convexity of our problem.

The above derivation can be conducted for each task in the hierarchy (for inequality tasks, it simply requires the introduction of a slack variable, see [3]). We thus end up with a set of priority-ordered least-squares problems of the form (10) that we can pass to the hierarchical solver [5].

The next section presents our approach to compute $\boldsymbol{H}$.

III. Quasi-Newton scheme for Hessian calculation

A. Hessian calculation for single level with equalities only

The calculation of the second order derivatives $\nabla^2 \boldsymbol{f}_i$ in $\boldsymbol{H}$ of (8) is of complexity $\mathcal{O}(d^3)$, d is the number of the robot's DoF. We propose a faster method based on the BFGS algorithm. Our computational complexity for $\boldsymbol{H}$ is $\mathcal{O}(d^2)$.

The BFGS method iteratively computes an approximation of the whole Hessian $\boldsymbol{B}$, using only gradient information:

$$\boldsymbol{B} = \boldsymbol{B}^{k-1} + \frac{\boldsymbol{y}\boldsymbol{y}^T}{\boldsymbol{y}^T \Delta \boldsymbol{q}^{k-1}} - \frac{\boldsymbol{B}^{k-1}\Delta \boldsymbol{q}^{k-1}\Delta \boldsymbol{q}^{k-1,T}\boldsymbol{B}^{k-1}}{\Delta \boldsymbol{q}^{k-1,T}\boldsymbol{B}^{k-1}\Delta \boldsymbol{q}^{k-1}}. \tag{11}$$

$\boldsymbol{y}$ is the difference between the current $\nabla\Phi$ and the previous gradient $\nabla\Phi^{k-1}$: $\boldsymbol{y} = \nabla\Phi - \nabla\Phi^{k-1}$.

The update $\boldsymbol{B}$ is positive definite if $\boldsymbol{B}^{k-1}$ is also positive definite and the curvature is greater than zero $\boldsymbol{y}^T \Delta \boldsymbol{q}^{k-1} > \xi$, otherwise no update is done. In theory $\xi = 0$ but in practice we choose a numerical threshold like $\xi = 10^{-12}$.

$\boldsymbol{B}^{k-1}$ needs to be initialized with a positive definite matrix in the very first iteration $k = 1$, for example by

$$\boldsymbol{B}^0 = \boldsymbol{J}^{0,T}\boldsymbol{J}^0 + \mu \boldsymbol{I}. \tag{12}$$

At the start, $\boldsymbol{J}^0$ is not available so we use $\boldsymbol{J}$ instead. This resembles the true Hessian in (8). Since $\boldsymbol{J}$ and therefore also $\boldsymbol{J}^T\boldsymbol{J}$ might be close to singularity we fill the diagonal with a weighted identity matrix to make $\boldsymbol{B}^0$ regular.

As in [18], we set $\mu = \max(\underline{\mu}, \frac{1}{2}\|\boldsymbol{e}\|_2^2)$ with $\underline{\mu} = 1 \cdot 10^{-3}$ being a lower threshold to handle cases where the error is very small but the task is still augmented.

For our actual least-squares formulation the BFGS update $\boldsymbol{B}$ needs to be reduced by

$$\boldsymbol{H} = \boldsymbol{B} - \boldsymbol{J}^T\boldsymbol{J} \tag{13}$$

in order to receive the desired value $\boldsymbol{H}$. However, this results in an indefinite matrix $\boldsymbol{H}$ which can lead to unstable or unsmooth behaviour despite the regularization to a positive definite matrix. We therefore choose to not conduct this reduction (for consistency, the initialization is also done without adding $\boldsymbol{J}^T\boldsymbol{J}$). We observed that the QN-method is very forgiving in terms of simplifications in the second order information. As long as the second order information is positive definite we will converge at least linearly. Unfortunately, with the current formulation it is not possible to directly update $\boldsymbol{H}$.

B. Hessian calculation for hierarchies with equalities only

Solving a p levels hierarchy can be expressed as solving a sequence of constrained least-squares problems where the violation $\boldsymbol{w}_l$ of the objective of level l has to be minimized without increasing the (fixed) violation $\boldsymbol{w}_i^*$ of already minimized objectives of higher priority $i = 1, \ldots, l-1$:

$$\begin{aligned} &\text{For } l = 1, \ldots, p: \\ &\quad \min_{\Delta\boldsymbol{q}, \boldsymbol{w}_l} \quad \frac{1}{2}\|\boldsymbol{w}_l\|_2^2 \\ &\quad \text{subject to} \quad \boldsymbol{J}_l\Delta\boldsymbol{q} - \boldsymbol{e}_l = \boldsymbol{w}_l \\ &\qquad\qquad\quad \boldsymbol{J}_i\Delta\boldsymbol{q} - \boldsymbol{e}_i = \boldsymbol{w}_i^*,\ i = 1, \ldots, l-1 \end{aligned} \tag{14}$$

The corresponding Lagrange function for constrained optimization can be formulated as follows:

$$\begin{aligned} \mathcal{L}_l &= \frac{1}{2}\boldsymbol{e}_l^T\boldsymbol{e}_l - \Delta\boldsymbol{q}^T\boldsymbol{J}_l^T\boldsymbol{e}_l + \frac{1}{2}\Delta\boldsymbol{q}^T\boldsymbol{J}_l^T\boldsymbol{J}_l\Delta\boldsymbol{q} \\ &\quad + \sum_{i=1}^{l-1}\left(\boldsymbol{J}_i\Delta\boldsymbol{q} - \boldsymbol{e}_i - \boldsymbol{w}_i^*\right)^T\boldsymbol{\lambda}_{i,l} \\ &= \frac{1}{2}\boldsymbol{e}_l^T\boldsymbol{e}_l - \Delta\boldsymbol{q}^T\boldsymbol{J}_l^T\boldsymbol{e}_l + \frac{1}{2}\Delta\boldsymbol{q}^T\boldsymbol{J}_l^T\boldsymbol{J}_l\Delta\boldsymbol{q} \\ &\quad + \sum_{i=1}^{l-1}\left(\Delta\boldsymbol{q}^T\boldsymbol{J}_i^T - (\boldsymbol{e}_i + \boldsymbol{w}_i^*)^T\right)\boldsymbol{\lambda}_{i,l} \\ &= \frac{1}{2}\boldsymbol{e}_l^T\boldsymbol{e}_l - \sum_{i=1}^{l-1}(\boldsymbol{e}_i + \boldsymbol{w}_i^*)^T\boldsymbol{\lambda}_{i,l} \\ &\quad + \Delta\boldsymbol{q}^T\left(-\boldsymbol{J}_l^T\boldsymbol{e}_l + \sum_{i=1}^{l-1}\boldsymbol{J}_i^T\boldsymbol{\lambda}_{i,l}\right) + \frac{1}{2}\Delta\boldsymbol{q}^T\boldsymbol{J}_l^T\boldsymbol{J}_l\Delta\boldsymbol{q}. \end{aligned} \tag{15}$$

$\boldsymbol{\lambda}$ are the Lagrange multipliers where $\boldsymbol{\lambda}_{i,l}$ is the vector of Lagrange multipliers indicating the conflict of level l with level i. The following equivalence with the task slack variable holds [4]:

$$\boldsymbol{\lambda}_{l,l} = \boldsymbol{w}_l. \tag{16}$$

(15) has the characteristics of a hierarchical version of the GN-algorithm (4). Its gradient can be written as

$$\nabla\Phi(\boldsymbol{q}) = -\boldsymbol{J}_l^T\boldsymbol{e}_l + \sum_{i=1}^{l-1}\boldsymbol{J}_i^T\boldsymbol{\lambda}_{i,l}. \tag{17}$$

This hierarchical GN-gradient is then used to compute $\boldsymbol{y}_l$, and consequently $\boldsymbol{B}_l$.

We only use the Lagrange multipliers of the current iteration in the calculation of $\boldsymbol{y}_l = (\boldsymbol{J}_l - \boldsymbol{J}_l^{k-1})^T\boldsymbol{\lambda}_l$ [13].

$\boldsymbol{B}_l^0$ of level l is initialized as

$$\boldsymbol{B}_l^0 = \boldsymbol{J}_l^T\boldsymbol{J}_l + \sum_{i=1}^{l}\max(\underline{\mu}, \frac{1}{2}\|\boldsymbol{e}_i\|_2^2)\boldsymbol{I}_i^*. \tag{18}$$

$\boldsymbol{I}^*$ is an identity matrix where only diagonal entries are occupied when the corresponding joint is on the kinematic chain of the task. This prevents unnecessarily decreasing the null-space of this task and allows better results with lower priority tasks.

C. Hessian calculation for general hierarchies

When there are also inequality constraints, (14) becomes

$$\begin{aligned} &\text{For } l = 1, \ldots, p: \\ &\quad \min_{\Delta\boldsymbol{q}, \boldsymbol{w}_l} \quad \frac{1}{2}\|\boldsymbol{w}_l\|_2^2 \\ &\quad \text{subject to} \quad \boldsymbol{l}_l \le \boldsymbol{J}_l\Delta\boldsymbol{q} - \boldsymbol{w}_l \le \boldsymbol{u}_l \\ &\qquad\qquad\quad \boldsymbol{l}_i \le \boldsymbol{J}_i\Delta\boldsymbol{q} - \boldsymbol{w}_i^* \le \boldsymbol{u}_i,\ i = 1, \ldots, l-1 \end{aligned} \tag{19}$$

where $\boldsymbol{l_i}$ and $\boldsymbol{u_i}$ are lower and upper bounds, $i = 1, \ldots, p$, and an equality constraint is expressed by setting the corresponding lower and upper bound to the same value. Inequality constraints can be handled by the so-called active-set method (see [13]). A constraint is active when it holds as

an equality at the solution and inactive otherwise. The active set contains all equalities and all active inequalities. Inactive constraints do not impact the solution and can be ignored. Between resolutions, the active set can change.

Active set changes have to be considered due to the iterative nature of the BFGS-algorithm (see (11)). The current BFGS-update $\boldsymbol{B}_l^{k-1}$ of level l is a function of the old active set $\mathcal{A}^{k-1}$ since in a previous iteration k_0, $\boldsymbol{B}^{k_0}$ was initialized by $\boldsymbol{B}_l^{k_0} = \cdots + \sum_{i\in\mathcal{A}^{k_0-1}}^{l} \cdots \boldsymbol{I}_i^*$ (see (20)). Without reinitialization, the new BFGS-update $\boldsymbol{B}_l$ would unnecessarily occupy rank (or joints for a better visualization) of the inactive inequalities. These joints then can not be used by tasks on lower hierarchical levels, leading to a solution with higher error norms of these tasks. Therefore, when the new active set $\mathcal{A}$ differs from the old one $\mathcal{A}^{k-1}$ the BFGS-algorithm is reinitialized from the lowest level h, where an active set change occurred, to the last level p by

$$\boldsymbol{B}_l^{k-1} = \boldsymbol{J}_l^{k-1,T}\boldsymbol{J}_l^{k-1} + \sum_{i\in\mathcal{A}}^{l} \max(\underline{\mu}, \frac{1}{2}\|\boldsymbol{e}_i\|_2^2)\boldsymbol{I}_i^* \tag{20}$$

with $l = h, \cdots, p$. Inactive constraints are neglected in the summation. The following initialization

$$\boldsymbol{B}_l^{k-1} = \boldsymbol{J}_l^{k-1,T}\boldsymbol{J}_l^{k-1} + \sum_{i\in\mathcal{A}}^{l} \max(\underline{\mu}, \frac{1}{2}\|\boldsymbol{\lambda}_{i,l}\|_2^2)\boldsymbol{I}_i^* \tag{21}$$

with the Lagrange multipliers $\boldsymbol{\lambda}$ can lead to instability. Especially when constraints are nearly parallel to each other the Lagrange multipliers can grow unlimitedly. Therefore, it is safer to use the physical task error.

IV. SWITCHING STRATEGY BETWEEN GN-ALGORITHM AND QN-METHOD

The analytic expression for $\boldsymbol{H}$ becomes nil inherently for $\sum_{i=1}^{\dim(\boldsymbol{f})} \boldsymbol{e}_i\nabla^2\boldsymbol{f}_i = \boldsymbol{0}$. This is not the case for BFGS but has to be explicitly enforced. Additionally, the GN-algorithm converges quadratically when close to a solution. This is a very desirable quality over the possibly only linear convergence of the QN-method. However, whether the current state is in the proximity of a solution or not needs to be measured accordingly. In this work we propose to observe the model error $\tilde{\Phi} - \Phi$ of the previous iteration. We have that

$$\Phi(\boldsymbol{q}+\Delta\boldsymbol{q}) = \Phi(\boldsymbol{q}) - \Delta\boldsymbol{q}^T\boldsymbol{J}^T\boldsymbol{e} + \frac{1}{2}\Delta\boldsymbol{q}^T(\boldsymbol{J}^T\boldsymbol{J}+\boldsymbol{H})\Delta\boldsymbol{q} + \mathcal{O}(\Delta\boldsymbol{q}^3)$$

which can be calculated by (5). We define $\tilde{\Phi}$ as its approximated model. For the GN case we have

$$\tilde{\Phi}(\boldsymbol{q}+\Delta\boldsymbol{q}) = \Phi(\boldsymbol{q}) - \Delta\boldsymbol{q}^T\boldsymbol{J}^T\boldsymbol{e} + \frac{1}{2}\Delta\boldsymbol{q}^T\boldsymbol{J}^T\boldsymbol{J}\Delta\boldsymbol{q}$$

and for the QN-method we have

$$\tilde{\Phi}(\boldsymbol{q}+\Delta\boldsymbol{q}) = \Phi(\boldsymbol{q}) - \Delta\boldsymbol{q}^T\boldsymbol{J}^T\boldsymbol{e} + \frac{1}{2}\Delta\boldsymbol{q}^T(\boldsymbol{J}^T\boldsymbol{J}+\boldsymbol{R}^T\boldsymbol{R})\Delta\boldsymbol{q}.$$

The approximation error is at least of order $\mathcal{O}(\Delta\boldsymbol{q}^3)$ since the Taylor series of our non-linear model function is limited to second order. However, the error will be dominated by a $\mathcal{O}(\Delta\boldsymbol{q}^2)$ originating from the second order information either being omitted (GN-method) or approximated (QN-method). In practice it is sufficient to observe the quadratic norm residual of the GN-algorithm or QN-method

$$\epsilon = \frac{1}{2}\|\boldsymbol{w}\|_2^2 = \frac{1}{2}\|\boldsymbol{J}\Delta\boldsymbol{q} - \boldsymbol{e}\|_2^2. \tag{22}$$

In case of the slack being zero, the least squares is solved without residual

$$\boldsymbol{f}_d = \boldsymbol{f}(\boldsymbol{q}) + \boldsymbol{J}\Delta\boldsymbol{q}. \tag{23}$$

However, the third order approximation error $\mathcal{O}(\Delta\boldsymbol{q}^3)$ might still be present and we only have

$$\boldsymbol{f}_d \approx \boldsymbol{f}(\boldsymbol{q}+\Delta\boldsymbol{q}). \tag{24}$$

We switch from the QN- to the GN-method whenever ϵ becomes smaller than a certain threshold $\overline{\epsilon}$ (typically 10^{-12}), and from GN to QN when it becomes bigger.

Note that for the case of a GN-QN switch on level c we reinitialize the BFGS from level c to the last level l.

V. TRUST REGION

The Taylor series represents the original function well enough only in a small neighbourhood Δ of the current state $\boldsymbol{q}$. This can be enforced by subjecting either the GN-algorithm or the QN-method to a constraint of the form

$$\|\Delta\boldsymbol{q}\|_\infty < \Delta \tag{25}$$

In practice, this trust region constraint is put on the very first level of the hierarchy as $-\boldsymbol{\Delta} \leq \Delta\boldsymbol{q} \leq \boldsymbol{\Delta}$.

Choosing the trust region $\boldsymbol{\Delta}$ such that a well defined Taylor approximation persists is a delicate process. Self-tuning methods from pure optimization are difficult to transfer to the robot control case. Most importantly, we cannot recalculate solutions with different radii due to the real-time constraint. An important point is also that to the best of our knowledge none of these methods are specifically designed for optimization with hierarchical constraints.

Therefore, we adapt the original trust region Δ with the following method:

- Loop through all variables $i = 1, \cdots, d$ of the solution $\Delta\boldsymbol{q}$ and check if a single entry $\Delta\boldsymbol{q}_i$ changed its sign compared to $\Delta\boldsymbol{q}_i^{k-1}$.
- If so, $\boldsymbol{\eta}_i = \min(\overline{\eta}, \rho_{\text{dec}}^{\boldsymbol{\alpha}_i}\cdot\boldsymbol{\eta}_i)$, and increase $\boldsymbol{\alpha}_i$ by 1.
- If not, $\boldsymbol{\eta}_i = \max(1, 1/\rho_{\text{inc}}\cdot\boldsymbol{\eta}_i)$, and decrease $\boldsymbol{\alpha}_i$ by 1.
- Change the trust region radius for joint i by $\underline{\boldsymbol{\Delta}}_i = -\Delta/\boldsymbol{\eta}_i$ and $\overline{\boldsymbol{\Delta}}_i = \Delta/\boldsymbol{\eta}_i$.

$\overline{\eta}$ is an upper threshold with for example $\overline{\eta} = 10^6$. $\rho_{\text{dec}} = \rho_{\text{inc}} = 1.2$ are the trust region decrease or increase factors. $\underline{\boldsymbol{\Delta}}$, $\overline{\boldsymbol{\Delta}}$, $\boldsymbol{\eta}$ and $\boldsymbol{\alpha}$ are vectors with d entries. $\boldsymbol{\eta}$ and $\boldsymbol{\alpha}$ are initialized with 1's.

It is essential that the original trust region Δ is already chosen as well as possible. From our experience, for every kinematic structure a radius can be found which leads to good convergence for the wide range of applications we tested. In this work, we set $\Delta = 0.01$[rad] or [m].

Algorithm 1 Lexicographic second order augmentation

Input: $\underline{\boldsymbol{B}}, \underline{\boldsymbol{J}}, \underline{\boldsymbol{J}}_{-}, \underline{\boldsymbol{e}}, \underline{\boldsymbol{\lambda}}, \mathcal{A}, c_{\mathcal{A}}, \Delta\boldsymbol{q}$

$\underline{\boldsymbol{y}} = \underline{\boldsymbol{0}}$, $\underline{\boldsymbol{R}} = \underline{\boldsymbol{0}}$
if $t = 0$ **or** $c_{\mathcal{A}} > -1$ **then** $\underline{\boldsymbol{B}}_{c_{\mathcal{A}}} = \boldsymbol{0}$
for all $i \in \mathcal{A}$ **do**
  if $c_{\mathcal{A}} > -1$ **and** $i \geq c_{\mathcal{A}}$ **then** $\boldsymbol{B}_i = \boldsymbol{J}_i^{k-1,T}\boldsymbol{J}_i^{k-1}$
  for all $j \geq i$ **and** $j \in \mathcal{A}$ **do**
    $\boldsymbol{y}_j = \boldsymbol{y}_j + (\boldsymbol{J}_i^T - \boldsymbol{J}_{-,i}^T)\boldsymbol{\lambda}_{i,j}$
    if $c_{\mathcal{A}} > -1$ **and** $i \geq c_{\mathcal{A}}$ **and** $j \in \mathcal{A}$ **then**
      $\boldsymbol{B}_j \mathrel{+}= \max(\underline{\mu}, \frac{1}{2}\|\boldsymbol{e}_i\|^2)\boldsymbol{I}_i^*$
    if $t = 0$ **then** $\boldsymbol{R}_i = \sqrt{\boldsymbol{B}_i}$
    else if $\frac{1}{2}\|\boldsymbol{w}_i\|^2 > \bar{\epsilon}$ **then**
      if $\boldsymbol{y}_i^T\Delta\boldsymbol{q} > \xi$ **then**
        $\boldsymbol{B}_i = \boldsymbol{B}_i + \frac{\boldsymbol{y}_i\boldsymbol{y}_i^T}{\boldsymbol{y}_i^T\Delta\boldsymbol{q}} - \frac{\boldsymbol{B}_i\Delta\boldsymbol{q}\Delta\boldsymbol{q}^T\boldsymbol{B}_i}{\Delta\boldsymbol{q}^T\boldsymbol{B}_i\Delta\boldsymbol{q}}$
        $\boldsymbol{R}_i$ = LDLT$(\boldsymbol{B}_i - \boldsymbol{J}_i^T\boldsymbol{J}_i)$
return $\underline{\boldsymbol{R}}$, $\underline{\boldsymbol{B}}$

VI. Validation

An algorithm overview for the calculation of $\boldsymbol{R}$ named *LexLSAUG2* (**Lex**icographical **L**east **S**quares **Aug**mented with **2**nd order information) is given in Alg. 1. The trust region constraint is on level 0 and is not handled in this algorithm. Any underlined variable contains the values from all levels. $\underline{\boldsymbol{B}}_i$ contains $\boldsymbol{B}$ from level i to the last level l. $c_{\mathcal{A}}$ shows the lowest level where a change of the active set occurred. $c_{\mathcal{A}}$ is set to -1 if there is no change of the active set. LDLT gives back the regularized $\boldsymbol{LDL}^T$ decomposition of the input matrix.

We assess our method GN-QN (which solves with LexLSI [5] the problem (19) possibly augmented by LexLSAUG2) in simulations with two different robots, and compare with three solvers that resolve singularities in hierarchical problems:

1) the GN-algorithm without any Hessian augmentation
2) the hierarchical least squares with adaptive damping (ADLS) described in [1] for hierarchies with any number of levels of equalities. When having one level of equalities, it is equivalent to the LM-algorithm. The maximum damping is set to 2 while the threshold for the minimum singular value is 0.5.
3) the constant damping hierarchical quadratic programming (DHQP) described in [10], which handles both equalities and inequalities. For equalities only problems DHQP is identical to ADLS with constant damping. The damping is set to 2.

A. Test bench

We tested our method with 20 different test cases. Their length is limited to 25000 iterations.

The test cases **T1** to **T19** are performed on a 2D 4 DoF robot with two end-effectors (see Fig. 1). It possesses a translational DoF at its base and three revolute joints, one at the base and two at its 'shoulder' enabling the two arms to rotate. All links are of length 1[m]. Their task hierarchy then looks as follows:

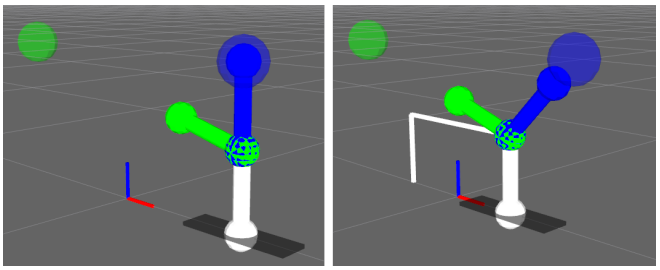

Fig. 1. Robot for **T1** (left) and **T2** (right) for GN-QN. The robot can translate along the long axis of the black slider. The other joints are revolute. The green end-effector is tracking the green target while the blue end-effector is tracking the blue target. In both frames, the targets are at their turning points $[2, 2]$[m] (blue ball) and $[-2, 2]$[m] (green ball). The robot is able to just reach the blue target for T1 while for T2 both targets remain unreachable. This is due to the position of the shoulder being bound inside the white box.

Hierarchy **A** (T1, T3 to T19) and **B** (T2) (2D robot)

0) 4 trust region limits $\underline{\boldsymbol{\Delta}} \leq \boldsymbol{I}\dot{\boldsymbol{q}} \leq \overline{\boldsymbol{\Delta}}$ (GN and GN-QN) or 4 velocity limits $\underline{\dot{\boldsymbol{q}}}_{vl} \leq \boldsymbol{I}\dot{\boldsymbol{q}} \leq \overline{\dot{\boldsymbol{q}}}_{vl}$ (DHQP)
1) 2 equality constraints on blue end-effector as follows: $\boldsymbol{J}_{\text{ef}_2}\dot{\boldsymbol{q}} = \boldsymbol{f}_d^{\text{ef}_2}$
2) 2 equality constraints on green end-effector as follows: $\boldsymbol{J}_{\text{ef}_3}\dot{\boldsymbol{q}} = \boldsymbol{f}_d^{\text{ef}_3}$
3) Minimal norm solution $\dot{\boldsymbol{q}} = 0$

For hierarchy **B**, an inequality constraint on the white end-effector / 'shoulder' $\boldsymbol{J}_{\text{ef}_1}\dot{\boldsymbol{q}} \in \boldsymbol{f}_d^{\text{ef}_1}$ is added between level 0 and 1 of hierarchy A.

For ADLS, the joint velocity constraint is omitted since inequalities can not be handled. The initial robot configuration is $[0, 0]$[m] for the free flyer and $[-\pi/2, 0, 0]$[rad] for the revolute joints.

The end-effectors at the tip of the arms must track two targets oscillating diagonally with amplitude $[2, 2]$[m] and crossing at $[0, 0]$[m]. This means that for **T1** the targets are just reachable for the robot in the corner cases. For **T3**, the targets oscillate with amplitude $[2, 3]$[m]. For **T2** an additional inequality constraint (hierarchy **B**) is imposed on the position of the shoulder. The constraint box reaches from $[-1, 0]$[m] to $[1, 1]$[m].

For **T4** to **T8**, we use static targets for level 1 and 2 at $[0, y]$[m] and $[x, 1]$[m] with (x, y) equal to $(1, 2)$, $(1+\epsilon, 2+\epsilon)$, $(1-\epsilon, 2-\epsilon)$, $(1+10, 2+10)$ and $(1-0.25, 2-0.25)$. The three first case are slight variations to test robustness with $\epsilon = 0.001$[m] (1/1000 of the link length) such that both targets are either just out of reach or just in reach. The two last cases test for targets well out of reach or fully reachable.

For **T9**, both targets change their position randomly within $[\pm 2, \pm 1000]$[m] at each of the first 12500 iterations, and then are static at $[-1.9, -422]$[m] and $[-1.5, 299]$[m] respectively for the rest of the test. This test shows that the second-order approximation can deal with highly noisy targets.

For **T10** to **T14**, the targets are at $[t, y]$[m] and $[-2, 2]$[m] where t increases linearly by 0.001 with the number of iterations, and y takes the same value as for T4 to T8.

For **T15** to **T19**, the first target is at $[0, y]$[m], with y defined as above, while the second target oscillates as in T1.

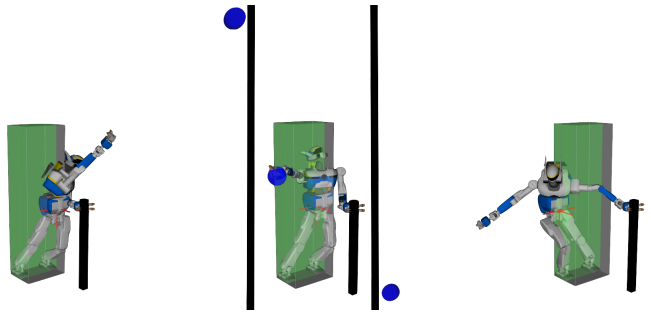

Fig. 2. **T20** screenshots for GN-QN, from left to right: HRP2 reaching for the target on its top left (out of reach, both CoM tasks are active), then the target is in reach (the left hand task is switched to the GN-algorithm) and finally the target is out of reach again in HRP2's far right bottom. The accompanying video also shows how the robot tries to stay as long as possible in the smaller (green) CoM bounding box on the last level as it is tracking the target (made possible by switching to the GN-algorithm when the target is in reach, and which can not be observed for DHQP).

T10 to T19 show that our method is capable of tracking static and dynamic targets at the same time while interchanging their priority and testing for robustness.

T20 shows that our method is scalable to any number of DoF. The robot HRP-2Kai with 38 DoF is standing in a stable position in a cluttered aircraft workspace environment (see accompanying video). Both feet and the left hand are given fixed positions either on the ground or grabbing a pole in front of the robot. On the next level the centre of mass (CoM) projection is asked to remain in a rectangle a bit larger than the polygon spanned by the feet. The right hand is then used to track a target swinging from the robot's far top left to its far right bottom. At the lowest priority level, we give a smaller polygon for the CoM projection, corresponding to the feet only. The hierarchy then looks as follows:

Hierarchy **C** (T20) (HRP-2Kai)

0) 38 trust region limits $\underline{\boldsymbol{\Delta}} \leq \boldsymbol{I}\dot{\boldsymbol{q}} \leq \overline{\boldsymbol{\Delta}}$ (GN and GN-QN) or 38 velocity limits $\underline{\dot{\boldsymbol{q}}}_{vl} \leq \boldsymbol{I}\dot{\boldsymbol{q}} \leq \overline{\dot{\boldsymbol{q}}}_{vl}$ (DHQP)

1) 38 joint limits $\underline{\dot{\boldsymbol{q}}}_{jl} \leq \boldsymbol{I}\dot{\boldsymbol{q}} \leq \overline{\dot{\boldsymbol{q}}}_{jl}$

2) 18 in-reach equality constraints (translation, rotation) on left, right foot and left hand: $\boldsymbol{J}_{\text{lf}}\dot{\boldsymbol{q}} = \boldsymbol{f}_d^{\text{lf}}$, $\boldsymbol{J}_{\text{rf}}\dot{\boldsymbol{q}} = \boldsymbol{f}_d^{\text{rf}}$, $\boldsymbol{J}_{\text{lh}}\dot{\boldsymbol{q}} = \boldsymbol{f}_d^{\text{lh}}$

3) 3 inequality constraints on the CoM: $\boldsymbol{J}_c\dot{\boldsymbol{q}} \in \boldsymbol{f}_d^{\text{c1}}$

4) 3 in/out-of-reach equality constraints (translation, oscillating target in 3D-space) on right hand: $\boldsymbol{J}_{\text{rh}}\dot{\boldsymbol{q}} = \boldsymbol{f}_d^{\text{rh}}$

5) 3 stricter inequality constraints on the CoM: $\boldsymbol{J}_c\dot{\boldsymbol{q}} \in \boldsymbol{f}_d^{\text{c2}}$

6) Minimal norm solution $\dot{\boldsymbol{q}} = 0$

B. Evaluation criteria

The main problem that can arise near a singularity is oscillations. We measure the presence of oscillations by tracking the change of sign between iterations for each component, and summing the amplitude of the change.

$$\Sigma = \sum_{k=2}^{25000} \sum_{i=1}^{d} \begin{cases} |\Delta \boldsymbol{q}_i^k| & \text{if } \operatorname{sgn}(\Delta \boldsymbol{q}_i^k) \neq \operatorname{sgn}(\Delta \boldsymbol{q}_i^{k-1}) \\ 0 & \text{otherwise} \end{cases}$$

A low Σ means few or no oscillations.

We also integrate the normed error of the equality with highest priority ('ef_2' task for hierarchy A and B, 'rh' task for hierarchy C) over time and then normalize it by the smallest value of all solvers for this test case (Ξ). The performance of the other tasks is not considered since bad convergence of a task with high priority can lead to better convergence of a task with lower priority, then lacking of any relevance. Since the joint velocity constraint is omitted for ADLS and thus allowing for way faster convergence from the initial position to the targets its performance on Ξ is not considered (grey). For static test cases, the iteration where every entry of the solution vector becomes smaller than 10^{-6}[rad] or [m] is noted (Ψ).

C. Evaluation

The evaluation results of all test cases are given in Table I. Our method is stable (low Σ) and performs best in terms of convergence ($\Xi \approx 1$). While DHQP and ADLS seem to be unstable for many test cases, it is very important to mention that the damping can be tuned in such a way that $\Sigma = 0$[rad] (possibly with very high damping terms for T10 to T15, and thus with very slow convergence). Because this has to be done on a per-case basis, and after running a test, in our work the damping is chosen such that $\Sigma = 0$[rad] for T1. These values are applied for all the other test cases to enable a fair comparison for the convergence of T1, T2 and T20.

The test cases show the following qualities of our method:

- Stability in the presence of singularities
  - when the target is far away (T7, T9, T10, T11, T12, T13, T14, T18)
  - when the target is at the border of reachability (T1, T4, T5, T6, T10, T11, T12, T15, T16, T17)
- Fast and good convergence properties
- Stable for all test cases without the need for damping tuning.
- Works for inequalities (T2, T20)
- Scalable, applicable for any number of DoF (T20)
- Strong self-regulating capabilities (T9)
- The hierarchy is not disturbed as it is the case with the damping term which introduces a model discontinuity when the rank of a matrix changes and the damping is not applied in the same null-space (T15, T16).

More detailed results for **T1** are given in Fig. 3 showing the task error norms. GN-QN tracks the level 1 target very well due to the switching to the GN-algorithm allowing for quadratic convergence. This behaviour can also be observed at times for ADLS when the damping term becomes zero. For level 2, DHQP and ADLS have lower task error norms at times but this is due to their worse convergence of level 1, putting the level 2 end-effector closer to its target. In general, DHQP converges the worst due to the constantly present damping.

The GN-algorithm is unstable with a high $\Sigma = 574$[rad]. It shows that the GN-algorithm alone is not a safe method for a real world robot application.

For **T2**, the task error norms are given in Fig. 4. GN-QN and DHQP track the level 2 target very well with slight

	T1	T2	T3	T4	T5	T6	T7	T8	T9	T10
GN-QN	0.3 / **1**	**0.06** / **1**	**0.3** / **1.**	**0** / 1.-201	**0** / **1**-169	**0** / **1**-171	**0** / **1**-162	**0** / **1**-210	**0.1** / **1**-13k	**0.02** / **1**
DHQP	**0** / 2.4	0.3 / 1.	284 / 1.1	**0** / 1.3-15k	**0** / 1.4-17k	**0** / 1.3-10k	828 / 21	**0** / 1.1-280	470 / 10	808 / 4.8
ADLS	**0** / 0.5	-	$4 \cdot 10^6$ / 10	**0** / 0.3-15k	**0** / 0.5-17k	**0** / 0.3-10k	10^5 / 30	**0** / 0.1-29	$3 \cdot 10^3$ / 1.1	10^5 / 600
GN	574 / 1.	155 / 1.2	547 / 1	**0** / **1**-169	10^3 / 1.2	**0** / **1**-163	10^3 / 21	**0** / **1**-205	551 / 10	912 / 1.02
	T11	**T12**	**T13**	**T14**	**T15**	**T16**	**T17**	**T18**	**T19**	**T20**
GN-QN	**0** / **1**	$\mathbf{10^{-6}}$ / **1**	**0** / **1**	**0** / **1**	**0.03** / **1**	**0** / **1**	**0.1** / **1**	**0** / **1**	0.02 / **1**	0.7 / **1**
DHQP	809 / 4.6	806 / 4.8	743 / 1.	693 / 2.1	7.4 / 1.4	12 / 1.3	1.9 / 1.3	605 / 1.	**0** / 1.	**0.2** / 1.1
ADLS	10^5 / 600	10^5 / 600	10^5 / 1.4	10^5 / 600	4.6 / 0.4	6.5 / 0.5	1.9 / 0.3	10^5 / 1.4	**0** / 0.1	-
GN	821 / 1.01	955 / 1	966 / 1.	849 / 1	775 / 1.	696 / 1.	890 / 1.	971 / 1.	516 / 1	2000 / 1.

TABLE I

TABLE ENTRIES ARE OF FOLLOWING FORM: Σ / Ξ - Ψ (ONLY FOR STATIC CASES AND WHEN SMALLER THAN 25000); RED COLOURED ENTRIES MEAN THAT THE TEST CASE WAS UNSTABLE (HIGH Σ). BLUE COLOURED ENTRIES MEAN THAT THIS SOLVER PERFORMED BEST FOR THIS TEST CASE.

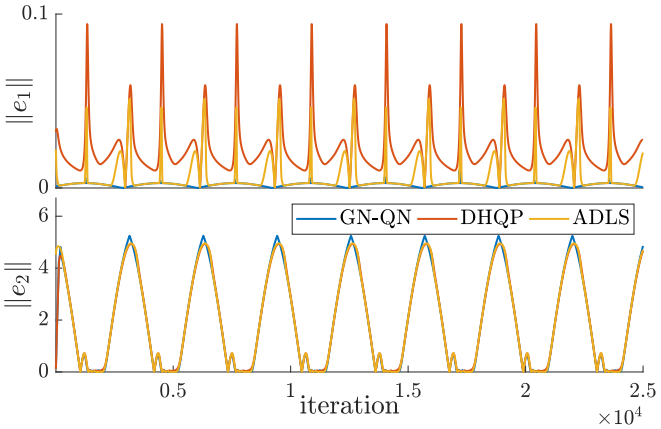

Fig. 3. Hierarchy **T1**, norm of task errors for level 1 and 2 with different methods.

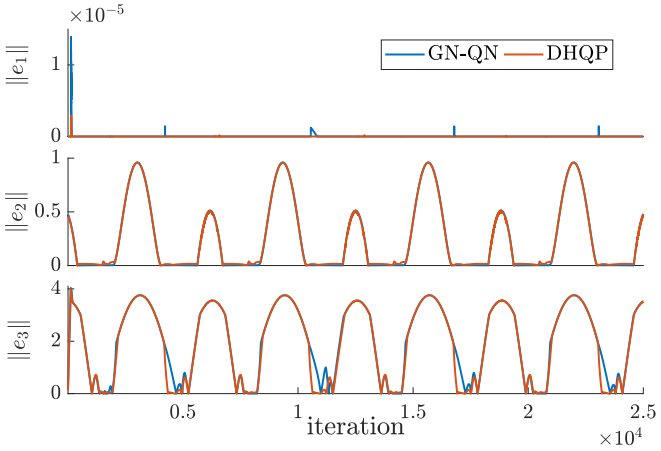

Fig. 4. **T2**, norm of task errors for level 1, 2 and 3 for GN-QN and DHQP.

advantage for GN-QN. The inequality constraint is violated at some instances for both methods at a low amplitude. For level 3, DHQP has a lower task error norm at times but this is due to the worse convergence of level 2, putting the level 3 end-effector closer to its target.

For the static test cases **T4** to **T8**, GN-QN behaves in a stable manner while DHQP becomes unstable when the targets are at $[0, 2 + 10]$[m] and $[1 + 10, 1]$[m]. Especially, DHQP converges exponentially whereas the solution of GN-QN decreases much faster to numerical zero by switching to the GN-algorithm.

For **T9** and GN-QN, the robot does not follow the random

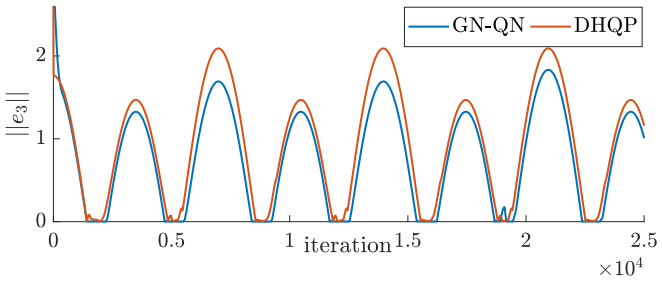

Fig. 5. **T20**, norm of task error for the right hand on hierarchy level 3.

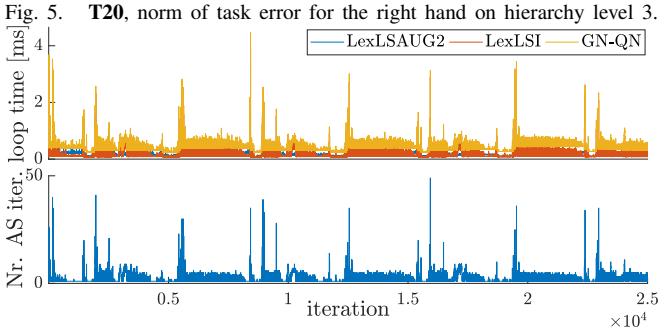

Fig. 6. **T20**, control loop times in [ms]. The maximum loop times are 4.5[ms] for GN-QN, 4.0[ms] for LexLSI and 0.5[ms] for LexLSAUG2. The maximum number of active set iterations in LexLSI is 49.

noise of the targets. This is due to the trust region adaptation method.

For **T10** to **T14**, GN-QN is stable while DHQP is only stable in the beginning and becoming unstable as the robot moves away from the second target.

For **T15** to **T16**, it can be seen (video) how the hierarchy is violated by DHQP. The first end-effector is reaching for the target at $[0, 2]$[m] and $[0, 2 + 0.001]$ respectively. The robot is kinematically able to reach $[0, 2]$. While for GN-QN the first three joints do not move since they are fully occupied by the level 1 task, for DHQP these joints visibly move as they are involved in the achievement of the level 2 task.

For **T20**, both GN-QN and DHQP track the target in a stable manner. Fig. 5 shows the better convergence of GN-QN of the right hand (the remaining tasks are not shown since they are either converged onto their targets or within their prescribed bounds, for both methods). Especially when the target is in reach GN-QN has near zero task error norm due to the switching to the GN-algorithm. DHQP only gets into the proximity of the reachable target.

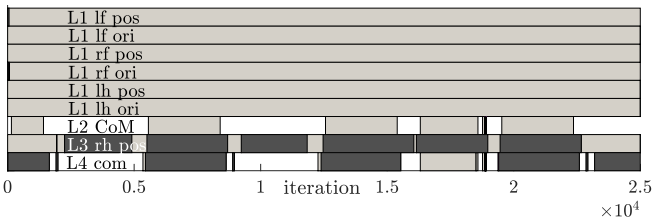

Fig. 7. **T20**, map of activity (light gray) and QN-method (dark gray).

The activation of the tasks and whether they are augmented or not is shown in Fig. 7. If a task is augmented it also means that it is active.

The computation speed on an Intel Core i7-4720HQ CPU @ 2.60GHz with 8 GB of RAM is given in Fig. 6. The computation times lay usually well below 1[ms] even when several tasks are augmented with second order information. However, there are some peaks up to 4.5[ms], especially when either the right hand or the CoM task are switched to the QN-method and a lot of active set iterations take place in LexLSI. This number of iterations seems too large, and we are investigating it. We do not have an optimized version of DHQP for fair comparison, but for a similar size of problem and similar hardware, [10] reports a computation time of about 3[ms]. This is coherent with the timing difference reported in [4] between a dedicated solver and a cascade of LSP.

VII. Conclusion

We proposed a new robust method to deal with singularities in prioritized inverse kinematics control schemes. We showed that we are able to solve any robotic setup with any number of hierarchical levels of equalities and inequalities, may they be feasible or not, while most importantly, ensuring numerical stability. Thereby, our approach provides higher accuracy w.r.t current state-of-the-art methods. Furthermore, it allows the use of the fastest off-the-shelf hierarchical least-squares solvers.

Our approach borrows much inspiration from constrained optimization: the Quasi-Newton method avoids the expensive calculation of the analytic Hessian while being stable, highly accurate with at least linear convergence. Due to our adaptation method, we switch reliably to the quadratically converging GN-algorithm whenever a problem becomes feasible. Thereby, the GN-algorithm and the QN-method are solved by the same fast state of the art hierarchical linear least-squares solver due to proper reformulation.

We observed that the QN-method is very forgiving in terms of simplifications in the second order information (we omitted the reduction of the Hessian by $\boldsymbol{J}^T\boldsymbol{J}$). As long as the second order information is positive definite we will converge at least linearly. However, it would be desirable to either directly update $\boldsymbol{H}$ or even its decomposition $\boldsymbol{R}$ like in [21] in order to improve the convergence behaviour and save computation time.

Our ongoing work is dedicated to extending our method to dynamic control.

References

[1] S. Chiaverini, "Singularity-robust task-priority redundancy resolution for real-time kinematic control of robot manipulators," *IEEE Transactions on Robotics and Automation*, vol. 13, no. 3, pp. 398–410, 1997.

[2] B. Siciliano and J.-J. E. Slotine, "The general framework for managing multiple tasks in high redundant robotic systems," in *International Conference on Advanced Robotics*, 1991, pp. 1211 – 1216 vol.2.

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

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

[5] D. Dimitrov, A. Sherikov, and P.-B. Wieber, "Efficient resolution of potentially conflicting linear constraints in robotics," Aug. 2015, submitted to IEEE TRO (05/August/2015). [Online]. Available: https://hal.inria.fr/hal-01183003

[6] Y. Nakamura and H. Hanafusa, "Inverse Kinematic Solutions With Singularity Robustness for Robot Manipulator Control," *J. Dyn. Sys., Meas., Control*, vol. 108, no. 3, pp. 163–171, 1986.

[7] A. A. Maciejewski and C. A. Klein, "Numerical filtering for the operation of robotic manipulators through kinematically singular configurations," *Journal of Robotic Systems*, vol. 5, no. 6, pp. 527–552, 1988.

[8] S. R. Buss and J.-S. Kim, "Selectively damped least squares for inverse kinematics," *Journal of Graphics Tools*, vol. 10, no. 3, pp. 37–49, 2005. [Online]. Available: https://doi.org/10.1080/2151237X.2005.10129202

[9] W. Wolovich and H. Elliott, "A computational technique for inverse kinematics," *IEEE Conference on Decision and Control*, vol. 23, no. December, pp. 1359–1363, 1984.

[10] A. Herzog, N. Rotella, S. Mason, F. Grimminger, S. Schaal, and L. Righetti, "Momentum control with hierarchical inverse dynamics on a torque-controlled humanoid," *Autonomous Robots*, vol. 40, no. 3, pp. 473–491, Mar 2016. [Online]. Available: https://doi.org/10.1007/s10514-015-9476-6

[11] A. S. Deo and I. D. Walker, "Adaptive non-linear least squares for inverse kinematics," in *IEEE International Conference on Robotics and Automation*, vol. 1, May 1993, pp. 186–193.

[12] P.-B. Wieber, A. Escande, D. Dimitrov, and A. Sherikov, "Geometric and numerical aspects of redundancy," in *Geometric and Numerical Foundations of Movements*, 2017. [Online]. Available: https://hal.inria.fr/hal-01418462

[13] J. Nocedal and S. J. Wright, *Numerical Optimization*, 2nd ed. New York, NY, USA: Springer, 2006.

[14] J. E. Dennis, Jr., D. M. Gay, and R. E. Walsh, "An adaptive nonlinear least-squares algorithm," *ACM Trans. Math. Softw.*, vol. 7, no. 3, pp. 348–368, Sep. 1981. [Online]. Available: http://doi.acm.org/10.1145/355958.355965

[15] P. L. Toint, "On large scale nonlinear least squares calculations," *Siam Journal on Scientific and Statistical Computing*, vol. 8, 05 1987.

[16] C. G. Broyden, "The Convergence of a Class of Double-rank Minization Algorithms," *Journal of the Mathematics and its Applications*, vol. 6, pp. 76–90, 1970.

[17] J. Zhao and N. I. Badler, "Inverse kinematics positioning using nonlinear programming for highly articulated figures," *ACM Trans. Graph.*, vol. 13, no. 4, pp. 313–336, Oct. 1994. [Online]. Available: http://doi.acm.org/10.1145/195826.195827

[18] T. Sugihara, "Solvability-unconcerned inverse kinematics by the levenbergmarquardt method," *IEEE Transactions on Robotics*, vol. 27, no. 5, pp. 984–991, October 2011.

[19] J. R. Bunch, L. Kaufman, and B. N. Parlett, "Decomposition of a symmetric matrix," *Numerische Mathematik*, vol. 27, no. 1, pp. 95–109, 1976.

[20] J. J. Moré and D. C. Sorensen, "On the use of directions of negative curvature in a modified newton method," *Mathematical Programming*, vol. 16, no. 1, pp. 1–20, 1979.

[21] R. Fletcher, "A new low rank quasi-Newton update scheme for nonlinear programming," *IFIP TC7 Conference*, no. August, pp. 275–293, 2006.