Singularity Resolution for Multi-Level Constrained Dynamically Feasible Kinematic Control

Kai Pfeiffer, Adrien Escande, Pierre Gergondet, Abderrahmane Kheddar

Singularity Resolution for Multi-Level Constrained Dynamically Feasible Kinematic Control

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

***Abstract*— With this work we introduce the notion of physical feasibility into our previously presented method of kinematic and algorithmic singularity resolution for kinematics based robotic control. We begin by deriving Newton's method of multi-level constrained optimization and give a suitable expression for the hierarchical analytic Hessian. The link between optimization and robotic control is then created. We proceed to first reveal the difficulties of damping approaches for singularity resolution in acceleration based control. Consequently, Newton's method and the previously presented Quasi-Newton method are only well-defined in the velocity domain. This requires the conversion of the second-order equation of motion and motion controllers into first order by suitably applying forward integration. This way we achieve dynamically feasible kinematic control while being robust towards singularities by switching from the Gauss-Newton algorithm to the Newton's method using a reliable switching method. We verify our approach in three robot experiments on the HRP-2KAI humanoid robot where we supersede a classical damping based constrained optimization controller in terms of accuracy. Furthermore, the need for damping tuning is discarded. Thereby, the least-squares formulation of the Gauss-Newton algorithm and the Newton's method greatly facilitates real-time control by enabling the use of fast state-of-the art hierarchical quadratic programming solvers.**

I. Introduction

In [1] a novel solver was devised which allows multiple task-space objectives formulated as optimization-based controllers to be ordered in a strict hierarchy and solved very efficiently. A further improvement on efficiency has been introduced with the work of [2]. However, the evaluation of these powerful solvers was confined to simulation or well-calibrated experiments. The reason is that singularities (whose nature is detailed later) generate unpredictable instabilities and risky behaviours [3]. With this paper we aim at solving this drawback in order to provide end-users designing robotic motions a safe control framework. Therefore it is very important to be able to specify multiple task-space objectives and constraints without worrying about their potential conflicts and resulting singularities [4]. This speeds up the programming of robots for new operations especially in industrial settings. According to our industrial partners, this can take from three up to six months depending on the complexity of the robot at hand and the type of operations. Here end-users are requested to specify a set of usual tasks (set-point, tip force control, trajectory tracking...) [5], [6] under various predefined constraints (joint limits...) or constraints updated from online sensor readings especially when concerned with the stability of the robot [7], [8], [9], [10].

In classical constrained-quadratic-programming-based control approaches, such tasks can be specified in a two-level hierarchy where the equation of motion and all the corresponding dynamics constraints are put on the constraints level, while the robot control is put on the objective level, see e.g. [11]. With the rise of new hierarchical solvers [12], [1], [13] the robot control can handle more priority levels, for example respecting joint limits over the control of any other task. This allows designing very safe controllers, strictly prioritizing safety, possible stability constraints and objective tasks. Additionally, general hierarchies can be handled in a very efficient manner, superseding soft (or weighted) 2-level hierarchies in terms of computational effort [2].

Problems arise however, if tasks on different levels of the hierarchy get in conflict. Such algorithmic singularities need to be resolved alongside kinematic singularities. Otherwise the nearly rank deficient Jacobian –or its projection onto the Jacobians of higher priority tasks in case of algorithmic singularities– leads to numerical instability with high joint velocities. Kinematic singularities can be predicted and prevented with an analytical robot workspace analysis [14], [15], [16], [17]. Algorithmic singularities however depend on the conflict with higher priority tasks at the current robot configuration and therefore are harder to analyse [18]. Furthermore, hierarchical solvers are intended to be employed on humanoid robots in any industrial setting like Airbus airplane assembly halls. Engineers without deeper understanding of these issues should be able to set up robot problems safely and easily, even if for example sensor readings give targets that are outside of the feasible workspace of the robot.

One remedy would be to introduce a weighted regularization term [3], [19], [20], [18], [21], [22], [23], [24] on the joint velocities (also referred to as the Levenberg-Marquardt (LM) algorithm or damped-least-squares).

Damping the joint velocities can be interpreted as approximating the second order derivatives of the Taylor expansion of the quadratic task error norm [25], [26] as a weighted identity matrix. However, in our previous work [27] we showed that using an approximation of the true second order derivatives by the Broyden [28], Fletcher [29], Goldfarb [30] and Shanno [31] (BFGS) method yields better convergence. Additionally, the tuning of the damping parameter is not straightforward [32].

This previously presented Quasi-Newton method was only

[1] CNRS-AIST Joint Robotics Laboratory, JRL UMR3218/RL, Tsukuba, Japan.

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

aimed at resolving singularities in hierarchical kinematics based control problems. While kinematic control of robots can be sufficient for fixed base robots [33], [34], this does not necessarily hold for legged humanoids with an un-actuated free-flyer base and unilateral friction contacts. Only with a model of the forces and torques acting on the robot's body it can aim to maintain a physically stable posture [35]. Therefore, we present ways to include the equation of motion and dynamics constraints into our scheme to guarantee physically feasible motions.

This work then presents the following new contributions:

- The hierarchical Newton's method is presented by giving an expression for the hierarchical analytical Hessian (see sec. III). In the following we refer to both the Quasi-Newton method and Newton's method as "Newton's method";
- We adapt the Gauss-Newton (GN) algorithm and Newton's method, which are both tools from optimization, to control (see sec. IV);
- We show how regularization terms like damping negatively influence the exponential convergence of second-order motion controllers (see sec. V-A);
- Second-order motion controllers are then suitably adapted to fit into the velocity based Newton's method of control (see sec. V-B);
- The dynamics in form of the equation of motion are adapted accordingly and then integrated into the hierarchical control scheme (see sec. V-C);
- Experimental assessment of our developments with the HRP-2Kai humanoid robot (see sec. IX).

II. Preliminaries

The dynamics of a robot composed of rigid links is determined by the equation of motion

$$\boldsymbol{M}(\boldsymbol{q})\ddot{\boldsymbol{q}} + \boldsymbol{N}(\boldsymbol{q}, \dot{\boldsymbol{q}}) = \boldsymbol{S}^T\boldsymbol{\tau} + \boldsymbol{J}_c^T\boldsymbol{\gamma}\,. \tag{1}$$

$\boldsymbol{q}$ represents the configuration of the robot's joints and free-flyer base if any. The equation is non-linear in $\boldsymbol{q}$ and $\dot{\boldsymbol{q}}$ but linear in the accelerations $\ddot{\boldsymbol{q}}$, torques $\boldsymbol{\tau}$, and the generalized contact wrenches $\boldsymbol{\gamma}$. $\boldsymbol{M}(\boldsymbol{q})$ is the whole-body inertia matrix, $\boldsymbol{N}(\boldsymbol{q}, \dot{\boldsymbol{q}})$ is the Coriolis and gravity vector and $\boldsymbol{J}_c$ the contact points' Jacobian matrix. $\boldsymbol{S}$ is a selection matrix to exclude the unactuated free-flyer.

On the other hand we have motion controllers $\ddot{\boldsymbol{e}}^{\text{ctrl}} = -\boldsymbol{J}\ddot{\boldsymbol{q}} - \dot{\boldsymbol{J}}\dot{\boldsymbol{q}}$ (ctrl: control) where we want to drive some non-linear geometric error function $\boldsymbol{e}(\boldsymbol{q}) = \boldsymbol{f}_d(t) - \boldsymbol{f}(\boldsymbol{q})$ to zero ($\boldsymbol{e} : \mathbb{R}^n \to \mathbb{R}^m$, with sufficient continuity properties). $\ddot{\boldsymbol{e}}^{\text{ctrl}} = -k_p\boldsymbol{e} - k_v\dot{\boldsymbol{e}}$ is given by a Proportional Derivative (PD) controller with positive gains k_p and k_v. We define $\boldsymbol{J} = \nabla_{\boldsymbol{q}}\boldsymbol{f} \in \mathbb{R}^{m,n}$ so $\nabla_{\boldsymbol{q}}\boldsymbol{e} = -\boldsymbol{J}$.

Now, we aim at solving a hierarchical dynamic control problem with p levels [36], [37], [1], [13]

$$\underset{\boldsymbol{x}^{(k+1)}, \boldsymbol{w}_i^{(k+1)}}{\text{lexmin.}} (\|\boldsymbol{w}_1^{(k+1)}\|^2, \|\boldsymbol{w}_2^{(k+1)}\|^2, \cdots, \|\boldsymbol{w}_p^{(k+1)}\|^2) \tag{2}$$

$$\text{s.t.} \quad \boldsymbol{A}_i^{(k)}\boldsymbol{x}^{(k+1)} + \boldsymbol{b}_i^{(k)} \;\leqq\; \boldsymbol{w}_i^{(k+1)} \qquad i = 1..p\,. \tag{3}$$

at the control time step k for the new robot state $\boldsymbol{x}^{(k+1)} = \begin{bmatrix}\ddot{\boldsymbol{q}}^{(k+1),T} & \boldsymbol{\tau}^{(k+1),T} & \boldsymbol{\gamma}^{(k+1),T}\end{bmatrix}^T$. Symbol $\leqq$ gathers equality ($=$) and inequality ($\leq$) constraints. $\boldsymbol{A}^{(k)}$ and $\boldsymbol{b}^{(k)}$ are determined by the equation of motion or the motion objectives and $\boldsymbol{w}_i^{(k+1)} \in \mathbb{R}^m$ is a slack variable which relaxes infeasible objectives for example due to task conflict.

In the following we drop the index k to simplify the writing but keep the indices $k-1$, $k+1$.

The problem can also be formulated as a sequence of program for $l = 1..p$

$$\min_{\boldsymbol{x}, \boldsymbol{w}_l} \quad \frac{1}{2}\|\boldsymbol{w}_l^{(k+1)}\|^2 \qquad l = 1..p \tag{4}$$

$$\text{s.t.} \quad \boldsymbol{A}_l\boldsymbol{x}^{(k+1)} + \boldsymbol{b}_l \;\leqq\; \boldsymbol{w}_l^{(k+1)} \tag{5}$$

$$\underline{\boldsymbol{A}}_{l-1}\boldsymbol{x}^{(k+1)} + \underline{\boldsymbol{b}}_{l-1} \;\leqq\; \underline{\boldsymbol{w}}_{l-1}^{*,(k+1)} \qquad . \tag{6}$$

$\underline{\boldsymbol{w}}_{l-1}^{*,(k+1)}$ are the optimal values obtained from solving the problems for $i < l$. An underlined vector $\underline{\boldsymbol{v}}_r$ (or $\underline{\boldsymbol{v}}_{r,u}$) or matrix $\underline{\boldsymbol{V}}_r$ is the stacked vector $\begin{bmatrix}\boldsymbol{v}_1^T & \cdots & \boldsymbol{v}_r^T\end{bmatrix}^T$ or stacked matrix $\begin{bmatrix}\boldsymbol{V}_1^T & \cdots & \boldsymbol{V}_r^T\end{bmatrix}^T$.

As it is, the above hierarchy cannot deal intrinsically with kinematic singularities of the motion objectives, nor with objectives that are conflicting altogether, resulting in algorithmic singularities. In order to resolve these singularities we take a step back from control to optimization where we derive the Gauss-Newton (GN) algorithm and Newton's method for lexicographically constrained optimization problems (see sec. III).

Then, our derived optimization methods are put back to kinematics control (see sec. IV). Our idea is that the GN algorithm –that can be interpreted as the solver of the motion control objectives– doesn't approximate sufficiently well the original non-linear geometric function around singularities. This is due to neglecting second order information, making a switch to Newton's method necessary. The switching method proposed in our previous work is briefly synthetized in sec. VII.

In sec. V we show how a dynamic control problem as given in (6) can be expressed at the velocity level. This is necessary because Newton's method in acceleration-based control leads to overshooting behaviour without tricky adaptive control gain tuning (see sec. V-A).

We present several methods to compute the hierarchical Hessian for Newton's method, be it analytically (see sec. VI-B) or by BFGS approximations (see sec. VI-A and sec. VI-C). Lastly, we implemented the theory in our controller software with architecture integration to validate our approaches in three complex experiments with the HRP-2Kai (see sec. IX).

III. Minimizing a non-linear geometric function

Let us consider the problem from a pure optimization viewpoint. In order to derive the hierarchical GN algorithm and Newton's method we start with the following non-linear

least-squares problem

$$\min_{\boldsymbol{q},\boldsymbol{w}_l} \quad \frac{1}{2}\left\|\boldsymbol{w}_l\right\|^2 \qquad l=1..p \tag{7}$$

$$\text{s.t.} \quad \boldsymbol{e}_l(\boldsymbol{q}) \leqq \boldsymbol{w}_l \tag{8}$$

$$\underline{\boldsymbol{e}}_{l-1}(\boldsymbol{q}) \leqq \underline{\boldsymbol{w}}^*_{l-1}\,. \tag{9}$$

The goal is to drive the constraint violation $\boldsymbol{w}_l$ of each level l to zero 'at best' such that the task error $\boldsymbol{e}_l(\boldsymbol{q})=\boldsymbol{0}$. Already obtained optimal violations of previous levels $\underline{\boldsymbol{w}}^*_{l-1}$ must stay unchanged.

If $p=1$ and with the presence of equalities only we get

$$\min_{\boldsymbol{q},\boldsymbol{w}} \quad \frac{1}{2}\left\|\boldsymbol{w}\right\|^2 \tag{10}$$

$$\text{s.t.} \quad \boldsymbol{e}(\boldsymbol{q})=\boldsymbol{w}\,. \tag{11}$$

The first order optimality condition of this problem is

$$\nabla_{\boldsymbol{q},\boldsymbol{w},\boldsymbol{\lambda}}\mathcal{L}=\boldsymbol{K}(\boldsymbol{q},\boldsymbol{w},\boldsymbol{\lambda})=\begin{bmatrix}\boldsymbol{J}^T\boldsymbol{\lambda}\\ \boldsymbol{w}+\boldsymbol{\lambda}\\ \boldsymbol{w}-\boldsymbol{e}\end{bmatrix}=\boldsymbol{0} \tag{12}$$

with the Lagrangian $\mathcal{L}=\frac{1}{2}\boldsymbol{w}^T\boldsymbol{w}+\boldsymbol{\lambda}^T(\boldsymbol{w}-\boldsymbol{e})$ and the vector of Lagrange multipliers $\boldsymbol{\lambda}\in\mathbb{R}^m$ (recall that $\nabla_{\boldsymbol{q}}\boldsymbol{e}=-\boldsymbol{J}$).

Applying a Newton step $\boldsymbol{K}(\boldsymbol{x}+\boldsymbol{\Delta x})=\boldsymbol{K}(\boldsymbol{x})+\nabla_{\boldsymbol{x}}\boldsymbol{K}(\boldsymbol{x})\boldsymbol{\Delta x}=\boldsymbol{0}$ with $\boldsymbol{x}^T=\begin{bmatrix}\boldsymbol{q}^T & \boldsymbol{w}^T & \boldsymbol{\lambda}^T\end{bmatrix}$ to this condition yields

$$\begin{bmatrix}\boldsymbol{J}^T\boldsymbol{\lambda}\\ \boldsymbol{w}+\boldsymbol{\lambda}\\ \boldsymbol{w}-\boldsymbol{e}\end{bmatrix}+\begin{bmatrix}\sum_{d=1}^m\lambda_d\boldsymbol{H}_d & \boldsymbol{0} & \boldsymbol{J}^T\\ \boldsymbol{0} & \boldsymbol{I} & \boldsymbol{I}\\ \boldsymbol{J} & \boldsymbol{I} & \boldsymbol{0}\end{bmatrix}\begin{bmatrix}\boldsymbol{\Delta q}^{(k+1)}\\ \boldsymbol{\Delta w}^{(k+1)}\\ \boldsymbol{\Delta\lambda}^{(k+1)}\end{bmatrix}=\boldsymbol{0}\,. \tag{13}$$

We have $\nabla^2_{\boldsymbol{q}}\mathcal{L}=\nabla_{\boldsymbol{q}}\boldsymbol{J}^T\boldsymbol{\lambda}=\sum_{d=1}^m\lambda_d\boldsymbol{H}_d$ which we will refer to as the Hessian of the Lagrangian function throughout this paper. Using the variable change $\boldsymbol{w}^{(k+1)}=\boldsymbol{w}+\boldsymbol{\Delta w}^{(k+1)}$ and $\boldsymbol{\lambda}^{(k+1)}=\boldsymbol{\lambda}+\boldsymbol{\Delta\lambda}^{(k+1)}$, we get the system

$$\begin{bmatrix}\sum_{d=1}^m\lambda_d\boldsymbol{H}_d & \boldsymbol{0} & \boldsymbol{J}^T\\ \boldsymbol{0} & \boldsymbol{I} & \boldsymbol{I}\\ \boldsymbol{J} & \boldsymbol{I} & \boldsymbol{0}\end{bmatrix}\begin{bmatrix}\boldsymbol{\Delta q}^{(k+1)}\\ \boldsymbol{w}^{(k+1)}\\ \boldsymbol{\lambda}^{(k+1)}\end{bmatrix}=\begin{bmatrix}\boldsymbol{0}\\ \boldsymbol{0}\\ \boldsymbol{e}\end{bmatrix}\,. \tag{14}$$

Solving above system yields the step $\boldsymbol{\Delta q}^{(k+1)}$ (in the following we simply write $\boldsymbol{\Delta q}$) which can be integrated to the new configuration $\boldsymbol{q}^{(k+1)}=\boldsymbol{q}+\boldsymbol{\Delta q}$.

This is also the optimality condition $\nabla_{\boldsymbol{x}}\mathcal{L}=\boldsymbol{0}$ of the quadratic problem

$$\min_{\boldsymbol{\Delta q},\boldsymbol{w}^{(k+1)}} \quad \frac{1}{2}\left\|\boldsymbol{w}^{(k+1)}\right\|^2+\frac{1}{2}\boldsymbol{\Delta q}^T\sum_{d=1}^m\lambda_d\boldsymbol{H}_d\boldsymbol{\Delta q} \tag{15}$$

$$\text{s.t.} \quad \boldsymbol{e}-\boldsymbol{J}\boldsymbol{\Delta q}=\boldsymbol{w}^{(k+1)} \tag{16}$$

with the Lagrangian function $\mathcal{L}=\frac{1}{2}{\boldsymbol{w}^{(k+1)}}^T\boldsymbol{w}^{(k+1)}+\frac{1}{2}\boldsymbol{\Delta q}^T\sum_{d=1}^m\lambda_d\boldsymbol{H}_d\boldsymbol{\Delta q}+{\boldsymbol{\lambda}^{(k+1)}}^T(\boldsymbol{w}^{(k+1)}-\boldsymbol{e}+\boldsymbol{J}\boldsymbol{\Delta q})$. This form is akin to Newton's method of optimization. If $\hat{\boldsymbol{H}}=\sum_{d=1}^m\lambda_d\boldsymbol{H}_d$ is positive definite above problem can be rewritten to least squares form

$$\min_{\boldsymbol{\Delta q}} \quad \frac{1}{2}\left\|\begin{bmatrix}\boldsymbol{J}\\ \boldsymbol{R}\end{bmatrix}\boldsymbol{\Delta q}-\begin{bmatrix}\boldsymbol{e}\\ \boldsymbol{0}\end{bmatrix}\right\|^2_2 \tag{17}$$

with the Cholesky decomposition $\hat{\boldsymbol{H}}=\boldsymbol{R}^T\boldsymbol{R}$ and with $\boldsymbol{w}^{(k+1)}$ being given implicitly.

If the second order information $\hat{\boldsymbol{H}}$ is neglected

$$\min_{\boldsymbol{\Delta q}} \quad \frac{1}{2}\left\|\boldsymbol{J}\boldsymbol{\Delta q}-\boldsymbol{e}\right\|^2_2 \tag{18}$$

we get to the GN algorithm.

In the following we refer to Newton's method as being the 'augmented' (as in augmented with second order information) version of the GN algorithm [25].

The Lagrangian function of the general problem (7) at level l with equalities only is

$$\mathcal{L}_l=\frac{1}{2}\boldsymbol{w}_l^T\boldsymbol{w}_l+\boldsymbol{\lambda}_{l,l}^T(\boldsymbol{w}_l-\boldsymbol{e}_l)+\underline{\boldsymbol{\lambda}}_{l-1,l}^T(\underline{\boldsymbol{w}}^*_{l-1}-\underline{\boldsymbol{e}}_{l-1})\,. \tag{19}$$

Note that $\boldsymbol{\lambda}\in\mathbb{R}^{\sum_{i=1}^p m_i,p}$ is now a matrix with p columns.

The optimality condition is $\nabla_{\boldsymbol{q},\boldsymbol{w},\boldsymbol{\lambda}}\mathcal{L}_l=\boldsymbol{K}_l(\boldsymbol{q},\boldsymbol{w}_l,\boldsymbol{\lambda}_{l,l},\underline{\boldsymbol{\lambda}}_{l-1,l})=\boldsymbol{0}$, with $\boldsymbol{\lambda}_{l,l}$ being the Lagrange multiplier associated to (8), $\underline{\boldsymbol{\lambda}}_{l-1,l}$ being the one associated to all the constraints (9) and with

$$\boldsymbol{K}_l(\boldsymbol{q},\boldsymbol{w}_l,\boldsymbol{\lambda}_{l,l},\underline{\boldsymbol{\lambda}}_{l-1,l})=\begin{bmatrix}\boldsymbol{J}_l^T\boldsymbol{\lambda}_{l,l}+\underline{\boldsymbol{J}}_{l-1}^T\underline{\boldsymbol{\lambda}}_{l-1,l}\\ \boldsymbol{w}_l+\boldsymbol{\lambda}_{l,l}\\ \boldsymbol{w}_l-\boldsymbol{e}_l\\ \underline{\boldsymbol{w}}^*_{l-1}-\underline{\boldsymbol{e}}_{l-1}\end{bmatrix}\,. \tag{20}$$

We then again apply a Newton step $\boldsymbol{K}_l(\boldsymbol{x}+\boldsymbol{\Delta x})=\boldsymbol{K}_l(\boldsymbol{x})+\nabla\boldsymbol{K}_l(\boldsymbol{x})\boldsymbol{\Delta x}=\boldsymbol{0}$ with $\boldsymbol{x}^T=\begin{bmatrix}\boldsymbol{q}^T & \boldsymbol{w}_l^T & \boldsymbol{\lambda}_{l,l}^T & \underline{\boldsymbol{\lambda}}_{l-1,l}^T\end{bmatrix}$. Now, the Hessian writes

$$\hat{\boldsymbol{H}}_l=\sum_{d=1}^{m_l}\lambda_{l,l,d}\boldsymbol{H}_{l,d}+\sum_{i=1}^{l-1}\sum_{d=1}^{m_i}\lambda_{i,l,d}\boldsymbol{H}_{i,d}\,. \tag{21}$$

If it is positive definite, the least squares formulation becomes

$$\min_{\boldsymbol{\Delta q}} \quad \frac{1}{2}\left\|\begin{bmatrix}\boldsymbol{J}_l\\ \boldsymbol{R}_l\end{bmatrix}\boldsymbol{\Delta q}-\begin{bmatrix}\boldsymbol{e}_l\\ \boldsymbol{0}\end{bmatrix}\right\|^2_2 \qquad l=1..p \tag{22}$$

$$\text{s.t.} \quad \underline{\boldsymbol{e}}_{l-1}-\underline{\boldsymbol{J}}_{l-1}\boldsymbol{\Delta q}=\underline{\boldsymbol{w}}^{*,(k+1)}_{l-1}\,, \tag{23}$$

for Newton's method or

$$\min_{\boldsymbol{\Delta q}} \quad \frac{1}{2}\left\|\boldsymbol{J}_l\boldsymbol{\Delta q}-\boldsymbol{e}_l\right\|^2_2 \qquad l=1..p \tag{24}$$

$$\text{s.t.} \quad \underline{\boldsymbol{e}}_{l-1}-\underline{\boldsymbol{J}}_{l-1}\boldsymbol{\Delta q}=\underline{\boldsymbol{w}}^{*,(k+1)}_{l-1} \tag{25}$$

for the GN-algorithm if the second order information $\boldsymbol{R}$ is neglected.

Such hierarchical least squares problems can be solved in a very efficient manner, see e.g. [1] or [2].

Inequality constraints (tasks) can be incorporated due to the problem formulation with slack variables, see [12]. Above first order optimality conditions extend to the Karush-Kuhn-Tucker conditions. Both solvers mentioned above implement the active set method. Inactive constraints i thereby result in $\boldsymbol{w}_i=\boldsymbol{0}$, $\boldsymbol{\lambda}_{j,i}=\boldsymbol{0}$, not further influencing the Hessian $\hat{\boldsymbol{H}}_j$ on some level $j\geq i$.

IV. From optimization to kinematic control

In the previous section III we introduced the GN algorithm and Newton's method of constrained optimization to drive a non-linear geometric error function to zero. As we saw in our previous work [27], this is equivalent to defining a quadratic Taylor approximation [26]

$$\Phi(\boldsymbol{q}+\boldsymbol{\Delta q}) \approx$$

$$\Phi(\boldsymbol{q}) - \boldsymbol{\Delta q}^T \boldsymbol{J}^T \boldsymbol{e} + \frac{1}{2}\boldsymbol{\Delta q}^T (\boldsymbol{J}^T\boldsymbol{J} + \boldsymbol{H})\boldsymbol{\Delta q} \quad (26)$$

$$= \frac{1}{2}\left\| \begin{bmatrix} \boldsymbol{J} \\ \boldsymbol{R} \end{bmatrix} \boldsymbol{\Delta q} - \begin{bmatrix} \boldsymbol{e} \\ \boldsymbol{0} \end{bmatrix} \right\|_2^2 \quad (27)$$

of the quadratic function

$$\Phi(\boldsymbol{q}) = \frac{1}{2}\boldsymbol{e}^T\boldsymbol{e} \quad (28)$$

around $\boldsymbol{q}$ and looking for a bounded step $\boldsymbol{\Delta q}$ in a certain neighbourhood (called trust region) of it. Assuming a control time step of $\Delta t = 1$ s with $\boldsymbol{\Delta q} = \Delta t \dot{\boldsymbol{q}} = \dot{\boldsymbol{q}}$ we can make a trivial connection between optimization, aiming to make a step $\boldsymbol{\Delta q}$ towards the optimum, and control, aiming to determine the new robot state triple $\left[\boldsymbol{q}, \dot{\boldsymbol{q}}, \ddot{\boldsymbol{q}}\right]$ while minimizing the error function with a certain behaviour.

However, a robot is usually controlled at a much lower period with $\Delta t \ll 1$ s. Especially when considering the robot dynamics, one needs to compute the new velocity $\dot{\boldsymbol{q}}^{(k+1)}$ at each control step k without making any assumption about the time step Δt. One can look for the new velocity $\dot{\boldsymbol{q}}^{(k+1)}$ with the relation

$$\dot{\boldsymbol{e}}_{\text{ctrl}} = -\boldsymbol{J}\dot{\boldsymbol{q}}^{(k+1)}\,, \quad (29)$$

yielding a given error velocity $\dot{\boldsymbol{e}}^{\text{ctrl}}$. We can define a simple proportional controller like $\dot{\boldsymbol{e}}^{\text{ctrl}} = -k_p\boldsymbol{e}$ with gain k_p.

The required error decrease may only be achievable in a linear least-squares sense

$$\min_{\dot{\boldsymbol{q}}^{(k+1)}} \quad \frac{1}{2}\|\boldsymbol{J}\dot{\boldsymbol{q}}^{(k+1)} + \dot{\boldsymbol{e}}^{\text{ctrl}}\|_2^2\,. \quad (30)$$

A solution is given by

$$\dot{\boldsymbol{q}}^{(k+1)} = -\boldsymbol{J}^+\dot{\boldsymbol{e}}^{\text{ctrl}}\,, \quad (31)$$

using the Moore-Penrose pseudo-inverse $\boldsymbol{J}^+$. This formulation corresponds to the GN algorithm.

Then, we integrate the velocities to the next state $\boldsymbol{q}^{(k+1)}$

$$\boldsymbol{q}^{(k+1)} = \boldsymbol{q} + \Delta t\dot{\boldsymbol{q}}^{(k+1)} = \boldsymbol{q} + \boldsymbol{\Delta q}\,. \quad (32)$$

However, this approach is not suitable in the vicinity of singularities: $\boldsymbol{J}$ is almost loosing at least one rank and $\dot{\boldsymbol{q}}^{(k+1)}$ becomes very large due to numerical limitations.

In such situations we proposed in [27] to use Newton's method instead. Our claim was that neglecting the second order information $\boldsymbol{H}$ in the second order Taylor approximation (26) (as done in the GN algorithm), is only sufficient if away from singularities. Using the second order information close to singularities restores the original accuracy of the second order Taylor approximation and prevents singular behaviour.

By taking $\boldsymbol{H}$ as a multiple of the identity, there is a connection with the LM algorithm, which can be considered as the classical method for singularity resolution in kinematics control [18].

However, the LM algorithm is a 'bad' approximation of the true second order information with worse convergence behaviour. Later (see sec. VI) we present how to use the true second order information, or how to gain a positive definite approximation of it by the BFGS algorithm.

In the hierarchy, each non-linear task $\boldsymbol{f}(\boldsymbol{q}) = \boldsymbol{f}_d$ or $\boldsymbol{f}(\boldsymbol{q}) \leq \boldsymbol{f}_d$ is rewritten to the constrained GN algorithm of control

$$\min_{\dot{\boldsymbol{q}}^{(k+1)}} \quad \frac{1}{2}\|\boldsymbol{J}_l\dot{\boldsymbol{q}}^{(k+1)} + \dot{\boldsymbol{e}}_l^{\text{ctrl}}\|_2^2 \quad (33)$$

$$\text{s.t.} \quad -\underline{\dot{\boldsymbol{e}}}_{l-1}^{\text{ctrl}} - \underline{\boldsymbol{J}}_{l-1}\dot{\boldsymbol{q}}^{(k+1)} \leqq \underline{\boldsymbol{w}}_{l-1}^{*,(k+1)} \quad (34)$$

or the constrained Newton's method of control

$$\min_{\dot{\boldsymbol{q}}^{(k+1)}} \quad \frac{1}{2}\left\| \begin{bmatrix} \boldsymbol{J}_l \\ \boldsymbol{R}_l \end{bmatrix} \dot{\boldsymbol{q}}^{(k+1)} + \begin{bmatrix} \dot{\boldsymbol{e}}_l^{\text{ctrl}} \\ \boldsymbol{0} \end{bmatrix} \right\|_2^2 \quad (35)$$

$$\text{s.t.} \quad -\underline{\dot{\boldsymbol{e}}}_{l-1}^{\text{ctrl}} - \underline{\boldsymbol{J}}_{l-1}\dot{\boldsymbol{q}} \leqq \underline{\boldsymbol{w}}_{l-1}^{*,(k+1)}\,. \quad (36)$$

The set of priority-ordered least-squares problems is then passed to the hierarchical solver [2].

Our switching strategy between the GN algorithm and Newton's method is described in sec. VII.

Note that Δt connects the two entities of 'optimization' (optim) and 'control'

$$\boldsymbol{J}\boldsymbol{\Delta q} + \Delta t\dot{\boldsymbol{e}}^{\text{ctrl}} = \Delta t(\boldsymbol{J}\dot{\boldsymbol{q}}^{(k+1)} + \dot{\boldsymbol{e}}^{\text{ctrl}}) =$$

$$\boldsymbol{w}^{\text{optim}} = \Delta t\boldsymbol{w}^{\text{ctrl}}\,. \quad (37)$$

That is, we calculate a new velocity $\dot{\boldsymbol{q}}^{(k+1)}$ but only do a model based step $\boldsymbol{\Delta q} = \Delta t\dot{\boldsymbol{q}}^{(k+1)}$ towards the optimum. Consequently, the model needs to be updated with the Hessian of the Lagrangian (19) using $\boldsymbol{w}^{\text{optim}}$ and $\boldsymbol{\lambda}^{\text{optim}}$. Since solving the constrained control problems (33) or (35) yields $\boldsymbol{w}^{\text{ctrl}}$ and $\boldsymbol{\lambda}^{\text{ctrl}}$, a scaling of the form $\boldsymbol{w}^{\text{optim}} = \Delta t\boldsymbol{w}^{\text{ctrl}}$ according to (37) is required. Due to the linear dependency between the slack $\boldsymbol{w}$ and the Lagrange multipliers $\boldsymbol{\lambda}$, see [2], we further get $\boldsymbol{\lambda}^{\text{optim}} = \Delta t\boldsymbol{\lambda}^{\text{ctrl}}$. In what follows, we write $\boldsymbol{w} = \boldsymbol{w}^{\text{optim}}$ and $\boldsymbol{\lambda} = \boldsymbol{\lambda}^{\text{optim}}$.

V. Dynamically feasible kinematic control

The equation of motion ensuring physical feasibility is of second order and correspondingly second order motion controllers are defined. However, in the previous section IV we have formulated the GN algorithm and Newton's method only in the velocity domain. In sec. V-A we argue why the GN algorithm and Newton's method cannot be extended easily to the acceleration domain. Therefore, our idea for acceleration-based control is to change the right hand side $\dot{\boldsymbol{e}}^{\text{ctrl}}$ from a linear proportional controller to some controller $\dot{\boldsymbol{e}}_{\text{PD}}^{\text{ctrl}}$ that emulates second order PD control $\ddot{\boldsymbol{e}}^{\text{ctrl}}$ in the velocity domain. In sec. V-B we show how this can be achieved and apply the corresponding necessary adaptations to the equation of motion which we include into our control framework (see sec. V-C).

A. Damping in acceleration-based control

In the following we show that

$$\min_{\ddot{\boldsymbol{q}}^{(k+1)}} \quad \frac{1}{2}\left\| \begin{bmatrix} \boldsymbol{J} \\ \boldsymbol{R} \end{bmatrix} \ddot{\boldsymbol{q}}^{(k+1)} + \begin{bmatrix} \dot{\boldsymbol{J}}\dot{\boldsymbol{q}}^{(k)} + \ddot{\boldsymbol{e}}^{\text{ctrl}} \\ \boldsymbol{0} \end{bmatrix} \right\|_2^2 \tag{38}$$

leads to low frequency oscillations around a minimum. For simplicity, we assume $\boldsymbol{R} = \mu \boldsymbol{I}$.

Let's take a point mass ($m = 1$) robot in 1D (x-axis) with position $f = x$. Its desired position is $x_d = 0$. The task error is then $e = x_d - x = -x$. The Jacobian of this robot is $J = \frac{df}{dx} = \frac{dx}{dx} = 1$, the time derivative of the Jacobian is $\dot{J} = 0$.

1) Velocity-based control: The control law is usually written as first order ordinary differential equation (ODE)

$$\dot{e}^{\text{ctrl}} = -k_p e = k_p x = -J\dot{x} \tag{39}$$

$$\dot{x} + k_p x = 0 \tag{40}$$

with the solution

$$\int \frac{dx}{x} = -k_p \int dt \tag{41}$$

$$\ln x = -k_p t + C \tag{42}$$

$$x = D \exp(-k_p t)\,, \tag{43}$$

where $x \to 0$ for $t \to \infty$. C and D are constants of integration.

The numerical integration can be formulated as

$$x^{(k+1)} = x - \Delta t k_p x\,. \tag{44}$$

2) Velocity-based control with damping: If we introduce damping we get the following form

$$\begin{bmatrix} 1 \\ \mu \end{bmatrix} \dot{x} + \begin{bmatrix} k_p \\ 0 \end{bmatrix} x = \begin{bmatrix} 0 \\ 0 \end{bmatrix}\,. \tag{45}$$

The least squares solution to this problem (applying the pseudo-inverse $\boldsymbol{A}^+ = (\boldsymbol{A}^T\boldsymbol{A})^{-1}\boldsymbol{A}^T$ for a full rank matrix $\boldsymbol{A}$) is the first order ODE

$$\dot{x} + \frac{k_p}{1+\mu^2} x = 0 \tag{46}$$

and has the same solution as above (see sec. V-A.1)

3) Acceleration-based control: The control law for *acceleration-based control* is usually written as

$$\ddot{e}^{\text{ctrl}} = -k_p e - k_v \dot{e} = k_p x + k_v \dot{x} = -J\ddot{x} - \dot{J}\dot{x} \tag{47}$$

$$\ddot{x} + k_v \dot{x} + k_p x = 0\,. \tag{48}$$

The solution for this ODE is

$$x = \exp(-\frac{1}{2}\delta t)(A\cos(w_d t) + B\sin(w_d t)) \tag{49}$$

where $\delta = k_v/2/m$ and $w_d = \sqrt{k_p/m - \delta^2}$. Critical damping with exponential convergence can be achieved for $k_v(k_p) = 2\sqrt{m k_p}$ such that $w_d = 0$.

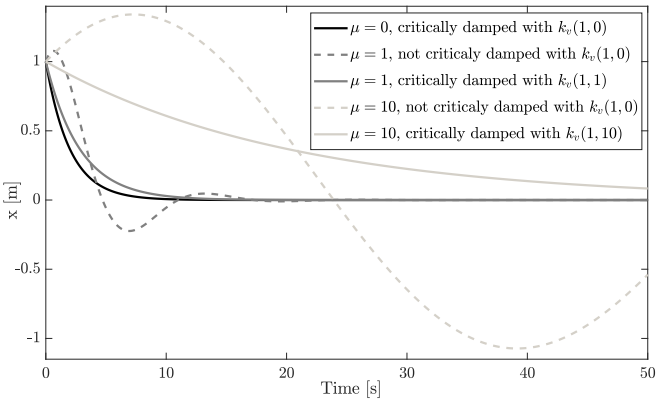


Fig. 1: (49) plotted for $m = 1$, $k_p = 1$, different μ and $k_v = f(k_p) = 2\sqrt{m k_p}$ (black line and dashed lines) or $k_v = f^*(k_p, \mu) = 2\sqrt{m(1+\mu^2)k_p}$.

4) Acceleration-based control with damping: For the augmented system

$$\begin{bmatrix} 1 \\ \mu \end{bmatrix} \ddot{x} + \begin{bmatrix} k_v \\ 0 \end{bmatrix} \dot{x} + \begin{bmatrix} k_p \\ 0 \end{bmatrix} x = \begin{bmatrix} 0 \\ 0 \end{bmatrix} \tag{50}$$

again we apply the pseudo-inverse and get following ODE:

$$\ddot{x} + \frac{k_v}{1+\mu^2}\dot{x} + \frac{k_p}{1+\mu^2} x = 0 \tag{51}$$

The numerical integration writes as

$$x^{(k+1)} = x + \Delta t \dot{x} + \frac{\Delta t^2}{2}\left(-\frac{k_v}{1+\mu^2}\dot{x} - \frac{k_p}{1+\mu^2} x\right), \tag{52}$$

clearly exposing the influence of the damping μ on the critically damped task gains k_p and k_v.

Critical damping can now be achieved with $k_v(k_p, \mu) = 2\sqrt{m(1+\mu^2)k_p}$. Some convergence curves are plotted in fig. 1. It can be clearly seen that the damping μ influences the critically damped system negatively (i.e. overshooting) if the gain is chosen according to $k_v(k_p)$ instead of $k_v(k_p, \mu)$.

For a more complicated 3D robot with more and especially coupled degree of freedom (DoF), and a time varying $\boldsymbol{R}$, it seems cumbersome to find the expression for critical damping $k_v(k_p, \boldsymbol{R})$ such that overshooting behaviour can be prevented.

Therefore, we favour to shift the whole problem into the velocity domain and emulate acceleration-based control by adapting the right hand-side accordingly. This way we can easily achieve exponential convergence as shown below.

B. Acceleration-based control expressed in the velocity domain

In velocity based control, the new joint velocity obtained from (31) is integrated to the new joint configuration $\boldsymbol{q}^{(k+1)}$ by

$$\boldsymbol{q}^{(k+1)} = \boldsymbol{q} + \Delta t \dot{\boldsymbol{q}}^{(k+1)} = \boldsymbol{q} - \Delta t \boldsymbol{J}^{\ddagger}\dot{\boldsymbol{e}}^{\text{ctrl}} \tag{53}$$

where $\boldsymbol{J}^{\ddagger}$ denotes the hierarchical pseudo-inverse [1].

The same scheme as in velocity-based control can be used for acceleration-based control, leading to

$$\ddot{\boldsymbol{e}}^{\text{ctrl}} = -\boldsymbol{J}\ddot{\boldsymbol{q}}^{(k+1)} - \dot{\boldsymbol{J}}\dot{\boldsymbol{q}}\,. \tag{54}$$

$\ddot{\boldsymbol{e}}^{\text{ctrl}}$ is defined as a PD controller

$$\ddot{\boldsymbol{e}}^{\text{ctrl}} = -k_p \boldsymbol{e} - k_v \dot{\boldsymbol{e}} \,. \tag{55}$$

The integration to the new joint configuration then takes the form

$$\dot{\boldsymbol{q}}^{(k+1)} = \dot{\boldsymbol{q}} + \Delta t \ddot{\boldsymbol{q}}^{(k+1)} = \dot{\boldsymbol{q}} - \Delta t \boldsymbol{J}^{\ddagger}(\ddot{\boldsymbol{e}}^{\text{ctrl}} + \dot{\boldsymbol{J}}\dot{\boldsymbol{q}}) \,, \tag{56}$$

$$\boldsymbol{q}^{(k+1)} = \boldsymbol{q} + \Delta t \dot{\boldsymbol{q}} + \frac{\Delta t^2}{2} \ddot{\boldsymbol{q}}^{(k+1)} \tag{57}$$

$$= \boldsymbol{q} + \Delta t \dot{\boldsymbol{q}} - \frac{\Delta t^2}{2} \boldsymbol{J}^{\ddagger}(\ddot{\boldsymbol{e}}^{\text{ctrl}} + \dot{\boldsymbol{J}}\dot{\boldsymbol{q}}) \,. \tag{58}$$

Let's now replace the accelerations in (54) by forward differences

$$\ddot{\boldsymbol{q}}^{(k+1)} = \frac{\dot{\boldsymbol{q}}^{(k+1)} - \dot{\boldsymbol{q}}}{\Delta t} \tag{59}$$

such that we get

$$\dot{\boldsymbol{e}}^{\text{ctrl}}_{\text{PD}} = -\boldsymbol{J}\dot{\boldsymbol{q}}^{(k+1)} \,. \tag{60}$$

$\dot{\boldsymbol{e}}^{\text{ctrl}}_{\text{PD}}$ is defined as

$$\dot{\boldsymbol{e}}^{\text{ctrl}}_{\text{PD}} = -\boldsymbol{J}\dot{\boldsymbol{q}} + \Delta t(\ddot{\boldsymbol{e}}^{\text{ctrl}} + \dot{\boldsymbol{J}}\dot{\boldsymbol{q}}) \tag{61}$$

for the GN algorithm and

$$\dot{\boldsymbol{e}}^{\text{ctrl}}_{\text{PD}} = -\begin{bmatrix}\boldsymbol{J}\\ \boldsymbol{R}\end{bmatrix}\dot{\boldsymbol{q}} + \Delta t(\ddot{\boldsymbol{e}}^{\text{ctrl}} + \dot{\boldsymbol{J}}\dot{\boldsymbol{q}}) \tag{62}$$

for Newton's method. This acceleration-based PD control in velocity gives

$$\dot{\boldsymbol{q}}^{(k+1)} = -\boldsymbol{J}^{\ddagger}\dot{\boldsymbol{e}}^{\text{ctrl}}_{\text{PD}} \tag{63}$$

$$= \boldsymbol{J}^{\ddagger}\boldsymbol{J}\dot{\boldsymbol{q}} - \Delta t \boldsymbol{J}^{\ddagger}(\ddot{\boldsymbol{e}}^{\text{ctrl}} + \dot{\boldsymbol{J}}\dot{\boldsymbol{q}}) \tag{64}$$

$$\boldsymbol{q}^{(k+1)} = \boldsymbol{q} - \Delta t \boldsymbol{J}^{\ddagger}\dot{\boldsymbol{e}}^{\text{ctrl}}_{\text{PD}} \tag{65}$$

$$= \boldsymbol{q} + \Delta t \boldsymbol{J}^{\ddagger}\boldsymbol{J}\dot{\boldsymbol{q}} - \Delta t^2 \boldsymbol{J}^{\ddagger}(\ddot{\boldsymbol{e}}^{\text{ctrl}} + \dot{\boldsymbol{J}}\dot{\boldsymbol{q}}) \tag{66}$$

$$= \boldsymbol{q} + \Delta t \dot{\boldsymbol{q}} - \Delta t^2 \boldsymbol{J}^{\ddagger}(\ddot{\boldsymbol{e}}^{\text{ctrl}} + \dot{\boldsymbol{J}}\dot{\boldsymbol{q}}) \tag{67}$$

for the GN algorithm and

$$\dot{\boldsymbol{q}}^{(k+1)} = -\begin{bmatrix}\boldsymbol{J}\\ \boldsymbol{R}\end{bmatrix}^{\ddagger}\dot{\boldsymbol{e}}^{\text{ctrl}}_{\text{PD}} \tag{68}$$

$$= \begin{bmatrix}\boldsymbol{J}\\ \boldsymbol{R}\end{bmatrix}^{\ddagger}\begin{bmatrix}\boldsymbol{J}\\ \boldsymbol{R}\end{bmatrix}\dot{\boldsymbol{q}} - \Delta t \begin{bmatrix}\boldsymbol{J}\\ \boldsymbol{R}\end{bmatrix}^{\ddagger}(\ddot{\boldsymbol{e}}^{\text{ctrl}} + \dot{\boldsymbol{J}}\dot{\boldsymbol{q}}) \tag{69}$$

$$\boldsymbol{q}^{(k+1)} = \boldsymbol{q} - \Delta t \begin{bmatrix}\boldsymbol{J}\\ \boldsymbol{R}\end{bmatrix}^{\ddagger}\dot{\boldsymbol{e}}^{\text{ctrl}}_{\text{PD}} \tag{70}$$

$$= \boldsymbol{q} + \Delta t \begin{bmatrix}\boldsymbol{J}\\ \boldsymbol{R}\end{bmatrix}^{\ddagger}\begin{bmatrix}\boldsymbol{J}\\ \boldsymbol{R}\end{bmatrix}\dot{\boldsymbol{q}} - \Delta t^2 \begin{bmatrix}\boldsymbol{J}\\ \boldsymbol{R}\end{bmatrix}^{\ddagger}(\ddot{\boldsymbol{e}}^{\text{ctrl}} + \dot{\boldsymbol{J}}\dot{\boldsymbol{q}}) \tag{71}$$

$$= \boldsymbol{q} + \Delta t \dot{\boldsymbol{q}} - \Delta t^2 \begin{bmatrix}\boldsymbol{J}\\ \boldsymbol{R}\end{bmatrix}^{\ddagger}(\ddot{\boldsymbol{e}}^{\text{ctrl}} + \dot{\boldsymbol{J}}\dot{\boldsymbol{q}}) \tag{72}$$

for Newton's method if $\dot{\boldsymbol{e}}^{\text{ctrl}}_{\text{PD}}$ is used instead of $\dot{\boldsymbol{e}}^{\text{ctrl}}$ in (53). If $\text{rank}(\boldsymbol{J}) = m$ or $\text{rank}(\begin{bmatrix}\boldsymbol{J} & \boldsymbol{R}\end{bmatrix}^T) = n$ then $\boldsymbol{J}^{\ddagger}\boldsymbol{J} = \boldsymbol{I}$ or $\begin{bmatrix}\boldsymbol{J} & \boldsymbol{R}\end{bmatrix}^{T\ddagger}\begin{bmatrix}\boldsymbol{J} & \boldsymbol{R}\end{bmatrix}^T = \boldsymbol{I}$ [1] and the above value of $\dot{\boldsymbol{q}}^{(k+1)}$ in (67) is the same as the one obtained with acceleration-based control in (58). However, the joint positions are missing the factor 0.5 in front of the third term of (67) and (72).

Therefore, an adjustment in the calculation of the joint positions need to be made. We calculate the joint positions by

$$\boldsymbol{q}^{(k+1)}_{\text{mod}} = \boldsymbol{q} + \Delta t(\frac{1}{2}\dot{\boldsymbol{q}}^{(k+1)} + \frac{1}{2}\dot{\boldsymbol{q}}) \tag{73}$$

$$= \boldsymbol{q} + \Delta t(\frac{1}{2}(\dot{\boldsymbol{q}} - \Delta t \boldsymbol{J}^{\ddagger}(\ddot{\boldsymbol{e}}^{\text{ctrl}} + \dot{\boldsymbol{J}}\dot{\boldsymbol{q}})) + \frac{1}{2}\dot{\boldsymbol{q}}) \tag{74}$$

$$= \boldsymbol{q} + \Delta t \dot{\boldsymbol{q}} - \frac{\Delta t^2}{2}\boldsymbol{J}^{\ddagger}(\ddot{\boldsymbol{e}}^{\text{ctrl}} + \dot{\boldsymbol{J}}\dot{\boldsymbol{q}}) \,. \tag{75}$$

which corresponds to the one of the acceleration based equation of motion. The joint velocities $\dot{\boldsymbol{q}}^{(k+1)}$ stay untouched since they already correspond to the ones of the acceleration based PD controller. Accordingly, we need to calculate the step $\boldsymbol{\Delta q}$ for LexLSAug2BFGS by

$$\boldsymbol{\Delta q}_{\text{mod}} = \boldsymbol{q}^{(k+1)}_{\text{mod}} - \boldsymbol{q}_{\text{mod}} \tag{76}$$

instead of $\boldsymbol{\Delta q} = \Delta t \dot{\boldsymbol{q}}^{(k+1)}$. Both LexLSAug2AH and LexLSAug2BFGS are robust with regard to the inconsistent Lagrange multipliers which correspond to the step $\boldsymbol{\Delta q}$ and not to the true step $\boldsymbol{\Delta q}_{\text{mod}}$. However, it is subject to discussion whether consistency with the acceleration based problem or consistency with the optimization problem is preferred. For the real robot experiments in the validation section IX, we use (59) without the corresponding modification of the joint positions (73). This poses a good compromise between consistency with the acceleration based problem and the optimization theory.

In [38] it is proposed to simply control a robot in velocity-based control

$$\dot{\boldsymbol{e}}^{\text{ctrl}} = -\boldsymbol{J}\dot{\boldsymbol{q}}^{(k+1)} \tag{77}$$

with $\dot{\boldsymbol{e}}^{\text{ctrl}} = -k_p \boldsymbol{e}$ since both velocity-based and acceleration-based control are consistent. However, desired behaviours like PD control can not be realized. That is why we choose to implement the above presented method for acceleration-based control in velocity, solving (33) or (35) with our new right hand side $\dot{\boldsymbol{e}}^{\text{ctrl}}_{\text{PD}}$ (61).

To come back to the 1-D robot example from the previous section V-A, we can now see that damping terms do not influence the critically damped system k_p, k_v. The new velocity for a control step k can be calculated by

$$\dot{x}^{(k+1)} = -\dot{e}^{\text{ctrl}}_{\text{PD}} \tag{78}$$

$$= J^{-1}J\dot{x} - \Delta t J^{-1}(\ddot{e}^{\text{ctrl}} + \dot{J}\dot{x}) = \dot{x} - \Delta t \ddot{e}^{\text{ctrl}} \tag{79}$$

with $J = 1$ and $\dot{J} = 0$. The numerical integration can be formulated as

$$x^{(k+1)} = x + \Delta t \dot{x} - \Delta t^2 \ddot{e}^{\text{ctrl}} \,. \tag{80}$$

Note that for exponential convergence the original system $\ddot{x} + k_v \dot{x} + k_p x = 0$ still needs to be critically damped with $k_v = 2\sqrt{m k_p}$.

For the augmented system

$$\begin{bmatrix}1\\ \mu\end{bmatrix}\dot{x} + \begin{bmatrix}\dot{e}^{\text{ctrl}}_{\text{PD}}\\ 0\end{bmatrix} = \begin{bmatrix}0\\ 0\end{bmatrix} \tag{81}$$

again we apply the pseudo-inverse and get the following expression for the new velocity $\dot{x}^{(k+1)}$:

$$\dot{x}^{(k+1)} = -\frac{1}{1+\mu^2}\dot{e}_{\text{PD}}^{\text{ctrl}} = \frac{1}{1+\mu^2}(\dot{x} - \Delta t\ddot{e}^{\text{ctrl}})\,. \tag{82}$$

The numerical integration can be formulated as

$$x^{(k+1)} = x + \frac{\Delta t}{1+\mu^2}(\dot{x} - \Delta t\ddot{e}^{\text{ctrl}})\,. \tag{83}$$

The original critically damped system k_p, k_v is not influenced by the damping μ, ensuring exponential convergence.

Note that this scheme of linear integration of the joint velocities to the joint angle configuration requires the orientation of the robot-base to be expressed with Euler angles instead of quaternions. Singular cases (gimbal lock) need to be avoided. We assume small changes of the base orientation between control iterations. We then can set the robot's free-flyer base configuration (for example the Denavit-Hartenberg parameters) to the current joint angle configuration. At the same time the Euler angles are set to the zero $x-y-x$ Euler configuration.

C. Including the dynamics

We have formulated our second order motion controllers in the velocity domain. Similarly, the acceleration components of the equation of motion (1) are replaced by forward differences:

$$\boldsymbol{M}(\boldsymbol{q})\frac{\dot{\boldsymbol{q}}^{(k+1)} - \dot{\boldsymbol{q}}}{\Delta t} + \boldsymbol{N}(\boldsymbol{q},\dot{\boldsymbol{q}}) = \boldsymbol{S}^T\boldsymbol{\tau}^{(k+1)} + \boldsymbol{J}_c^T\boldsymbol{\gamma}^{(k+1)}\,, \tag{84}$$

or rewritten into matrix form

$$\begin{bmatrix}\frac{\boldsymbol{M}(\boldsymbol{q})}{\Delta t} & -\boldsymbol{S}^T & -\boldsymbol{J}_c^T\end{bmatrix}\begin{bmatrix}\dot{\boldsymbol{q}}^{(k+1)}\\ \boldsymbol{\tau}^{(k+1)}\\ \boldsymbol{\gamma}^{(k+1)}\end{bmatrix} = \boldsymbol{M}\frac{\dot{\boldsymbol{q}}}{\Delta t} - \boldsymbol{N}(\boldsymbol{q},\dot{\boldsymbol{q}})\,. \tag{85}$$

However, for numerical robustness it is desirable to keep the conditioning of the system matrix by

$$\begin{bmatrix}\boldsymbol{M}(\boldsymbol{q}) & -\boldsymbol{S}^T & -\boldsymbol{J}_c^T\end{bmatrix}\begin{bmatrix}\dot{\boldsymbol{q}}^{(k+1)}\\ \Delta t\boldsymbol{\tau}^{(k+1)}\\ \Delta t\boldsymbol{\gamma}^{(k+1)}\end{bmatrix} = \boldsymbol{M}\dot{\boldsymbol{q}} - \Delta t\boldsymbol{N}(\boldsymbol{q},\dot{\boldsymbol{q}})\,. \tag{86}$$

This computes $\dot{\boldsymbol{q}}^{(k+1)}$, $\Delta t\boldsymbol{\tau}^{(k+1)}$ and $\Delta t\boldsymbol{\gamma}^{(k+1)}$, so the solution vector has to be changed accordingly to get $\boldsymbol{\tau}^{(k+1)}$ and $\boldsymbol{\gamma}^{(k+1)}$ by dividing $\Delta t\boldsymbol{\tau}^{(k+1)}$ and $\Delta t\boldsymbol{\gamma}^{(k+1)}$ by Δt.

The equation of motion can be considered full rank if the inertia matrix $\boldsymbol{M}$ is physically consistent and therefore positive definite [39], [40]. This means that the system matrix of the equation of motion $\begin{bmatrix}\boldsymbol{M} & -\boldsymbol{S}^T & -\boldsymbol{J}_c^T\end{bmatrix}$ is not concerned with kinematic singularities of the contact Jacobians $\boldsymbol{J}_c$.

Furthermore, we do not apply Newton's method. The equation of motion is already linear in the accelerations (or velocities in case of the forward integration), joint torques and generalized contact forces. It therefore fits into our lexicographical problem (6) with the system matrix $\boldsymbol{A} = \begin{bmatrix}\boldsymbol{M} & -\boldsymbol{S}^T & -\boldsymbol{J}_c^T\end{bmatrix}$ and the right hand side $\boldsymbol{b} = -\boldsymbol{M}\dot{\boldsymbol{q}} + \Delta t\boldsymbol{N}(\boldsymbol{q},\dot{\boldsymbol{q}})$ and we do not need to consider it in the Hessian calculation of the lower lever linearized constraints.

Dynamic constraints including the equation of motion and torque and contact force limits have the highest priority. This way their feasibility is guaranteed with $\boldsymbol{w}_{\text{dyn}} = \boldsymbol{0}$.

VI. Hessian calculation for Newton's method of hierarchical control

The calculation of the Hessian $\hat{\boldsymbol{H}}$, which is necessary for the augmentation of the GN algorithm to Newton's method, is presented in this section.

A. BFGS approximation of the Hessian based on the Lagrangian (19)

The Hessian $\hat{\boldsymbol{H}}_l$ of level l (21) can be approximated by the BFGS algorithm

$$\boldsymbol{B}_l^{(k+1)} = \boldsymbol{B}_l + \frac{\boldsymbol{y}_l\boldsymbol{y}_l^T}{\boldsymbol{y}_l^T\boldsymbol{\Delta q}^{(k)}} - \frac{\boldsymbol{B}_l\boldsymbol{\Delta q}^{(k)}\boldsymbol{\Delta q}^{(k),T}\boldsymbol{B}_l}{\boldsymbol{\Delta q}^{(k),T}\boldsymbol{B}_l}\boldsymbol{\Delta q}^{(k)}\,. \tag{87}$$

This update is positive definite if $\boldsymbol{B}_l$ is positive definite as well as the curvature is greater than zero $\boldsymbol{y}_l^T\boldsymbol{\Delta q}^{(k)} > \xi$. In theory $\xi = 0$ but in practice we choose a numerical threshold like $\xi = 1e^{-12}\Delta t^2$. If the curvature condition is not fulfilled no update is done but we keep augmenting with the last updated value $\boldsymbol{B}_l$. Note that in our control we solve for $\dot{\boldsymbol{q}}^{(k+1)}$ so we need to explicitly provide $\boldsymbol{\Delta q}^{(k)} = \Delta t\dot{\boldsymbol{q}}^{(k)}$ to the BFGS algorithm.

Following (20), we define the gradient as

$$\nabla_{\boldsymbol{q}}\mathcal{L}_l = \sum_{i=1}^{l}\boldsymbol{J}_i^T\boldsymbol{\lambda}_{i,l}\,. \tag{88}$$

Then $\boldsymbol{y}_l = \nabla_{\boldsymbol{q}}\mathcal{L}_l - \nabla_{\boldsymbol{q}}\mathcal{L}_l^{(k-1)}$ where we only use the Lagrange multipliers of the current iterate $\boldsymbol{y}_l = -\sum_{i=1}^{l}(\boldsymbol{J}_i - \boldsymbol{J}_i^{(k-1)})^T\boldsymbol{\lambda}_{i,l}$ (see [41]).

$\boldsymbol{B}_l$ is initialized by

$$\boldsymbol{B}_l = \sum_{i\in\mathcal{A}}^{l}\max(\zeta, \frac{1}{2}\|\Delta t\dot{\boldsymbol{e}}_{\text{PD},i}^{\text{ctrl}}\|_2^2)\boldsymbol{I}_i^* \tag{89}$$

where $\mathcal{A}$ is the current active set. $\boldsymbol{I}^*$ is an identity matrix with diagonal entries corresponding to the joints on the kinematic chain of the task. As in [23], $\zeta = 10^{-3}\Delta t^2$ is a lower threshold for augmented tasks with very small errors.

Note that the BFGS update actually computes $\boldsymbol{B} = \boldsymbol{J}^T\boldsymbol{J} + \hat{\boldsymbol{H}}$ and not just $\hat{\boldsymbol{H}}$ alone. However, we skip the reduction to $\hat{\boldsymbol{H}} = \boldsymbol{B} - \boldsymbol{J}^T\boldsymbol{J}$ since it may result in an indefinite matrix $\hat{\boldsymbol{H}}$.

We call this algorithm LexDynBFGS (**Lex**icographic Augmentation for **Dyn**amics with **BFGS**). It is the extension of LexLSAUG2 from our previous work [27].

B. Analytic Hessian

For the analytic Hessian calculation $\hat{\boldsymbol{H}}_l$ of level l (21) we need to calculate the second order derivatives

$$\boldsymbol{H} = \nabla_{\boldsymbol{q}}^2\boldsymbol{f}(\boldsymbol{q}) \tag{90}$$

of the geometric functions $\boldsymbol{f}(\boldsymbol{q})$ for all levels 1 to l. For this we follow [42] which is in $\mathcal{O}(n^2)$ and hence comparable to the BFGS algorithm.

Since the hierarchical Hessian $\hat{\boldsymbol{H}}$ can become negative definite a simple Cholesky decomposition can not be applied. Instead, we use the regularization proposed in [43] beforehand to get the closest semi positive definite matrix $\tilde{\boldsymbol{H}}$ by

$$\tilde{\boldsymbol{H}} = \frac{1}{2}(\hat{\boldsymbol{H}} + \boldsymbol{V}(\boldsymbol{\Sigma} + \epsilon \boldsymbol{I}^*)\boldsymbol{V}^T)\,. \tag{91}$$

The regularization requires a full polar decomposition for example by the SVD decomposition

$$\hat{\boldsymbol{H}} = \boldsymbol{U}\boldsymbol{\Sigma}\boldsymbol{V}^T \tag{92}$$

which is computationally expensive. An approximation of the polar decomposition based on Newton's method is available [44] but does not calculate the squared Eigenvalues $\boldsymbol{\Sigma}$ explicitly.

$\tilde{\boldsymbol{H}}$ is the matrix $\hat{\boldsymbol{H}}$ with all the negative Eigenvalues being zero. We add a small term $\epsilon \boldsymbol{I}^*$ with $\epsilon \ll 1$ in order to improve numerical stability and smoothness of the solution. However, this can influence the convergence behaviour negatively. Indeed, the tendency to only converge to suboptimal minima increases with the magnitude of ϵ.

Numerical behaviour can be improved if Hessians $\boldsymbol{H}$ are only added to $\hat{\boldsymbol{H}}$ if their corresponding Lagrange multiplier $\lambda > \rho$ with $\rho \approx 1e^{-12}\Delta t^2$ being a small numerical value.

We call this algorithm LexDynAH (**Lex**icographic Augmentation for **Dyn**amics with **A**nalytic **H**essian).

C. Hierarchical Hessian built from BFGS approximations of Hessians of non-linear geometric functions

While the BFGS method presented in sec. VI-A approximates the Hessian $\boldsymbol{J}^T\boldsymbol{J} + \hat{\boldsymbol{H}}$ of the Lagrangian function (19) we propose another BFGS method which approximates the Hessian $\boldsymbol{H}$ of the geometric function $\boldsymbol{f}$.

For this we develop the second order Taylor approximation of the non-linear geometric function $\boldsymbol{e}(\boldsymbol{q})$

$$\boldsymbol{f}(\boldsymbol{q} + \boldsymbol{\Delta q}) = \boldsymbol{f}(\boldsymbol{q}) + \boldsymbol{\Delta q}^T \nabla \boldsymbol{f} + \frac{1}{2}\boldsymbol{\Delta q}^T \nabla^2 \boldsymbol{f} \boldsymbol{\Delta q} \tag{93}$$

where $\nabla \boldsymbol{f}_{\boldsymbol{q}} = \boldsymbol{J}$ and $\nabla^2_{\boldsymbol{q}} \boldsymbol{f} = \boldsymbol{H}$.

The gradient $\boldsymbol{y}$ for each component of $\boldsymbol{f}$ is the difference of the transposed corresponding row of $\boldsymbol{J}$ and $\boldsymbol{J}^{(k-1)}$

$$\boldsymbol{y}_{1:m} = (\boldsymbol{J}_{1:m} - \boldsymbol{J}^{(k-1)}_{1:m})^T\,. \tag{94}$$

For an end-effector task in 3D task space $m = 3$. We therefore need to do three BFGS updates (87) for $\boldsymbol{B}_1$ with $\boldsymbol{y}_1$, $\boldsymbol{B}_2$ with $\boldsymbol{y}_2$ and $\boldsymbol{B}_3$ with $\boldsymbol{y}_3$, respectively.

Following (21), these single Hessian components $\boldsymbol{B}$ are then used to compose the Hessian $\hat{\boldsymbol{H}}_l$ of level l.

This means that even though the BFGS updates $\boldsymbol{B}$ are semi-positive definite we again can end up with an indefinite matrix $\hat{\boldsymbol{H}}_l^{\approx}$ which needs to be regularized by the Higham regularization.

This could be one of the reasons why some preliminary results for this BFGS method showed bad behaviour (i.e. unsmooth joint trajectories) so we do not further address it.

The Symmetric Rank 1 (SR1) method [45], [46], [47] might be a promising alternative since it allows indefinite updates and therefore gives a good approximation to the indefinite analytic Hessian.

VII. Switching strategy between GN algorithm and Newton's method and changes in the active set

As in our previous work [27] we switch to the GN-algorithm whenever the quadratic norm residual

$$\omega = \frac{1}{2}\|\boldsymbol{w}\|_2^2 = \frac{1}{2}\|\boldsymbol{J}\dot{\boldsymbol{q}} + \dot{\boldsymbol{e}}^{\text{ctrl}}_{\text{PD}}\|_2^2 < \nu\,. \tag{95}$$

The threshold ν is typically chosen as $\nu = 10^{-12}\Delta t^2$.

Reason is that the second order augmentation $\boldsymbol{R}$ is of full rank and therefore all the kinematic chain's joints of the task on the respective level are occupied. Hence, these joints cannot be used any more for the resolution of tasks on lower priority levels. Therefore, it is necessary to 'free' joints as soon as an augmentation is not necessary to achieve the best possible robot convergence on all levels.

The analytic hierarchical Hessian $\hat{\boldsymbol{H}}$ (21) becomes nil inherently if the corresponding Lagrange multipliers or second order information (90) become nil as well. Since for numerical reasons $\hat{\boldsymbol{H}} \neq \boldsymbol{0}$ even if we are close to the two conditions we use the above switching method even when relying on the analytic hierarchical Hessian.

Unlike in our previous work, we do not reinitialize the BFGS algorithm from level c to the last level l in the case of a GN algorithm to Newton's method switch, or an active set change on level c. Instead, we keep the last update $\boldsymbol{B}$ and only set to zero the rows and columns of joints that are not occupied any more by the current inactive constraints. In the case of constraint activation we put a small value on the diagonal of occupied joints by the newly activated constraints. Especially with the occurrence of several on and off switches of the same constraint over several control iterations, it seems unreasonable to discard every time of the so far gained second order information.

In some cases it might be more favourable to fully restart the BFGS algorithm in order to avoid second order artefacts of some inactivated constraints which might decelerate convergence of the current active set. However, in our previous work we showed in simulations that the BFGS algorithm possesses quick recovery capabilities, allowing second order artefacts to vanish over the course of few control iterations.

VIII. Practical considerations

A. Handling model insufficiencies

In addition to the trust region adaptation method from our previous work [27] we introduce a further adjustment that operates directly on the BFGS Hessian approximation $\boldsymbol{B}$. For this, we observe the joint accelerations. Once we have several (usually two) consecutive sign changes in the joint

accelerations we add a small value (ζ) on the diagonal of $\boldsymbol{B}$ corresponding to the joint where the event occurred. This way we add additional damping to decelerate this specific joint and avoid potential instability.

B. Computational speed

The problem (33) or (35) is built and solved in a matter of several hundred microseconds. Yet the real computational challenge is the active set method which might require solving (33) or (35) several times until the optimal active set is determined. Especially in dynamically challenging motions, e.g. contact switching, robot falling or being close to falling with almost loosing closure of the friction contacts... we observed cases of above hundred active set iterations. This is caused by an interplay between the dynamic constraints and the trust region constraint and requires further investigation.

Since making the solver [2] faster is highly involved (for example updating the QR-decomposition factors after an active set change or improving the active set search itself) we resorted to a makeshift in which we stop the active set method once a certain level is optimal enough. Usually this would be the contact constraints since we do not want to compromise on their optimality and risk falling.

After every active set iteration we decide upon several criteria whether we stop the active set search:

- Measure the norm of the slack ω and check if it is below a certain threshold ($1e^{-12}\Delta t^2$).
- We have gone through a certain number of iterations in the optimality phase (10 iterations). We switch from the feasibility to the optimality phase with the first deactivation of a constraint.
- We are above a certain number of overall active set iterations (30 iterations).

If so, we exit the solver and are satisfied with the current solution. At this point, the robot motion is physically feasible and all the hard constraints like contact wrench, torque and joint limits, the trust region constraint and the contact constraints are at their optimum. However, lower level constraints are only solved sub-optimally. Since cases of high active set iterations only occur over a limited number of control iterations the optimality of lower priority levels is restored quickly in subsequent control iterations.

We also use a slightly modified version of [2] where we only restart the decomposition from the level where the active set change occurred.

IX. Validation

For validation we conducted three experiments with the position controlled HRP-2Kai robot having 32 DoF (plus 6 DoF for the un-actuated free-flyer), 1.71 m height and 2.11 m arm span. As our solver we use LexLSI [2] which solves problem (33) or possibly (35) with second order augmentation from LexDynBFGS or LexDynAH. LexLSI is based on the active set method and we warm start it at every control iteration with the active set found in the previous control iteration. We solve for the variables $\dot{\boldsymbol{q}}^{(k+1)}$, $\Delta t\boldsymbol{\tau}^{(k+1)}$ and $\Delta t\boldsymbol{\gamma}^{(k+1)}$, see (86). The resulting velocities are then integrated to the joint positions (32) which are sent to the robot with a control frequency of 200 Hz ($\Delta t = 5$ ms).

The hierarchy is then composed as follows:

Hierarchy for LexDynBFGS and LexDynAH

1) $4n_c$ (n_c: number of contacts) bounds on generalized contact wrenches: $\boldsymbol{\gamma} > \mathbf{0}$
2) 32 joint limits using a velocity damper [48]
   - 38 integrated equations of motion
   - 38 torque limits
3) 38 trust region limits
4) 3 **inequality constraints for self-collision avoidance**
5) 18 (12 **for exp. 1**) **geometric contact constraints**
6) 1 equality constraint on the head yaw joint to put the vision marker into the field of view, not for exp. 1
7) 3 **inequality constraints on the CoM**
8) 6 **end-effector equality constraints for left and right hand**
   - 3 **equality constraints to keep the chest orientation upright, not in exp. 1**
9) 3 **stricter inequality constraints on the CoM, not for exp. 1.**
10) 38 constraints to minimize joint velocities: $\dot{\boldsymbol{q}} = \mathbf{0}$
11) $\boldsymbol{4n_c}$ constraints to minimize the generalized contact wrenches: $\boldsymbol{\gamma} = \mathbf{0}$

Constraints in bold have to be considered in the calculation of the hierarchical Hessian. Note that for bound constraints we generally have $\boldsymbol{B} = \mathbf{0}$ and $\boldsymbol{H} = \mathbf{0}$.

The hierarchical separation of the bound constraints on the generalized contact wrenches and the joint limits from the equation of motion allows LexLSI to cheaply handle the variable bounds on the first level.

The trust region constraint is necessary to ensure that the new step $\boldsymbol{\Delta q} = \Delta t\dot{\boldsymbol{q}}^{(k+1)}$ is bounded within a certain neighbourhood Δ of the current state $\boldsymbol{q}$, $\|\boldsymbol{\Delta q}\|_\infty < \Delta$. In this region we 'trust' the second order Taylor approximation of $\Phi(\boldsymbol{q})$ (28) to be accurate enough. In this work, we set $\Delta = 0.01$ rad or m.

Note that the trust region constraint and the joint velocity minimum norm task include the none actuated free-flyer. Indeed, the free-flyer is numerically relevant as it is present in all the Jacobians. Therefore, we put the constraint only after the equation of motion to avoid constricting the free-flyer velocity for example in the case of a robot free fall.

As comparison we use the least squares solver LSSOL [49] which is also used in our laboratory's robot control framework. It has a constraint (level 1) and an objective level (level 2). Thereby, inequality constraints are only allowed on level 1. It is also based on the active set method and we warm start it with the active set found in the previous control iteration. On both levels a soft hierarchy can be established by weighting tasks against each other (therefore we call this solver **W**eighted **L**east **S**quares - WLS). The constrained robot problem is then defined following previous works [5], [50], [51], [52], [53], [6], [54] which make use of control frameworks based on weighted constrained QP's:

Hierarchy for WLS	
1)	• $4n_c$ bounds on generalized contact wrenches: $\boldsymbol{\gamma} > \mathbf{0}$ • 32 joint limits using an acceleration damper [48] • 38 equations of motion • 38 torque limits • 3 inequality constraints for self collision avoidance • 18 (12 for exp. 1) geometric contact constraints • 3 inequality constraints on the CoM
2)	• 6 end-effector equality constraints for left and right hand • 32 equality constraints to maintain a reference posture (with the head yaw joint turned towards the vision marker, exp. 2 and exp. 3) • $4n_c$ constraints to minimize the generalized contact wrenches: $\boldsymbol{\gamma} = \mathbf{0}$

The hierarchy for WLS contains the dynamic constraints (equation of motion, $\boldsymbol{\gamma}$ bounds, $\boldsymbol{\tau}$ bounds), joint limits, self-collision avoidance, CoM task and contact tasks as constraints on level 1. All these tasks have the same priority without weighting. Since the notion of constraint relaxation is not introduced in LSSOL, the feasibility of these constraints has to be guaranteed in order to avoid solver failures. Especially the self-collision avoidance, the contact and the CoM constraints are the source of potential conflict or even of (unresolved) kinematic singularities. This highlights the significance of being able to easily design safe robot problems with LexDynBFGS or LexDynAH.

The reaching task is defined as an objective on level 2.

A posture reference task is also added at the objective level. Similarly to the LM algorithm it acts as a velocity damper, allowing to approach singular configurations. The task is added with a low weight $1e^{-3}$ ($5e^{-2}$ for exp 2 and exp 3), which needs to be tuned depending on the task to be performed. For example reaching for very far away targets requires a higher weight.

Additionally, another task $\boldsymbol{\gamma} = \mathbf{0}$ on the objective level is added to yield a fully determined and full rank problem. It has a small weight $1e^{-5}$ ($1e^{-4}$ for exp. 2 and exp.3).

We solve for the variables $\ddot{\boldsymbol{q}}^{(k+1)}$, $\boldsymbol{\tau}^{(k+1)}$ and $\boldsymbol{\gamma}^{(k+1)}$ and integrate the accelerations twice to the joint positions (57) which are then sent to the robot.

For both solvers we substitute the torques $\boldsymbol{\tau}^{(k+1)}$ with the equation of motion which reduces the number of variables from 108 to 70 for exp. 1, and 112 to 74 for exp. 2 and exp. 3 respectively.

A. Experiment 1

In the first experiment (exp. 1, see Fig. 2) the robot is controlled to take a simple stretching pose. It shows that LexDynBFGS and LexDynAH enable numerically stable robot behaviour even with the presence of kinematic and algorithmic singularities. At the same time the task error norm is minimized to a higher degree w.r.t WLS.

For this we establish two contacts with the environment on the two feet standing on the ground. Each foot consists of four contact points (overall $n_c = 8$) placed on each corner of the rectangular foot.

The robot then continues to move the left hand to $[0.4, 0.5, 1.3]$ m and the right hand to $[0.4, -0.5, 1.3]$ m which is on each side of the robot's chest. After that it targets $[0.4, -2, 2]$ m and $[0.4, 2, 2]$ m which are both out-of-reach positions on its left top and right top side respectively. This forces the robot to take a stretched configuration along the convergence process. At this point, the end-effector tasks are in algorithmic singularity due to conflict with the geometric contact constraints.

Thereby, LexDynBFGS shows the best minimization of the norm of the Euclidean distance of the left and right hand to their targets (see Fig. 3). Both legs and arms are fully stretched. Consequently, the contact Jacobians of the left and right foot are singular. This seemingly does not influence the robot's joint velocity (see Fig. 4), joint torque (see Fig. 12a) and contact wrench (see Fig. 13a) behaviour due to the full rank property of the equation of motion.

The convergence is the longest out of the three solvers as there is a large period of flat curvature with very small $\boldsymbol{y}^T \boldsymbol{\Delta q}^{(k)} \approx 1e^{-16}$ (between ca. 45 s and 90 s). This leads to slow joint motions since the BFGS Hessian approximation is not updated but rather remains as a static damping term similar to the LM algorithm.

The joint velocities are fairly smooth (see Fig. 4) and pose no problem on the real robot. If for some end-user the movements are too jerky a weighted and dotted identity matrix $\alpha \boldsymbol{I}^*$ could be added to $\boldsymbol{B}$. However, this has a light side effect on the convergence.

The switch to Newton's method is taking place on the way to the first way point at around 25 s. There is a quick switch to Newton's method and back at around 105 s on the left and right foot position task but does not further influence the robot behaviour (see Fig. 5).

For most of the time there is only one active set iteration per control iteration, yielding computation times well below 5 ms. However, at approximately 40 s there is a peak in active set iterations of 17 with a loop time of just below 5 ms (see Fig. 6). Other peaks are computational artefacts on the home PC which was used to remote control the robot (PC Intel Core i7-4720HQ CPU @ 2.60GHz with 8 GB of RAM).

Similar behaviour is seen for LexDynAH but it converges faster to a less optimal minimum (see Fig. 3 for the task error norm and Fig. 7 for joint behaviour)

In Fig. 9 a clear increase in computation times can be seen at around 26 s when the second order augmentation and the corresponding Higham regularization with SVD decomposition starts (see Fig. 8).

Note that the Hessian is computed all the time even if there is no augmentation. Its computation time is included in 'All'. Computing the CoM Hessian of the 38 DoF HRP-2Kai robot takes about 150 μs and calculating all the Hessians accounts for approximately less than 1 ms.

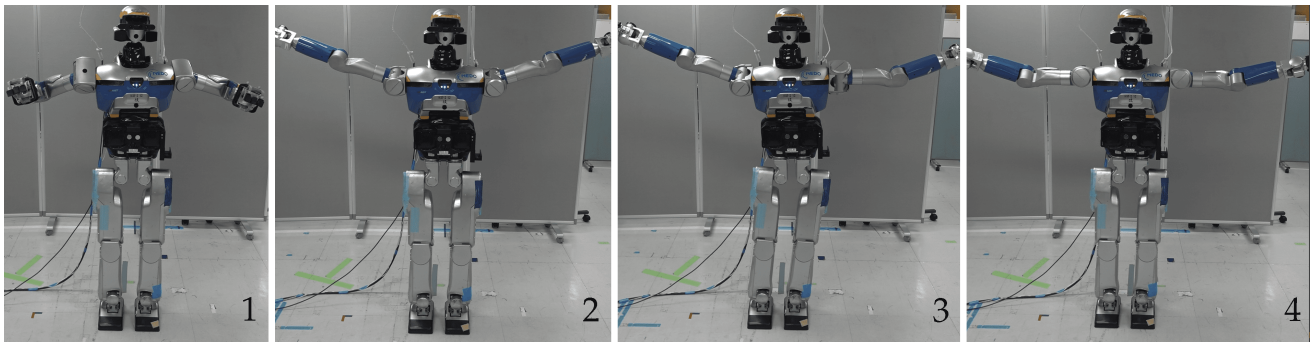


Fig. 2: **Exp. 1**, from left to right: 1.: LexDynBFGS, the robot has moved both its hands to the first target, 2.: LexDynBFGS, the robot has converged with both arms and both legs stretched 3.: LexDynAH, the robot has converged with one arm and both legs stretched, 4.: WLS, the robot has converged with only the legs stretched.

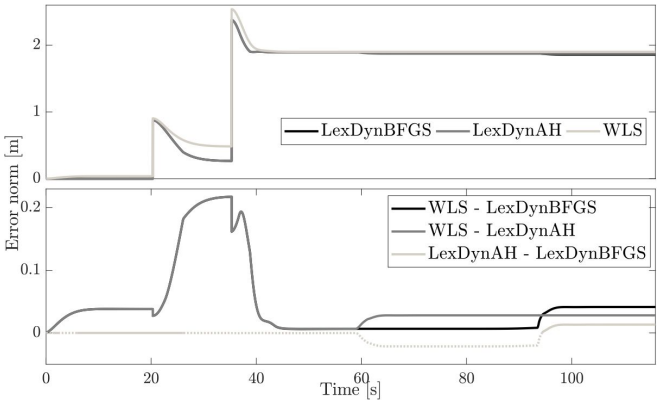


Fig. 3: **Exp. 1**, comparison of sum of the error norms of the right and the left hand. LexDynBFGS has a final error of 1.86 m, LexDynAH of 1.87 m and WLS of 1.9 m. The lower graph shows the differences of the error norms of the different methods. The data of LexDynAH and WLS is synchronized with the one of LexDynBFGS.

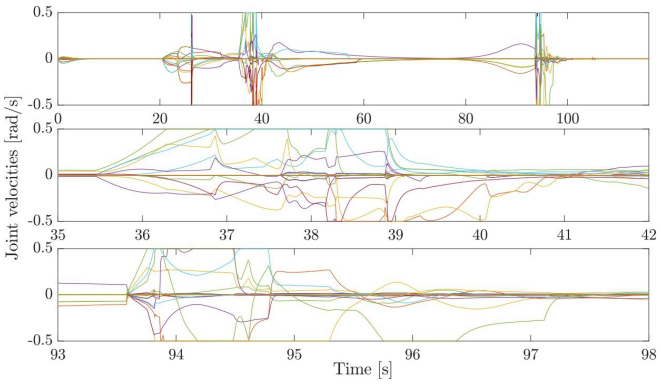


Fig. 4: **Exp. 1**, LexDynBFGS, joint velocities

The WLS method shows the worst convergence (see fig. 3). Especially during the first motion the robot fails to lift both its arms up due to conflict with the posture reference task. The joint velocities are very smooth but slow (see fig. 10) which is behaviour typically seen for the LM algorithm. Consequently, there are less active set iterations

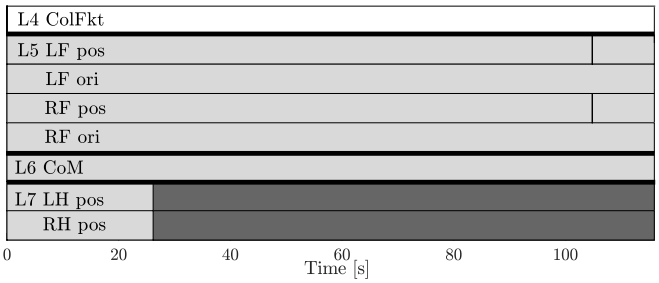


Fig. 5: **Exp. 1**, LexDynBFGS, map of activity (light gray) and Newton's method (dark gray)

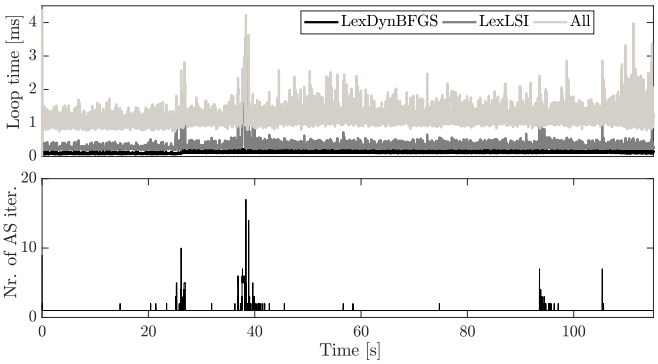


Fig. 6: **Exp. 1**, LexDynBFGS, computation times and number of active set iterations. LexDynBFGS peaks at 376 μs, LexLSI at 3.55 ms and overall at 4.23 ms. The maximum number of active set iterations is 17 iterations.

than seen for LexDynBFGS and LexDynAH due to less abrupt motion changes.

The computation times for LSSOL for a single iteration are a multiple of the ones of LexLSI. However, it only increases slightly with a higher number of active set iterations. Reason is that LSSOL cheaply updates decomposition factors for the current active set which have been computed for the previous one (see Fig. 11).

Figure 12 shows a comparison of the computed joint torques for LexDynBFGS and WLS. They are approximately of the same magnitude and show the validity of our approach

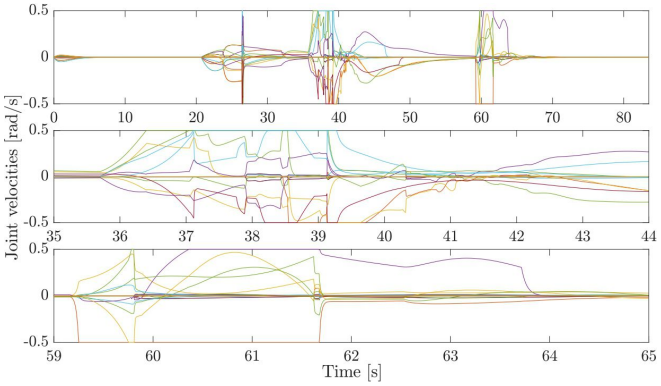


Fig. 7: **Exp. 1**, LexDynAH, joint velocities

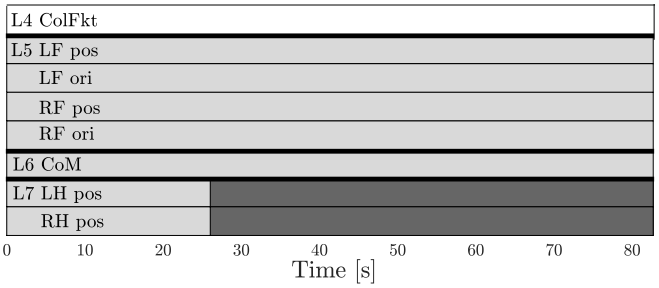


Fig. 8: **Exp. 1**, LexDynAH, map of activity (light gray) and Newton's method (dark gray)

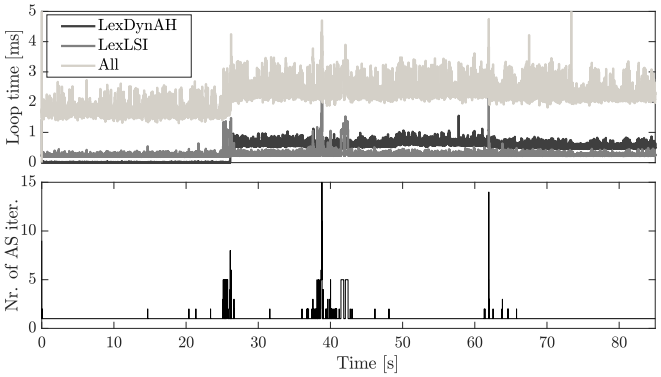


Fig. 9: **Exp. 1**, LexDynAH, computation times and number of active set iterations. LexDynAH peaks at 1.55 ms, LexLSI at 2.96 ms and overall at 4.75 ms. The maximum number of active set iterations is 15.

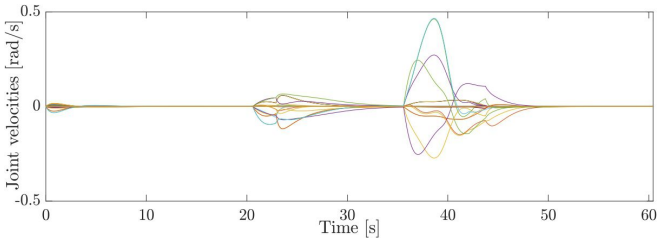


Fig. 10: **Exp. 1**, WLS, joint velocities.

of forward integrating the accelerations in the equation of motion. The joint torques for LexDynAH are very similar

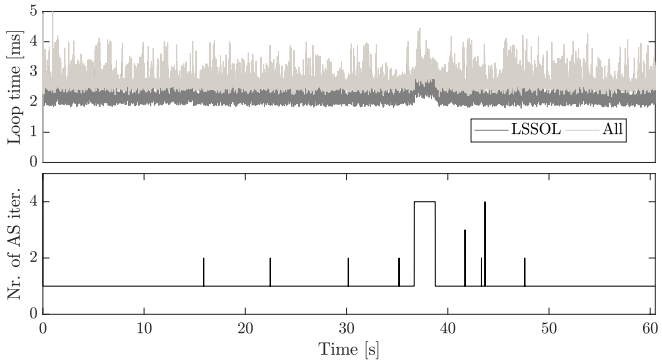
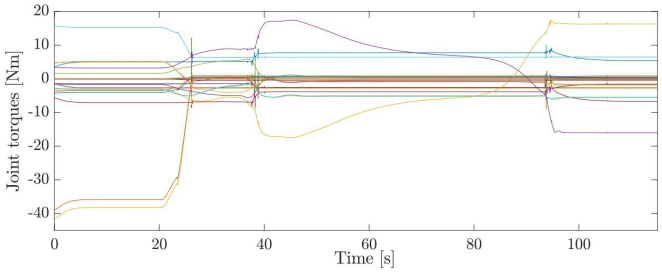
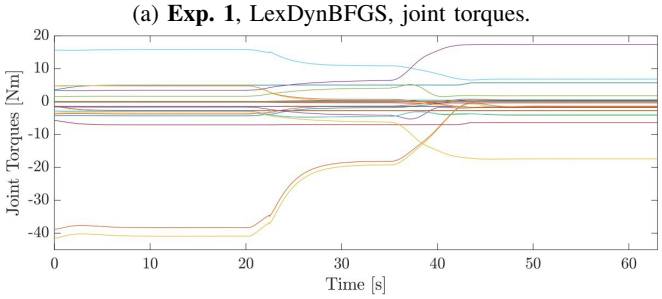


Fig. 11: **Exp. 1**, WLS, computation times and number of active set iterations. LSSOL peaks at 3.76 ms and overall at 4.46 ms with 4 active set iterations.



(a) **Exp. 1**, LexDynBFGS, joint torques.



(b) **Exp. 1**, WLS, joint torques.

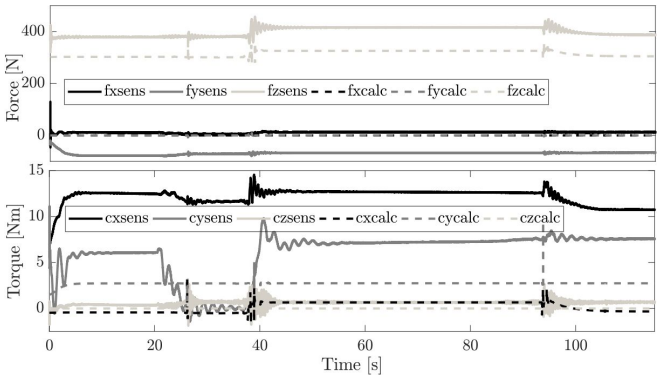
Fig. 12: Comparison of the joint torques between LexDynBFGS and WLS. They are of similar magnitude.

to the ones of LexDynBFGS and are therefore not depicted here.

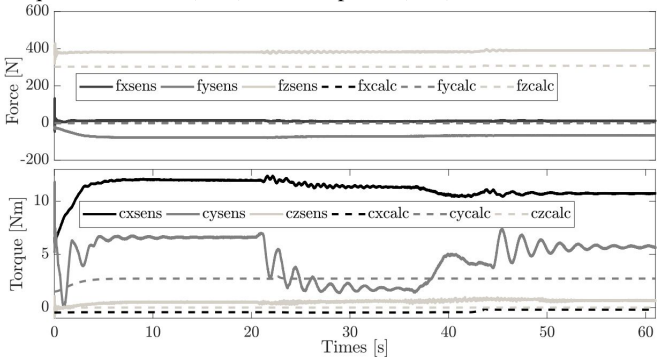
Figure 13 shows a comparison of the computed and measured ground reaction forces and torques of the left foot for LexDynBFGS and WLS. They are approximately of the same magnitude and again show the validity of our approach of forward integrating the accelerations in the equation of motion. The discrepancy between measured and sensed values comes to a large extent from the unmodelled elasticity of the ankle pitch joints but is consistent for LexDynBFGS and WLS. The ground reaction forces and torques for LexDynAH are very similar to the ones of LexDynBFGS and therefore are not depicted here.

B. Experiment 1 without or badly-tuned augmentation

For reference we show the robot joint behaviour for exp. 1 in the case of the pure GN algorithm without the switch to Newton's method in case of singularities.



(a) **Exp. 1**, LexDynBFGS, left foot ground reaction forces and torques, measured (sens) and computed (calc).



(b) **Exp. 1**, WLS, left foot ground reaction forces and torques, measured (sens) and computed (calc).

Fig. 13: **Exp. 1**, comparison of the ground reaction forces and torques of the left foot between LexDynBFGS and WLS. They are of similar magnitude.

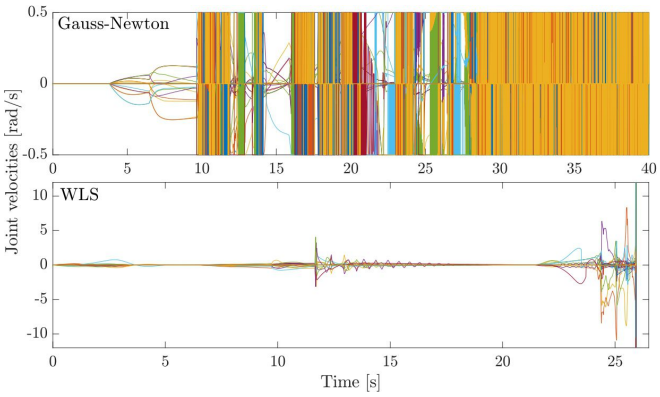


Fig. 14: **Exp. 1** without or badly-tuned augmentation, joint velocities. The pure GN algorithm suffers from important joint oscillations while WLS with only 1/1000 of the original weight for the posture reference task fails completely at 25.9 s.

Since singular behaviour would be highly harmful to the robot we conducted this experiment only in simulation.

The robot is clearly unstable on joint level (see Fig. 14) and sways heavily (see video). Reason is that joint torque limits are reached repeatedly (see Fig. 15) since the joint oscillations require very high motor torques.

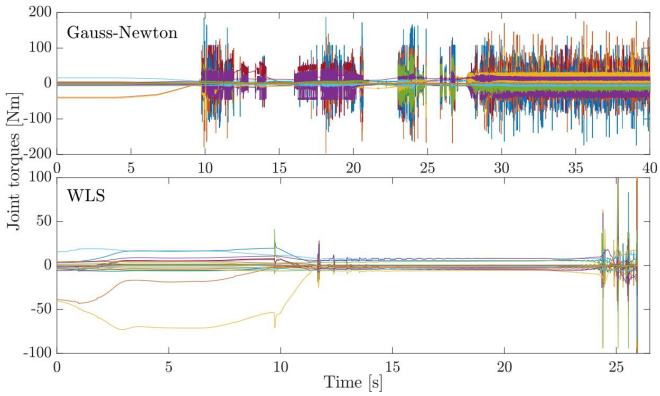


Fig. 15: **Exp. 1** without or badly-tuned augmentation, joint torques. Upper graph: the joint oscillations require high motor torques. Lower graph: Joint torques for WLS with a weight of $1e^{-6}$ for the posture reference task.

Additionally, exp. 1 is conducted with WLS where the reference posture task is weighted with only $1/1000$ of its original weight. The weight used is then $1e^{-6}$. While there are no joint oscillations the robot sways slightly until LSSOL fails completely at around 26 s with numerically high joint velocities due to approaching a singular configuration.

C. Experiment 2

The second experiment (exp. 2, see Fig. 16) is designed in such a way that a third contact between the left hand and a rigid pole must be established in order to prevent falling. This contact is not a friction contact but a fixed contact such that we only need to define one contact point here (overall $n_c = 9$). Also, the bound $\boldsymbol{\gamma} > \mathbf{0}$ needs to be omitted for this contact since we allow negative contact forces if the robot is 'hanging' from the pole.

In the following we describe the experiment for LexDynBFGS.

At first, the robot moves its left hand in front of a vertical metal pole which is supposed to be grabbed at approximately $[0.5, 0.25, 1]$ m. The exact pole position is determined via marker based vision provided by the `whycon` library [55]. During the movement, the CoM task on level 8 constrained to the bounding box $[\pm 0.03, \pm 0.1, \pm\infty]$ m gets activated. The right hand on the next level 9, which must remain at its current position, is therefore in conflict and starts to move slightly backwards. This is a purely algorithmic singularity and triggers the switch to Newton's method. Figure 19 shows the activation of the CoM task and the augmentation of the right hand position task and the chest orientation task at around 8 s.

The left gripper is closed and the left-hand-to-pole contact is added to the equation of motion. Then we open up the CoM bounding box in x-direction from $[\pm 0.03, \pm 0.1, \pm\infty]$ m to $[\pm 0.2, 0.05 \pm 0.05, \pm\infty]$ m since we have increased the support area of the robot in the sagittal plane due to the additional contact (see Fig. 21). However, the robot requires several control iterations with a high number of active set iterations to adjust the robot

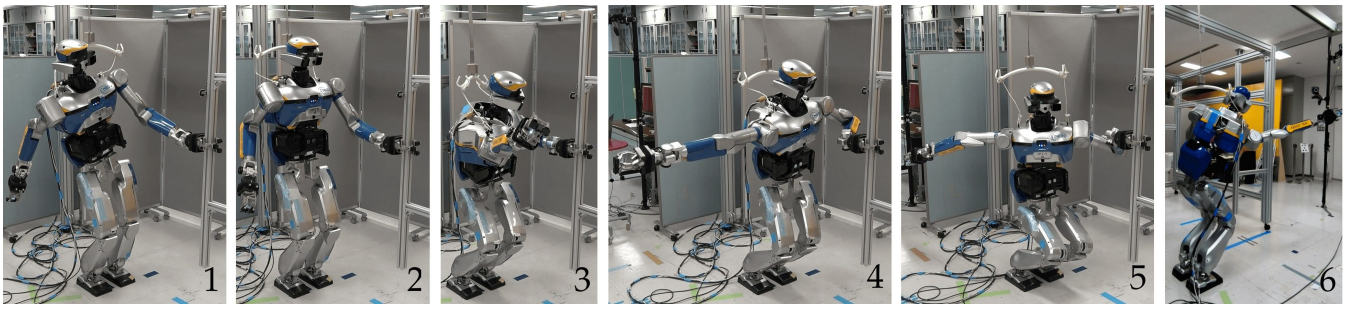


Fig. 16: Pictures of HRP-2Kai performing **exp. 2**, LexDynAH, from left to right: **1.:** The left hand has grabbed the pole while the right hand task on level 9 is in conflict with the CoM task on level 8. **2.:** The CoM on level 8 switched from box 1 to box 2 and is not in conflict with the right hand task any more The right hand task on level 9 is not augmented any more **3.:** HRP-2Kai is in full forward stretch. **4.:** The robot moves its right hand to the back. **5.:** HRP-2Kai is in full backward stretch. **6.:** The robot during its second forward stretch.

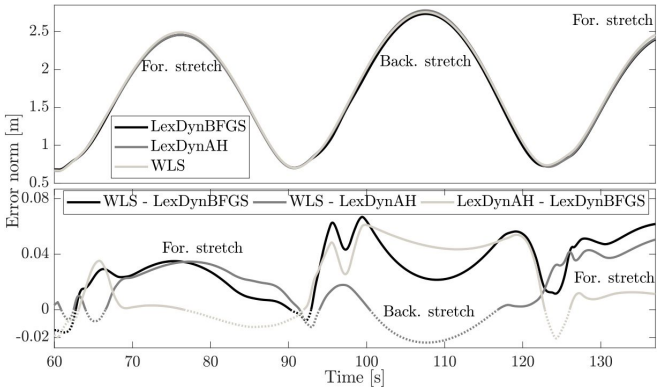


Fig. 17: **Exp. 2**, comparison of the error norm of the right hand tracking the swinging target during forward (for.) and backward (back.) stretch. The lower graph shows the differences of the error norms of the different methods. The data of LexDynAH and WLS is synchronized with the one of LexDynBFGS.

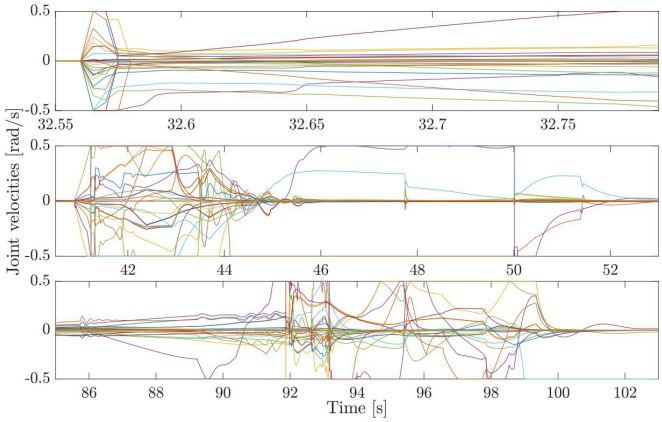


Fig. 18: **Exp. 2**, LexDynBFGS, joint velocities. The upper graph shows the moment when the CoM is released and the 52 active set iterations occur (followed by 5 occurrences of 30 iterations until 32.7 s). The middle one shows the moment when the right hand stretches to the right and starts being augmented. The lower graph shows the joint velocities when the robot stretches to the back and crouches.

state. In Fig. 20 at the 32 s mark, it can be seen that within 28 control iterations, peaks of 1×52 and 5×30 active set iterations occur. Figure 19 shows that at the instant of the CoM release at around 32 s both CoM tasks at level 8 and 10 are activated and shortly after deactivated again. Similar behaviour is seen for the collision constraint between the left elbow and the torso.

Furthermore, the upper graph of Fig. 18 shows how the velocity suddenly increases from zero to maximum velocity 0.5 s during the CoM release. This requires large joint torques of around 150 Nm as can be seen from the lower graph of Fig. 26a. The corresponding dynamic effects need to be adjusted for in the trust region and the contact force constraints (with interplay) which leads to the large number of active set iterations.

Note that without the active set iteration limitation method presented in section VIII-B the active set iteration count might easily go up to 200 or more.

At around 36 s the augmentation of the level 9 position task stops as the right hand arrives back at its prescribed

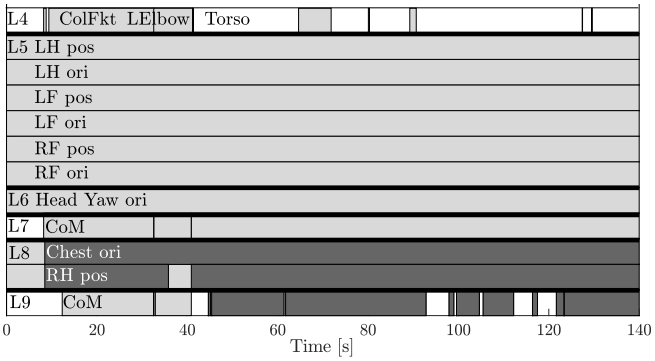


Fig. 19: **Exp. 2**, LexDynBFGS, map of activity (light gray) and Newton's method (dark gray)

position (see Fig. 19).

The right hand then first tries to reach $[0, -1.5, 1]$ m and continues to follow a target swinging from the front

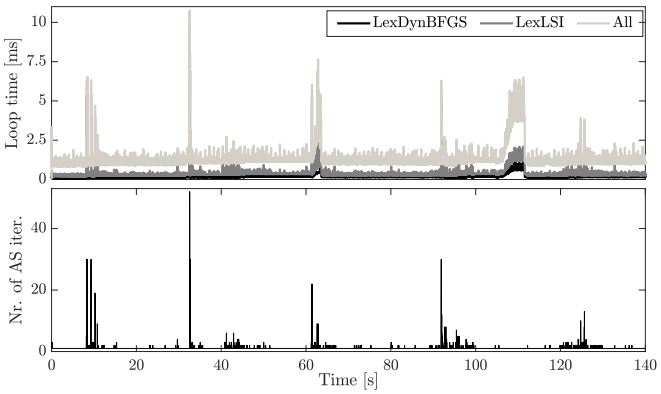


Fig. 20: **Exp. 2**, LexDynBFGS, computation times and number of active set iterations. LexDynBFGS peaks at 1.11 ms, LexLSI at 9.66 ms and overall at 10.75 ms. The maximum number of active set iterations is 52 iterations. At around 105 s and for the duration of a few seconds there is a computational artefact from the robot controlling PC.

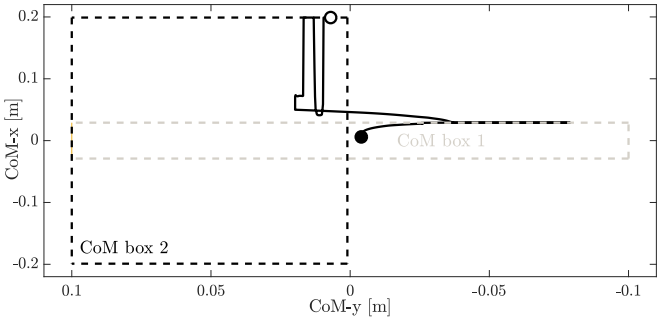


Fig. 21: **Exp. 2**, LexDynBFGS, CoM movement. The CoM starts at the black dot and moves until it arrives at the white dot, first being constrained in box 1 and then in box 2.

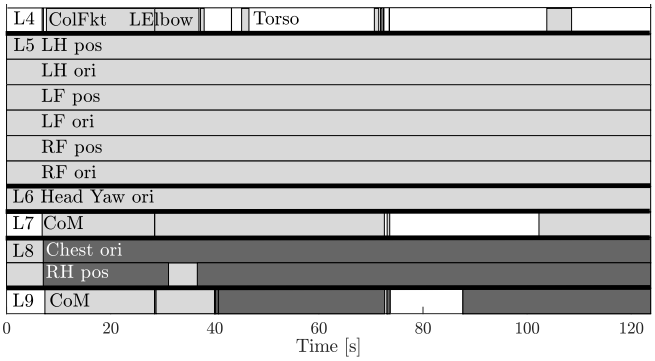


Fig. 22: **Exp. 2**, LexDynAH, map of activity (light gray) and Newton's method (dark gray)

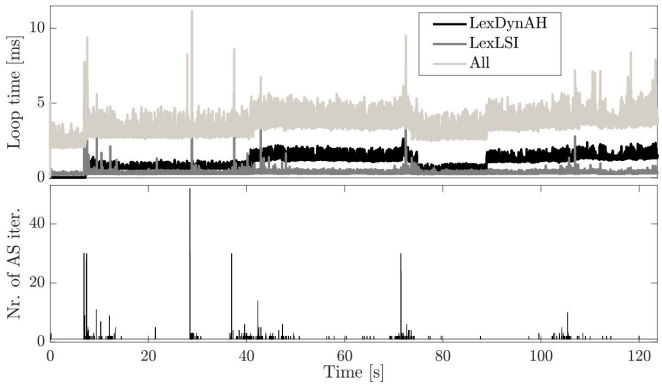


Fig. 23: **Exp. 2**, LexDynAH, computation times and number of active set iterations. LexDynAH peaks at 2.63 ms, LexLSI at 8.03 ms and overall at 11.16 ms. The maximum number of active set iterations is 52 iterations.

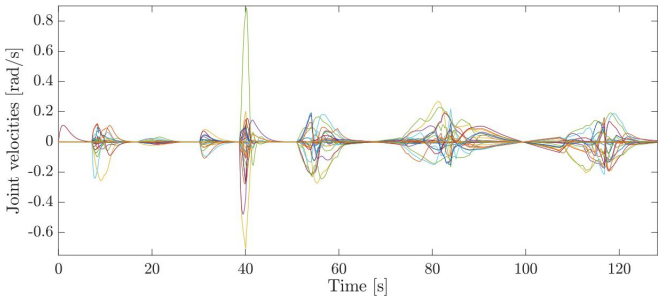


Fig. 24: **Exp. 2**, WLS, joint velocities.

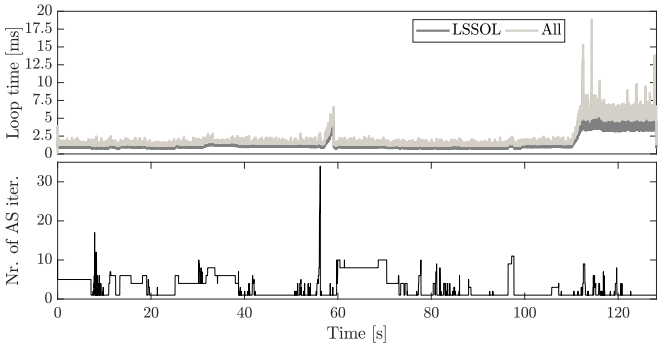
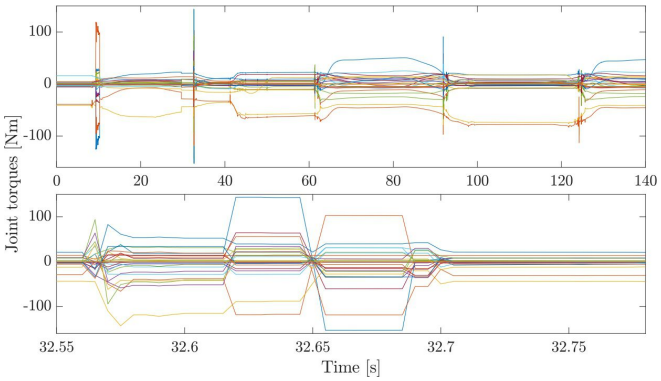


Fig. 25: **Exp. 2**, WLS, computation times and number of active set iterations. In the time segment from 0 s to 56 s (excluding the computational artefact from the controller PC, but including the active set peak), LSSOL peaks at 2.20 ms and overall at 2.94 ms with 34 active set iterations. Again, there are some computational artefacts on the robot controlling PC from 56 s and from 110 s.

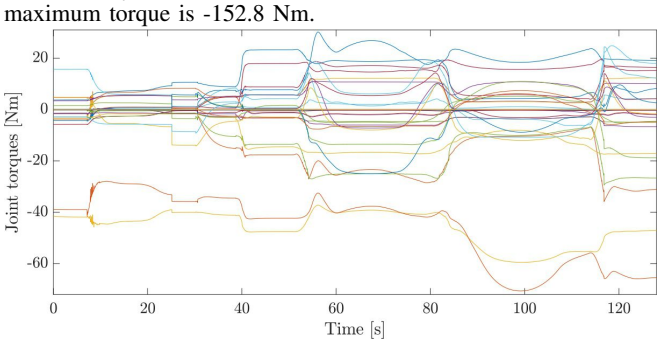
$[3, -1.5, 2]$ m to the back $[-3, -1.5, 0]$ m of the robot. This target is always out of reach.

From Fig. 27a it can be seen that during the forward stretches from 62 s and from 125 s the robot pushes its left hand against the pole with more than 100 N. This shows the necessity of this contact to prevent falling.

In Fig. 21, the CoM moves from the black to the white dot. During the stretch motions it is well outside the support polygon of the feet which covers about $[\pm 0.1, \pm 0.2, \pm\infty]$ m. The CoM at level 8 is first constrained to box 1 and then to box 2. Note that the CoM constraint at the white dot is also activated in y-direction. This is due to the velocity damper which starts to act 0.02 m before the actual bounding box edge. The continuous augmentation of the right hand and the chest task at level 9 occupies all joints. Therefore, the CoM

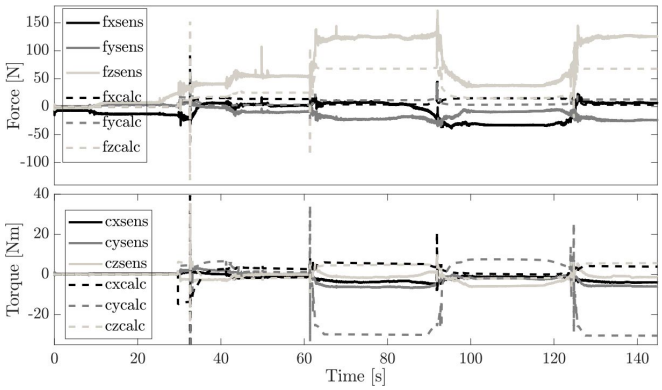


(a) **Exp. 2**, LexDynBFGS, joint torques. The upper graph shows the whole experiment while the lower graph only shows the moment when the CoM is released and the 52 active set iterations occur (followed by 5 occurrences of 30 iterations until 32.7 s). The maximum torque is -152.8 Nm.

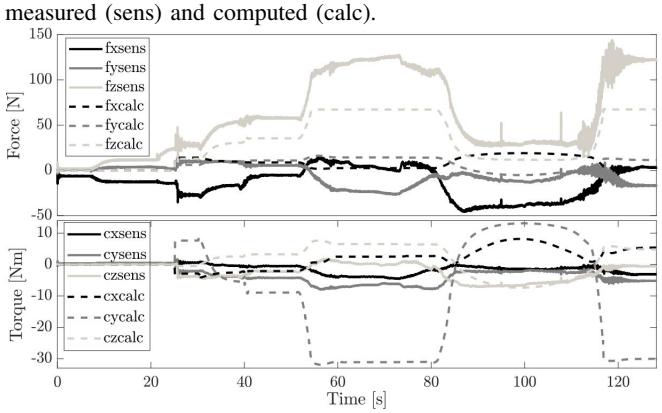


(b) **Exp. 2**, WLS, joint torques. The maximum joint torque is -71 Nm.

Fig. 26: Comparison of the joint torques between LexDynBFGS and WLS.



(a) **Exp. 2**, LexDynBFGS, left hand reaction forces and torques, measured (sens) and computed (calc).



(b) **Exp. 2**, WLS, left hand reaction forces and torques, measured (sens) and computed (calc).

Fig. 27: Comparison of the reaction forces and torques of the left hand grabbing the pole between LexDynBFGS and WLS

constraint on level 10 has no influence and is therefore not shown here.

Figure 17 shows the norms of the tracking error of the right hand during the stretch motions. Especially during the second forward stretch LexDynBFGS gets over 0.05 m closer to the target than WLS due to fully stretching its arm into kinematic singularity. Overall, most of the time the error norm difference $\text{Err}_{\text{WLS}} - \text{Err}_{\text{LexDynBFGS}} > 0$ and shows the better convergence of LexDynBFGS compared to WLS.

While LexDynAH also allows limbs to be fully stretched into kinematic singularities, it has a higher error norm than WLS especially during the backward stretch (see Fig. 17). This is despite the fact that the robot crouches more than seen for WLS (see video). The reason is partly due to local minima created by joint limits. However, at forward stretches LexDynAH performs better than WLS ($\text{Err}_{\text{WLS}} - \text{Err}_{\text{LexDynAH}} > 0$).

In Fig. 23 it can be seen that the average overall computation time 'All', which includes the calculation time of the analytic Hessian, increases by about 1.5 ms compared to the ones of LexDynBFGS in Fig. 20. An additional increase can be observed at 40 s for 'LexDynAH' when the CoM task on level 10 requires augmentation too (see Fig. 22). Now two expensive SVD decompositions for the Higham regularizations of the hierarchical analytic Hessian on level 9 and 10 are required.

In general, LexDynAH behaves very similar to LexDynBFGS and shows the capabilities of the BFGS algorithm to provide a valid approximation of the analytic hierarchical Hessian. Therefore, the joint velocities, CoM motion, ground and pole reaction forces and joint torques are not shown since they are very similar to the ones of LexDynBFGS.

As for WLS, it shows again the worst convergence (see Fig. 17) with very smooth joint velocities (see Fig. 24). Especially during the forward stretches the robot does not fully stretch its right arm. Also during the backward stretch the robot crouches to a lesser extent than seen for LexDynBFGS and LexDynAH (see video).

Since there are no jerky movements the joint torques do not show high peaks (see 26b). What is quite interesting is that there is still a peak of 32 active set iterations (see Fig. 25) during the first forward stretch at 55 s. However, the computation time only increases by a fraction due to updating factorizations in LSSOL.

D. Experiment 3

With the last experiment (exp. 3) we show the importance of handling kinematic and algorithmic singularities automat-

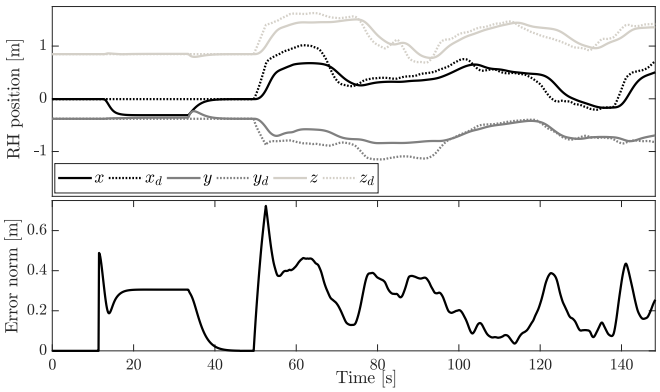


Fig. 28: **Exp. 3**, LexDynBFGS, right hand task error. The upper graph shows the actual right hand position in x, y and z direction and the desired ones x_d, y_d and z_d. The lower graph shows the error norm of the right and left hand. The error of the left hand holding the rail can be considered zero once the contact is created at around 20 s.

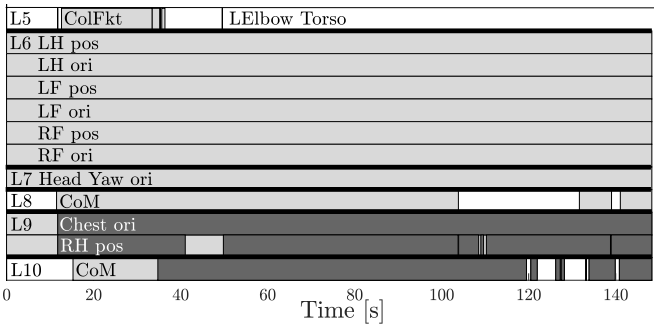


Fig. 29: **Exp. 3**, LexDynBFGS, map of activity (light gray) and Newton's method (dark gray).

ically.

This experiment is set up similarly to exp. 2, except that the robot right hand follows a target tracked by a motion capture system (10 Krestel cameras from Motion Analysis). Typical applications could be human to robot hand-overs. In such situations, and especially for a humanoid robot with multi-level prioritized constraints, it is difficult to determine the robot workspace beforehand. Here our proposed methods LexDynBFGS and LexDynAH allow to only have approximate knowledge or no knowledge at all of the robot workspace while still enabling safe control for the robot. A hand-over could happen at the border of the workspace with the end-effector being in kinematic singularity or being in algorithmic conflict with a stability task like the CoM task on a higher priority level. At the same time, the whole possible workspace is used without restricting it artificially for example with a badly tuned LM algorithm. Thereby, a non-adaptive LM algorithm can always be considered badly tuned. The LM algorithm needs to be tuned with a certain overhead in order to ensure non-singular behaviour over a range of robot configurations. This leads to bad convergence in singularity-free configurations where no damping is required.

Since the end-effector target is defined by user input, for the lack of comparability we only conducted the experiment with LexDynBFGS.

In order to prevent too fast motions we set the right hand proportional task gain to $k_p = 0.5$ instead of $k_p = 1$ in the two previous experiments. Additionally, the CoM bounding box 2 is chosen a bit more conservatively as $[\pm 0.1, 0.05 \pm 0.05, \pm\infty]$ m instead of $[\pm 0.2, 0.05 \pm 0.05, \pm\infty]$ m in exp. 2.

The right hand tracking error is given in Fig. 28. At around 10 s the left hand moves towards the pole before closing the gripper and adding the contact to the equation of motion. At around 35 s the CoM constraint changes from box 1 to box 2. The right hand stops being in algorithmic conflict with the CoM constraint and moves to its original position (see Fig. 29 at around 42 s, the right hand is not augmented any more). From around 50 s onwards, the right hand is following the position of the middle marker of a wand with 3 markers tracked by the motion capture system. The provided position x_d, y_d and z_d is filtered by a 3 s moving average filter (see dotted lines in upper graph of Fig. 28). We first move the marker outside of the approximate workspace of the robot (until 100 s). The right arm is in kinematic singularity and in conflict with the CoM constraint on level 8. We then continue to move the marker within the workspace of the robot. The CoM task at level 8 gets deactivated. However, the right hand switches from Newton's method to the GN algorithm only for a short period from 108.5 s to 109 s and from 109.75 s to 110.25 s before being augmented again. Note that the task error $\boldsymbol{e}$ does not necessarily need to be zero for this to happen. Our switching method only indicates that within the context of our linearized model we can (more or less) numerically stably provide the error velocity prescribed by the PD control emulated in velocity. Hence, there is no conflict with the kinematic restrictions of the robot or other constraints with the same or higher priority. The switching method could be tuned to be less sensitive although this could increase the risk that a singular task might not get augmented in certain situations. A good middle ground needs to be found in tuning the parameter ν of our switching method. Future work could also include the implementation of more accurate switching methods for example by observing the matrix rank of the constraint Jacobians or their projections onto the Jacobians of higher priority tasks.

E. Performance comparison

The overall performance of LexDynBFGS and LexDynAH (in combination with LexLSI) and WLS (in combination with LSSOL) in the three experiments is compiled comprehensively in table I.

LexDynBFGS and LexDynAH both performed best in the category 'Error convergence'. The performance of WLS is highly dependent of the level of chosen damping. Its behaviour can only be considered acceptable at best since determining 'perfect' damping without proper adaptation methods seems intractable.

The same holds for the 'stability' and 'smoothness' of the joint trajectories. If the damping weights for the posture

	LexDyn BFGS	LexDyn AH	WLS
Error convergence	+	+	o / –
Joint stability	+	+	+ / –
Joint smoothness	o	o	+ / –
Easiness of use	+	+	o
Computation time	+ / –	o / –	o
Lexico. separation	+	+	o
Handling of infeas. ctr.	+	+	–
Handling of inequ. ctr.	+	+	o

TABLE I: Comprehensive overview of the performance of LexDynBFGS, LexDynAH (in combination with LexLSI) and WLS (in combination with LSSOL) in the three experiments. The symbols +, o, – indicate best, acceptable and worst performance in the corresponding evaluation criteria. Criteria solely dependent on the respective solver are coloured in grey.

reference task of WLS are too low, the behaviour will be unstable and therefore not smooth. If the damping weights are too high we are at the other end of the spectrum with very smooth joint trajectories but bad error convergence. LexDynBFGS and LexDynAH is a very stable alternative with smooth joint trajectories. Especially, there is no need for damping tuning which makes it very easy to use ('Easiness of use').

As mentioned in sec. VIII-B, the 'computation times' for LexLSI can be very large depending of the number of active set iterations. However, with only a few active set iterations the computation times are a magnitude smaller than the ones seen for LSSOL. However, LexDynAH requires the expensive SVD decomposition for the Higham regularization of the analytic hierarchical Hessian which makes its computation time only acceptable at best.

Further advantages of LexLSI compared with LSSOL are the possibility of introducing more hierarchy levels ('lexicographical (lexico.) separation'). This allows clear distinction between feasibility, safety and stability constraints. LexLSI is also able to handle 'infeasible constraints' (infeas. ctr.) and 'inequality constraints' (inequ. ctr.) on any priority level which is not possible with LSSOL.

X. Conclusion

In this paper we extended our work [27] on handling kinematic and algorithmic singularities in kinematic robot control to dynamically feasible kinematic robot control. We showed that it enables the robot to approach and reach singular configurations smoothly and without the high velocities typically seen for unresolved singularities. Thereby, our approach consisting of forward integrating the accelerations proved viable to shift second-order motion controllers and the equation of motion into the velocity domain.

Evaluated in three experiments on the position controlled HRP-2Kai robot, our method supersedes classical damping methods in terms of accuracy and error norm reduction. At the same time the notion of hierarchy enables formulating problems with strict safety prioritizations. Some cases of jerky joint movements are handled easily by adding a small damping term.

Also, we introduced the possibility of augmenting the problem using the analytic hierarchical Hessian. Like the BFGS algorithm, the computational effort for the Hessian calculation is in $\mathcal{O}(n^2)$. However, the Hessian can become indefinite if not close to the solution. This requires an expensive SVD decomposition of order $\mathcal{O}(mn^2)$ in order to enforce semi-positive-definiteness of the Hessian using the Higham regularization.

Model inaccuracies are accounted for with a trust region adaptation method which is customized for constrained optimization. Also, we slow down particular joints by manipulating the BFGS approximation of the Hessian as a result of observing the joint accelerations. For future work it is desirable to properly handle cases of negative definite curvature in order to prevent slowing down convergence by augmenting with the last positive definite update.

Computation times of the hierarchical least squares solver are a limiting factor of our approach of augmentation with the BFGS Hessian as well as the analytic Hessian. Without important dynamic effects, the active set iteration count is limited to a few iterations. However, situations where several torque, trust region and contact force constraints become active are more challenging for the active set search. Our future work needs to focus on handling these situations cheaply, for example by factorization updates in the solver by [2] or a more effective active set search.

In future work we would like to extend our work to inverse dynamics control including force control and validate our results on torque controlled robots.

XI. Acknowledgement

We would like to deeply thank the Inria team of Pierre-Brice Wieber and Dimitar Dimitrov for providing us with the code for LexLSI [2] which was indispensable for this work. Furthermore, we would like to acknowledge that sec. III is the result of fruitful discussions with Pierre-Brice Wieber.

References

[1] 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.

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

[3] 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.

[4] 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.

[5] Y. Abe, M. da Silva, and J. Popović, "Multiobjective control with frictional contacts," in *Eurographics/ACM SIGGRAPH Symposium on Computer Animation*, San Diego, California, 2-4 August 2007, pp. 249–258.

[6] J. Vaillant, A. Kheddar, H. Audren, F. Keith, S. Brossette, A. Escande, K. Bouyarmane, K. Kaneko, M. Morisawa, P. Gergondet, E. Yoshida, S. Kajita, and F. Kanehiro, "Multi-contact vertical ladder climbing with an HRP-2 humanoid," *Autonomous Robots*, vol. 40, no. 3, pp. 561–580, 2016.

[7] K. Nishiwaki and S. Kagami, "Online walking control system for humanoids with short cycle pattern generation," *The International Journal of Robotics Research*, vol. 28, no. 6, pp. 729–742, 2009. [Online]. Available: https://doi.org/10.1177/0278364908097883

[8] M. Kalakrishnan, J. Buchli, P. Pastor, M. Mistry, and S. Schaal, "Learning, planning, and control for quadruped locomotion over challenging terrain," *The International Journal of Robotics Research*, vol. 30, no. 2, pp. 236–258, 2011. [Online]. Available: https://doi.org/10.1177/0278364910388677

[9] S. Mason, N. Rotella, S. Schaal, and L. Righetti, "Balancing and walking using full dynamics lqr control with contact constraints," in *IEEE-RAS 16th International Conference on Humanoid Robots (Humanoids)*, Nov 2016, pp. 63–68.

[10] V. Bonnet, K. Pfeiffer, P. Fraisse, A. Crosnier, and G. Venture, "Self-Generation of Optimal Exciting Motions for Identification of a Humanoid Robot," *International Journal of Humanoid Robotics*, vol. 15, no. 6, p. 1850024, Dec. 2018. [Online]. Available: https://hal.archives-ouvertes.fr/hal-02048085

[11] K. Bouyarmane, K. Chappellet, J. Vaillant, and A. Kheddar, "Quadratic programming for multirobot and task-space force control," *IEEE Transactions on Robotics*, vol. 35, no. 1, pp. 64–77, Feb 2019.

[12] 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.

[13] 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

[14] 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.

[15] T. Shamir, "The singularities of redundant robot arms," *The International Journal of Robotics Research*, vol. 9, no. 1, pp. 113–121, 1990. [Online]. Available: https://doi.org/10.1177/027836499000900105

[16] V. D. Tourassis and J. Marcelo H. Ang, "Identification and analysis of robot manipulator singularities," *The International Journal of Robotics Research*, vol. 11, no. 3, pp. 248–259, 1992. [Online]. Available: https://doi.org/10.1177/027836499201100307

[17] W. Khalil and E. Dombre, "Chapter 5 - direct kinematic model of serial robots," in *Modeling, Identification and Control of Robots*, W. Khalil and E. Dombre, Eds. Oxford: Butterworth-Heinemann, 2002, pp. 85 – 115. [Online]. Available: http://www.sciencedirect.com/science/article/pii/B9781903996669500051

[18] 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.

[19] A. A. Maciejewski and C. A. Klein, "The singular value decomposition: Computation and applications to robotics," *The International Journal of Robotics Research*, vol. 8, no. 6, pp. 63–79, 1989. [Online]. Available: https://doi.org/10.1177/027836498900800605

[20] S. Chiaverini, B. Siciliano, and O. Egeland, "Review of the damped least-squares inverse kinematics with experiments on an industrial robot manipulator," *IEEE Transactions on Control Systems Technology*, vol. 2, no. 2, pp. 123–134, June 1994.

[21] P. Baerlocher and R. Boulic, "An inverse kinematics architecture enforcing an arbitrary number of strict priority levels," *Visual Computer*, vol. 20, no. 6, pp. 402–417, 2004.

[22] 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

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

[24] P. Harish, M. Mahmudi, B. L. Callennec, and R. Boulic, "Parallel inverse kinematics for multithreaded architectures," *ACM Trans. Graph.*, vol. 35, no. 2, pp. 19:1–19:13, Feb. 2016. [Online]. Available: http://doi.acm.org/10.1145/2887740

[25] 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

[26] 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.

[27] K. Pfeiffer, A. Escande, and A. Kheddar, "Singularity resolution in equality and inequality constrained hierarchical task-space control by adaptive nonlinear least squares," *IEEE Robotics and Automation Letters*, vol. 3, no. 4, pp. 3630–3637, Oct 2018.

[28] 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.

[29] R. Fletcher, "A new approach to variable metric algorithms," *The Computer Journal*, vol. 13, no. 3, pp. 317–322, 01 1970. [Online]. Available: https://doi.org/10.1093/comjnl/13.3.317

[30] D. Goldfarb, "A family of variable-metric methods derived by variational means," *Mathematics of Computation*, vol. 24, no. 109, pp. 23–26, 1970. [Online]. Available: http://www.jstor.org/stable/2004873

[31] D. F. Shanno, "Conditioning of quasi-newton methods for function minimization," *Mathematics of Computation*, vol. 24, no. 111, pp. 647–656, 1970. [Online]. Available: http://www.jstor.org/stable/2004840

[32] D. N. Nenchev, Y. Tsumaki, and M. Uchiyama, "Singularity-consistent parameterization of robot motion and control," *The International Journal of Robotics Research*, vol. 19, no. 2, pp. 159–182, 2000. [Online]. Available: https://doi.org/10.1177/02783640022066806

[33] F. Caccavale, S. Chiaverini, and B. Siciliano, "Second-order kinematic control of robot manipulators with jacobian damped least-squares inverse: theory and experiments," *IEEE/ASME Transactions on Mechatronics*, vol. 2, no. 3, pp. 188–194, Sep. 1997.

[34] J. Wang, Y. Li, and X. Zhao, "Inverse kinematics and control of a 7-dof redundant manipulator based on the closed-loop algorithm," *International Journal of Advanced Robotic Systems*, vol. 7, no. 4, p. 37, 2010. [Online]. Available: https://doi.org/10.5772/10495

[35] B. Siciliano and O. Khatib, *Springer Handbook of Robotics*. Berlin, Heidelberg: Springer-Verlag, 2007.

[36] M. de Lasa, I. Mordatch, and A. Hertzmann, "Feature-based locomotion controllers," *ACM Transactions on Graphics*, vol. 29, no. 4, p. 1, 2010.

[37] L. Saab, O. E. Ramos, F. Keith, N. Mansard, P. Soueres, 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.

[38] F. Flacco and A. De Luca, "Discrete-time velocity control of redundant robots with acceleration/torque optimization properties," in *IEEE International Conference on Robotics and Automation (ICRA)*, May 2014, pp. 5139–5144.

[39] F. Udwadia and R. E. Kalaba, "A new perspective on constrained motion," *Proceedings of The Royal Society A: Mathematical, Physical and Engineering Sciences*, vol. 439, pp. 407–410, 11 1992.

[40] F. Udwadia and A. Schutte, "Equations of motion for general constrained systems in lagrangian mechanics," *Acta Mech*, vol. 213, 08 2010.

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

[42] K. Erleben and S. Andrews, "Inverse kinematics problems with exact hessian matrices," 11 2017, pp. 1–6.

[43] N. J. Higham, "Computing a nearest symmetric positive semidefinite matrix," *Linear Algebra and its Applications*, vol. 103, pp. 103 – 118, 1988. [Online]. Available: http://www.sciencedirect.com/science/article/pii/0024379588902236

[44] N. Higham, "Computing the polar decomposition – with applications," *SIAM Journal on Scientific and Statistical Computing*, vol. 7, no. 4, pp. 1160–1174, 1986. [Online]. Available: https://doi.org/10.1137/0907079

[45] A. Conn, N. Gould, and P. L Toint, "Testing a class of methods for solving minimization problems with simple bounds on the variables," *Mathematics of Computation*, vol. 50, pp. 399–430, 04 1988.

[46] A. R. Conn, N. I. M. Gould, and P. L. Toint, "Convergence of quasi-newton matrices generated by the symmetric rank one update," *Mathematical Programming*, vol. 50, no. 1-3, pp. 177–195, 1991.

[47] H. Khalfan, R. Byrd, and R. Schnabel, "A theoretical and experimental study of the symmetric rank-one update," *SIAM Journal on Optimization*, vol. 3, no. 1, pp. 1–24, 1993. [Online]. Available: https://doi.org/10.1137/0803001

[48] B. Faverjon and P. Tournassoud, "A local based approach for path planning of manipulators with a high number of degrees of freedom," INRIA, Tech. Rep. RR-0621, Feb. 1987. [Online]. Available: https://hal.inria.fr/inria-00075933

[49] P. E. E. Gill, S. J. Hammarling, W. Murray, M. A. Saunders, and M. H. Wright, "User's guide for lssol (version 1.0): a fortran package for constrained linear least-squares and convex quadratic programming," Standford University, Standord, California 94305, Tech. Rep. 86-1, January 1986.

[50] C. Collette, A. Micaelli, C. Andriot, and P. Lemerle, "Dynamic balance control of humanoids for multiple grasps and non coplanar frictional contacts," in *IEEE/RAS International Conference on Humanoid Robots*, Pittsburgh, PA, November 29 - December 1 2007, pp. 81–88.

[51] K. Bouyarmane and A. Kheddar, "Using a multi-objective controller to synthesize simulated humanoid robot motion with changing contact configurations," *IEEE International Conference on Intelligent Robots and Systems*, pp. 4414–4419, 2011.

[52] S. Feng, X. Xinjilefu, W. Huang, and C. G. Atkeson, "3d walking based on online optimization," in *IEEE-RAS International Conference on Humanoid Robots (Humanoids)*, Oct 2013, pp. 21–27.

[53] S. Kuindersma, F. Permenter, and R. Tedrake, "An efficiently solvable quadratic program for stabilizing dynamic locomotion," in *IEEE International Conference on Robotics and Automation (ICRA)*, May 2014, pp. 2589–2594.

[54] K. Pfeiffer, A. Escande, and A. Kheddar, "Nut fastening with a humanoid robot," in *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, Sep. 2017, pp. 6142–6148.

[55] M. Nitsche, T. Krajník, P. Čížek, M. Mejail, and T. Duckett, "WhyCon: an efficient, marker-based localization system," in *IEEE/RAS IROS Workshop on Open Source Aerial Robotics*, 2015.