The hierarchical Newton's method for numerically stable prioritized dynamic control

Kai Pfeiffer, Adrien Escande, Pierre Gergondet, Abderrahmane Kheddar, *Fellow, IEEE*

***Abstract*—This work links optimization approaches from hierarchical least-squares programming to instantaneous prioritized whole-body robot control. Concretely, we formulate the hierarchical Newton's method which solves prioritized non-linear least-squares problems in a numerically stable fashion even in the presence of kinematic and algorithmic singularities of the approximated kinematic constraints. These results are then transferred to control problems which exhibit the additional variability of time. This is necessary in order to formulate acceleration based controllers and to incorporate the second order dynamics. However, we show that the Newton's method without complicated adaptations is not appropriate in the acceleration domain. We therefore formulate a velocity based controller which exhibits second order proportional derivative convergence characteristics. Our developments are verified in toy robot control scenarios as well as in complex robot experiments which stress the importance of prioritized control and its singularity resolution.**

***Index Terms*—Constrained control, multibody dynamics, optimization algorithms, robot control**

I. Introduction

CONTROL hierarchies formulated as optimization problems can be solved very efficiently, see e.g., [1]. However, the evaluation of this solver was confined to simulations or well-calibrated experiments due to the lack of appropriate methods to resolve singularities in hierarchies. If unresolved, they lead to unpredictable instabilities and risky behaviors [2]. With this paper, and in the vein of [1], we aim at solving this drawback from an optimization based perspective.

Hierarchical inverse dynamics problems have been approached for the longest time by projector based methods [3]. While they provide a very intuitive insight into the physics of the problem, they lack the powerful tools of optimization. This stretches from the incorporation of inequality constraints to fast solver formulations. While some extensions for example for inequality constraints have been proposed [4], [5], only with the introduction of optimization based approaches such problems could be overcome in a holistic fashion [6], [7]. Nowadays, optimization based control is widespread in robotics and has been applied in numerous works [8]–[11]. Especially the implementation of quadratic programming (QP) based solvers acted as a catalyst for this process. In classical constrained QP based control approaches, control 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. [12]. With the rise of new hierarchical solvers [1], [13], [14] 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, physical stability constraints and objective tasks. Switches in the hierarchical ordering of tasks can be smoothly achieved as for example proposed in [15].

Problems arise if tasks on different levels of the hierarchy get into conflict. Such algorithmic singularities need to be resolved alongside kinematic singularities. Otherwise the near rank deficiency of the Jacobian of the task linearizations –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 quantified with the manipulability measure [16] which can be maximized for singularity avoidance in an optimization setting [17]. Another measure is the singularity index [18] which has been proposed in the context of gimbal attitude control of spacecraft. Analytical robot workspace analysis [19]–[21] enables the prediction and avoidance of kinematic singularities. In contrast, algorithmic singularities depend on the conflict with higher priority tasks at the current robot configuration and therefore are harder to analyze [22]. Furthermore, hierarchical solvers are intended to be employed on humanoid robots in any industrial setting, e.g. aircraft manufacturing [23] or construction [24]. Here end-users are requested to specify a set of usual tasks (set-point, tip force control, trajectory tracking...) [6], [10] under various predefined constraints (joint limits...) or constraints updated from online sensor readings especially when concerned with the physical stability of the robot [25]–[27]. Potential issues might arise if for example sensor readings give targets that are outside of the feasible workspace of the robot. Here engineers without deeper understanding of possible task conflicts and the resulting singularities should be able to set up robot problems safely, quickly and easily.

It is in the vein of optimization that singularity resolution methods like 'damping' in the projector based approach [2], [28]–[31] can be interpreted as their equivalent in optimization, here the Levenberg-Marquardt (LM) algorithm. At the same time, the LM algorithm can be interpreted as approximating the second order derivatives of the Taylor expansion of the quadratic task error norm [32], [33] as a weighted identity

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

K. Pfeiffer was with the CNRS-AIST Joint Robotics Laboratory (JRL), IRL, Tsukuba, Japan. He is now with the Nanyang Technological University, Singapore.

A. Escande, P. Gergondet and A. Kheddar are with the CNRS-AIST Joint Robotics Laboratory (JRL), UMI3218/RL, Tsukuba, Japan.

A. Kheddar is also with the CNRS-University of Montpellier, LIRMM, UMR5506, Interactive Digital Human, France.)

matrix. In our previous work [34] we showed that using an approximation of the true second order derivatives by the Broyden, Fletcher, Goldfarb and Shanno (BFGS) method [35] yields better convergence. Additionally, the tuning of the damping parameter is not straightforward [36].

The previously presented Quasi-Newton method [34] was only aimed at resolving singularities in hierarchical kinematics based control problems. While kinematic control of robots can be sufficient for fixed base robots [37], 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 the robot can aim to maintain a physically stable posture [38]. Therefore, we present ways to include the equation of motion and dynamics constraints into our scheme to generate physically feasible motions while borrowing approaches from numerical optimization.

This work then presents the following new contributions:

- We formulate the Newton's method for prioritized non-linear least-squares problems with a suitable formulation of the hierarchical Hessian (see sec. III).
- We adapt Newton's method, which is a tool from optimization, to constrained prioritized 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 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 in complex hierarchical control scenarios (see sec. VI).

A symbolic overview of this work is given in fig. 1. The corresponding nomenclature is summarized in sec. II.

II. Preliminaries

We present hereafter the nomenclatures and variables naming that are used all along the paper.

A. Kinematic control

In kinematic control we want to find a minimizer to a given non-linear kinematic error $e_2(q(t)) \in \mathbb{R}^{m_2}$ while not violating the constraint $e_1(q(t)) \in \mathbb{R}^{m_1} \leqq 0$

$$\begin{aligned} \min_{q(t)} \quad & e_2(q(t)) \coloneqq f_{2,d}(t) - f_2(q(t)) \\ \text{s.t.} \quad & e_1(q(t)) \leqq 0 \end{aligned} \tag{1}$$

The indices 2 and 1 symbolize the prioritization between the objectives and the constraints respectively. Both equalities and inequalities are encapsulated in the symbol $\leqq$. The vector $q(t) \in \mathbb{R}^n$ (n: number of joints) represents the kinematic configuration of the robot's joints and free-flyer base if any ($\in \mathrm{SE}(3)$, n_f: number of free-flyer joints). It is continuous in time t. $f : \mathbb{R}^n \to \mathbb{R}^m$ is a kinematic function with sufficient continuity properties representing the non-linear kinematics of the robot. m is the task dimension. $f_d(t) \in \mathbb{R}^m$ is the desired value. We define the Jacobian $J(q(t)) \coloneqq \nabla_{q(t)} f(q(t)) \in \mathbb{R}^{m \times n}$ so $\nabla_{q(t)} e(q(t)) = -J(q(t))$.

Non-linear optimization methods with global convergence properties have been proposed in order to find a minimizer to non-linear optimization problems (NLP) like (1) for some given norm [40], [41]. However, these methods are usually computationally heavy and cannot be applied in real-time control. Instead, one can iteratively (i.e. in every consecutive discrete control step k at time t_k) linearize the non-linear kinematic error in time t around the current point t_k with $\{q_k, \dot{q}_k, \ddot{q}_k\}$ and solve an easier since linear problem. This can be factually achieved by the second-order time derivative of $e(q(t))$ at time t_k which allows us to define motion controllers linear in the joint accelerations $\ddot{q}_k$

$$\ddot{e}^{\text{ctrl}}_{\text{PD},k} = -J_k \ddot{q}_k - \dot{J}_k \dot{q}_k \tag{2}$$

Thereby, $\ddot{e}^{\text{ctrl}}_{\text{PD}}$ (ctrl: control) is a PD controller of the form

$$\ddot{e}^{\text{ctrl}}_{\text{PD},k} \coloneqq -k_p e_k - k_v \dot{e}_k \tag{3}$$

with positive proportional and derivative gains k_p and k_v. In the case of velocity based control the first-order derivative suffices with

$$\dot{e}^{\text{ctrl}}_{\text{P},k} = -J_k \dot{q}_k \tag{4}$$

and the proportional controller

$$\dot{e}^{\text{ctrl}}_{\text{P},k} \coloneqq -k_p e_k \tag{5}$$

The joint accelerations $\ddot{q}_k$ (or velocities $\dot{q}_k$) resulting from (2) or (4) (for example as a least-squares solution with fixed $\dot{q}_k$ and q_k) consecutively let the robot converge to a local solution of (1). In sections III and IV we further detail how this approach can be linked to optimization and control.

B. Dynamic control

The motions at a time t_k resulting from the kinematic tasks should obey the instantaneous Newton-Euler equations (or forward dynamics (FD))

$$M(q_k)\ddot{q}_k + N(q_k, \dot{q}_k) = S^T \tau_k + J_c^T(q_k)\gamma, \tag{6}$$

in order to be physically feasible, especially with respect to dynamic limits of the robot. This includes limits on the joint torques $\tau \in \mathbb{R}^n$ and the contact forces $\gamma \in \mathbb{R}^{n_c}$ (n_c: number of contacts). $S \in \mathbb{R}^{n_f \times n}$ is a selection matrix to exclude the unactuated free-flyer. $M(q) \in \mathbb{R}^{n \times n}$ is the whole-body inertia matrix. $N(q, \dot{q}) \in \mathbb{R}^n$ combines Coriolis, centrifugal, gravitational and frictional force effects. $J_c \in \mathbb{R}^{n_c \times n}$ is the contact points' Jacobian matrix.

Note that the original non-linear dynamics are thereby linearized by instantiating the equation at a time t_k and keeping q_k and $\dot{q}_k$ fixed in each respective iteration.

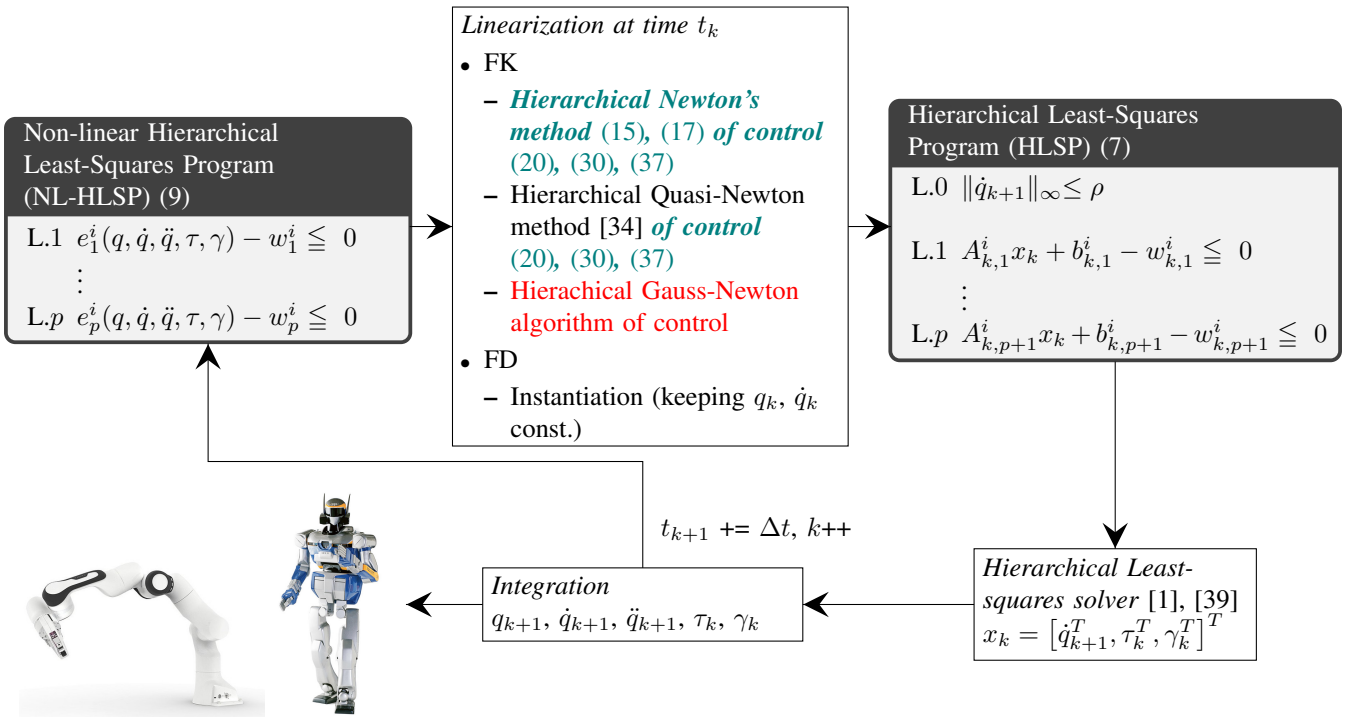

Fig. 1: A symbolic overview of the sequential hierarchical least-squares programming with trust region (S-HLSP) to solve non-linear hierarchical least-squares programs (NL-HLSP) (9) with p levels. This enables real-time robot control for a given set of non-linear forward dynamics (FD) or forward kinematics (FK) tasks. Contributions are marked in teal. Each priority level (l, L.) 1 to p contains m_l non-linear least-squares tasks (the norm notation is omitted for better readability). Each level l is constrained by the tasks of the previous priority levels 1 to $l-1$. First, the NL-HLSP is linearized either by optimization based methods (our contribution, the Hierarchical Newton's method of control (20) for FK) or by instantiation (for FD). Linearization by the Hierarchical Gauss-Newton alogorithm of control leads to numerical instabilities in the case of kinematic and algorithmic singularities (red). The state x_k is given by $x_k \coloneqq \begin{bmatrix}\dot{q}_{k+1}^T & \Delta t \tau_k^T & \Delta t \gamma_k^T\end{bmatrix}^T$ (37). The dependency on q_k, $\dot{q}_k$ of the linearization components $A_k(q_k, \dot{q}_k)$ and $b_k(q_k, \dot{q}_k)$ is omitted for better readability. The resulting (linear) hierarchical least-squares program (HLSP) (including a trust region constraint on $\dot{q}_{k+1}$, the number of priority levels is $p+1$) is then solved for $\dot{q}_{k+1}$, τ_k and γ_k by a hierarchical active-set method. The integrated solution is sent to the robot. Finally, the next iteration of the S-HLSP is instantiated at the current state q_k, $\dot{q}_k$, $\ddot{q}_k$, τ_k and γ_k after incrementation k++.

C. Hierarchical control

We can combine both the linearized kinematic tasks and the equation of motion in a hierarchical least-squares problem with p levels as done in [1], [14], [42], [43]. This boils down to solving a sequence of linear least-squares programs (in the sense of sequential hierarchical least-squares programming (S-HLSP) and assuming that we want to find a local minimizer of the task error (1) in the 2-norm) for $l = 1 \cdots p$

$$\begin{aligned} \min_{x_k, w_{k,l}} \quad & \frac{1}{2}\|w_{k,l}\|^2 \qquad\qquad l = 1 \cdots p \\ \text{s.t.} \quad & A_{k,l} x + b_{k,l} \leqq w_{k,l} \\ & \underline{A}_{k,l-1} x + \underline{b}_{k,l-1} \leqq \underline{w}^*_{k,l-1} \end{aligned} \tag{7}$$

This results in a new instantaneous robot state

$$x_k = \begin{bmatrix}\ddot{q}_k^T & \tau_k^T & \gamma_k^T\end{bmatrix}^T, \tag{8}$$

at a given discrete control time step t_k. A and b are determined by the linearizations of the equation of motion (6) or the kinematic tasks (2). Priorities can be conveniently chosen such that real-world constraints are respected without compromises as is the case for weight-based constrained optimization ($p = 2$) [10], [11]. $w_i \in \mathbb{R}^m$ is a slack variable which relaxes infeasible objectives for example due to task conflict. $\underline{w}^*_{l-1}$ are the optimal values obtained from solving the problems for $i < l$.

D. Kinematic and algorithmic singularities

As it is, the above hierarchy (7) cannot intrinsically deal with kinematic singularities of the Jacobians $A \coloneqq J$ in (2). More, objectives on different priority levels might conflict with each other resulting in algorithmic singularities. Therefore, appropriate singularity resolution methods have to be considered in order to prevent unstable robot behavior due to numerics.

III. The Hierarchical Newton's Method

As has been proposed by [33], Newton's method can be used to solve non-linear inverse kinematics problems (1) in a least-squares sense. This maintains critical second-order information in the vicinity of singularities of the Jacobian of the kinematic error (1) (see bellow). A similar reasoning is applicable for prioritized inverse kinematic control as observed in [34], where

only a hierarchical Quasi-Newton method was proposed. Now, we derive the hierarchical Newton's method. We start with the following non-linear hierarchical least-squares problem (NL-HLSP, the time dependency t is omitted to ease readability)

$$\begin{aligned} \min_{q,w_l} \quad & \frac{1}{2}\left\|w_l\right\|^2 \qquad\qquad l = 1 \cdots p \\ \text{s.t.} \quad & e_l(q) \leqq w_l \\ & \underline{e}_{l-1}(q) \leqq \underline{w}^*_{l-1} \end{aligned} \tag{9}$$

The goal is to minimize the constraint violation w_l of each level l 'at best' (in a least-squares sense). Already obtained optimal violations of previous levels $\underline{w}^*_{l-1}$ must stay unchanged.

In the sense of the active-set method, we formulate the Lagrangian of an equality only problem of (9) at level l

$$\mathcal{L}_l = \frac{1}{2} w_l^T w_l + \lambda_{l,l}^T (w_l - e_l) + \underline{\lambda}_{l-1,l}^T (\underline{w}^*_{l-1} - \underline{e}_{l-1}) \tag{10}$$

Only active constraints with $e_l \geq 0$ and $\underline{e}_{l-1} = \underline{w}^*_{l-1}$ are considered. $\lambda_{l,l}$ and $\underline{\lambda}_{l-1,l}$ are the Lagrange multipliers associated with the active constraints of level l and the previous levels $1, \ldots, l-1$, respectively.

The non-linear first order optimality conditions are

$$\begin{aligned} \nabla_{q,w_l,\lambda_{l,l},\underline{\lambda}_{l-1,l}} \mathcal{L}_l &= K_l(q, w_l, \lambda_{l,l}, \underline{\lambda}_{l-1,l}) \\ &= \begin{bmatrix} J_l^T \lambda_{l,l} + \underline{J}_{l-1}^T \underline{\lambda}_{l-1,l} \\ w_l + \lambda_{l,l} \\ w_l - e_l \\ \underline{w}^*_{l-1} - \underline{e}_{l-1} \end{bmatrix} = 0 \end{aligned} \tag{11}$$

We perturb the KKT system (not in time t but only in configuration space q, see Sec. IV) leading to the Newton step

$$K_l(x_k + \Delta x_k) = K_l(x_k) + \nabla K_l(x_k) \Delta x_k = 0 \tag{12}$$

Here, the index k indicates the current iteration of the Newton's method. The variable vector x (and its increment Δx) is given by

$$x^T = \begin{bmatrix} q^T & w_l^T & \lambda_{l,l}^T & \underline{\lambda}_{l-1,l}^T \end{bmatrix} \tag{13}$$

The Lagrangian Hessian is

$$\nabla K_l(x) = \begin{bmatrix} \hat{H}_l & 0 & J_l^T & \underline{J}_{l-1}^T \\ 0 & I & I & 0 \\ J_l & I & 0 & 0 \\ \underline{J}_{l-1} & 0 & 0 & 0 \end{bmatrix} \tag{14}$$

We refer to the expression

$$\nabla_q^2 \mathcal{L}_l = \hat{H}_l = \sum_{d=1}^{m_l} \lambda_{l,l,d} H_{l,d} + \sum_{i=1}^{l-1} \sum_{d=1}^{m_i} \lambda_{i,l,d} H_{i,d} \tag{15}$$

as the hierarchical Hessian. $H_l \coloneqq \nabla_q^2 f_l$ are the Hessians of the respective kinematics $f_l(q)$.

Equation (12) is also the optimality condition of the HLSP

$$\begin{aligned} \min_{\Delta q_k, w_{k,l}} \quad & \frac{1}{2}\left\|w_{k,l}\right\|^2 + \frac{1}{2} \Delta q_k^T \hat{H}_{k,l} \Delta q_k \\ s.t. \quad & e_{k,l} + J_{k,l} \Delta q_k = w_{k,l} \\ & \underline{e}_{k,l-1} + \underline{J}_{k,l-1} \Delta q_k = \underline{w}^*_{k,l-1} \end{aligned} \tag{16}$$

which we refer to as the hierarchical Newton's method in combination with the active-set method. As we repeatedly execute the above steps at the current iterate q_k, a non-linear hierarchical least-squares programming is turned into a linear one and solved until convergence; we can also refer to this method as sequential hierarchical least-squares programming (S-HLSP). If $\hat{H}_l$ is positive definite, its Cholesky decomposition $\hat{H}_l = R_l^T R_l$ exists and we get the least-squares program

$$\begin{aligned} \min_{\Delta q_k} \quad & \frac{1}{2} \left\| \begin{bmatrix} J_{k,l} \\ R_{k,l} \end{bmatrix} \Delta q_k - \begin{bmatrix} e_{k,l} \\ 0 \end{bmatrix} \right\|_2^2 \qquad l = 1 \cdots p \\ \text{s.t.} \quad & \underline{e}_{k,l-1} - \underline{J}_{k,l-1} \Delta q_k = \underline{w}^*_{k,l-1} \end{aligned} \tag{17}$$

If $\hat{H}_l$ is neglected, we obtain the GN algorithm. If the Jacobian J_l is rank deficient, the problem is ill-posed and results in a numerically unstable solution Δq_k. If $\hat{H}_l = \mu^2 I$ is chosen as an identity matrix with weight μ, we get the LM algorithm. If $\hat{H}_l$ is approximated by the BFGS algorithm [34] using the gradient $\nabla_q \mathcal{L}$ in (11), we get a Quasi-Newton method.

The hierarchical Newton's method is in the form of the hierarchical least-squares program formulated in (7) (with w_l given implicitly). Efficient solvers to (17) based on the active-set method are described in e.g. [1] or [39].

Inequality constraints are incorporated by means of active and inactive constraints and due to the problem formulation with slack variables, see [13]. Above first order optimality conditions extend to the Karush-Kuhn-Tucker conditions. Inactive constraints i thereby result in $w_i = 0$, $\lambda_{j,i} = 0$, not further influencing the Hessian $\hat{H}_j$ on some level $j \geq i$.

For the hierarchical Hessian calculation $\hat{H}_l$ of level l (15) we need to calculate the second order derivatives

$$H = \nabla_q^2 f(q) \tag{18}$$

of the kinematic functions $f(q)$ for all levels 1 to l. For this we follow [44] with computational complexity of $\mathcal{O}(n^2)$.

Since the hierarchical Hessian $\hat{H}$ can become indefinite, the Cholesky decomposition for obtaining the factor R can not be applied. We use the symmetric Schur decomposition ($O(9n^3)$ [45]) to obtain the spectral decomposition $\hat{H} = QUQ^T$ with $U = \operatorname{diag}(\lambda(\hat{H}))$. Negative eigenvalues in U are then replaced by a small positive threshold such that $R = \sqrt{U} Q^T$ and convexity of the optimization problem is maintained. This method proves to be faster than other regularization methods like [46] which is based on the (asymmetric) SVD decomposition ($O(12n^3)$ [45] plus an additional Cholesky decomposition $O(n^3/3)$ in order to obtain R). A positive definite approximation of the Hessian can also be obtained by the BFGS algorithm as detailed in [34].

IV. From optimization to kinematic control

In Section III we introduced the Newton's method of constrained optimization to bring a non-linear kinematic error to zero. The Newton's method corresponds to a quadratic Taylor approximation [33] around q that is valid within a neighborhood (called trust region) of it, and allows for a bounded step Δq. Assuming a zero order holder with a control time step of $\Delta t = 1$ s; the change of joint configuration is $\Delta q = \Delta t \dot{q} = \dot{q}$. This allows us to make a trivial connection between optimization, aiming to make a step Δq towards the optimum (note how the KKT system is not disturbed in time

t in the derivation of the hierarchical Newton's method (12)), and control, aiming to determine the next robot state triple $\{q_k, \dot{q}_k, \ddot{q}_k\}$. It is also noteworthy that in this case the linearization in time t as done in (2) corresponds to the Gauss-Newton algorithm and its subsequent numerical instability at kinematic singularities.

However, usually robots are controlled at much higher rates, that is $\Delta t \ll 1$ s. Yet, the time step Δt connects the two entities of 'optimization' (optim) and 'control' (assuming a simple proportional controller $\dot{e}^{\text{ctrl}}_{\text{P},k}$ for now)

$$w^{\text{optim}}_k := J_k \Delta q_k + \Delta t \dot{e}^{\text{ctrl}}_{\text{P},k} = \Delta t (J_k \dot{q}_k + \dot{e}^{\text{ctrl}}_{\text{P},k}) =: \Delta t w^{\text{ctrl}}_k \tag{19}$$

That is, we calculate a new velocity $\dot{q}_k$ but only make a step $\Delta q_k = \Delta t \dot{q}_k$ towards the optimum. Consequently, the model needs to be updated with the Hessian of the Lagrangian (10) using w^{optim}_k and λ^{optim}_k. Since solving the constrained control problems (20) yields w^{ctrl}_k and λ^{ctrl}_k, a scaling of the form $w^{\text{optim}}_k = \Delta t w^{\text{ctrl}}_k$ according to (19) is required. Due to the linear dependency between the slack w and the Lagrange multipliers λ [39] we further get $\lambda^{\text{optim}}_k = \Delta t \lambda^{\text{ctrl}}_k$. In what follows, we write $w = w^{\text{optim}}$ and $\lambda = \lambda^{\text{optim}}_k$.

With these considerations in mind we can directly derive the hierarchical Newton's method of control from (17) (with k now again representing the time t_k and not the k-th Newton iteration)

$$\begin{aligned} \min_{\dot{q}_k} \quad & \frac{1}{2} \left\| \begin{bmatrix} J_{k,l} \\ R_{k,l} \end{bmatrix} \dot{q}_k + \begin{bmatrix} \dot{e}^{\text{ctrl}}_{\text{P},k,l} \\ 0 \end{bmatrix} \right\|^2_2 \qquad \text{for } l = 1 \cdots p \\ \text{s.t.} \quad & -\underline{\dot{e}}^{\text{ctrl}}_{k,l-1} - \underline{J}_{k,l-1} \dot{q}_k \leqq \underline{w}^*_{k,l-1} \end{aligned} \tag{20}$$

Neglecting the second order information $\hat{H}_l$ (or its Cholesky factor R_l thereof) yields the hierarchical GN algorithm of control. If the Jacobian J_l is rank deficient the problem is ill-posed and results in a numerically unstable solution $\dot{q}_k$. If $\hat{H}_l = \mu^2 I$, I being the identity matrix with weight μ, we get the LM algorithm. If $\hat{H}_l$ is approximated by the BFGS algorithm [34], by using the gradient $\nabla_q \mathcal{L}$ in (11), the slacks w^{ctrl} and the Lagrange multipliers λ^{ctrl}, we get the Quasi-Newton method of control. The trust region constraint is converted into a limit on the joint velocities. 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 [32].

As described in [34] we switch between the GN algorithm and the Newton's method judging upon the residual of the GN algorithm

$$\frac{1}{2} \| J_{k,l} \dot{q}_k + \dot{e}^{\text{ctrl}}_{\text{P},k,l} \|^2_2 < \nu \tag{21}$$

By doing so, the joints are 'freed' from the full rank second order augmentation whenever the linearization represents the non-linear original task function sufficiently enough. This way we ensure the best possible convergence of lower priority levels. Note that the threshold $\nu = 10^{-12} \Delta t^2$ is now dependent of the time step Δt in accordance with (19).

V. Dynamically feasible kinematic control

In previous Sec. IV we formulated the GN algorithm and Newton's method of control only in the velocity domain. In this section, we extend our approach to equation of motion (6) of second order (dynamics). In Sec. V-A we argue why Newton's method cannot be straightforwardly extended to the acceleration domain. Note that this also concerns regularization of acceleration based tasks as is commonly done in robotics (albeit with a small weight), see for example [10], [14]. Therefore, our idea for acceleration-based control is to change the controller $\dot{e}^{\text{ctrl}}_{\text{P}}$ from a linear proportional controller to some controller $\dot{e}^{\text{ctrl}}_{\text{PD}}$ that emulates PD control $\ddot{e}^{\text{ctrl}}_{\text{PD}}$ in the velocity domain. In Sec. V-B we show how this can be achieved and applied to the equation of motion (see Sec. V-C).

A. Damping in acceleration-based control

In the following we show that the instantaneous acceleration based kinematic optimization at a discrete time t_k

$$\min_{\ddot{q}_k} \quad \frac{1}{2} \left\| \begin{bmatrix} J_k \\ R_k \end{bmatrix} \ddot{q}_k + \begin{bmatrix} \dot{J}_k \dot{q}_k + \ddot{e}^{\text{ctrl}}_{\text{PD},k} \\ 0 \end{bmatrix} \right\|^2_2 \tag{22}$$

leads to low frequency oscillations over time t. Solving (22) using the pseudo-inverse to obtain the instantaneous joint accelerations and using Euler integration to obtain the new joint velocities at time $t_{k+1} = t_k + \Delta t$ we get (assuming full rank of $\begin{bmatrix} J_k^T & R_k^T \end{bmatrix}^T$)

$$\begin{aligned} \dot{q}_{k+1} = {} & (J_k^T J_k + R_k^T R_k)^{-1} \\ & ((R_k^T R_k + (1 - \Delta t k_v) J_k^T J_k) \dot{q}_k - \Delta t J_k^T (k_p e_k + \dot{J}_k \dot{q}_k)) \end{aligned} \tag{23}$$

$$q_{k+1} = q_k + \Delta t \dot{q}_k \tag{24}$$

It can be observed that the positive definite augmentation $R^T R$ reduces the negative definite term $-\Delta t k_v J^T J$ (which represents the derivative term of the PD controller) and therefore leads to an under-damped system with characteristic overshooting behaviour. Damping in the form of $R = \mu I$ is commonly applied in robotics for regularization purposes [10], [14]. The weight μ is usually chosen small such that the oscillations are small in amplitude and do not negatively influence practical task achievement. From a theoretical point of view this still might be undesirable.

In order to illustrate the above derivations we assume a point mass with a continuous 1D translational degree of freedom (DoF) $q(t)$ and $f(t) = q(t)$. The desired position is $q_d = 0$. The task error is then $e(t) = q_d - q(t) = -q(t)$. The Jacobian of this robot is $J = \frac{df}{dq} = \frac{dq}{dq} = 1$, the time derivative of the Jacobian is $\dot{J} = 0$.

The acceleration based controller (2) becomes

$$\ddot{q}(t) + k_v \dot{q}(t) + k_p q(t) = 0 \tag{25}$$

with the solution

$$q(t) = \mathrm{e}^{-\delta t} (C \cos(w_d t) + D \sin(w_d t)) \tag{26}$$

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

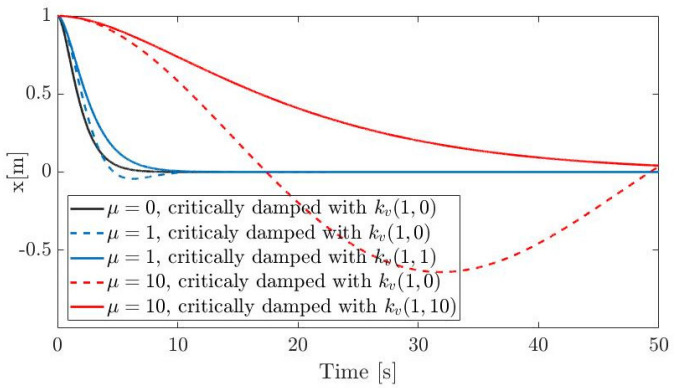

Fig. 2: Equation (26) plotted for $m = 1$, $k_p = 1$, different μ and $k_v(k_p) = 2\sqrt{mk_p}$ (black line and dashed lines) or $k_v(k_p, \mu) = 2\sqrt{m(1+\mu^2)k_p}$.

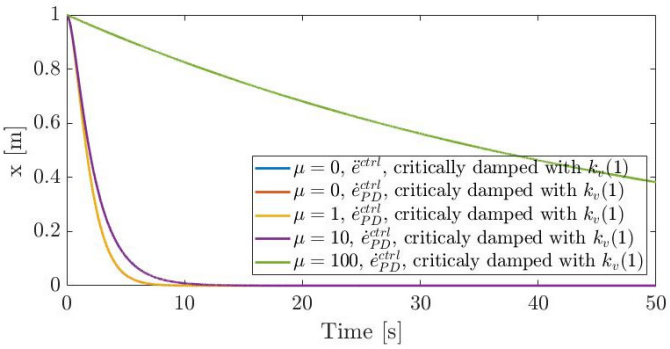

Fig. 3: Plots of convergence of the 1-D mass governed by (34). The larger μ the slower the robot behaves. Simultaneously, the difference to the control of the undamped ($\mu = 0$) acceleration based PD controller $\ddot{e}^{\text{ctrl}}$ increases. However, exponential convergence is always achieved.

We now apply a damping term μ to the system (25)

$$\begin{bmatrix}1\\ \mu\end{bmatrix}\ddot{q}(t) + \begin{bmatrix}k_v\dot{q}(t) + k_p q(t)\\ 0\end{bmatrix} = \begin{bmatrix}0\\ 0\end{bmatrix} \tag{27}$$

We can convert this system into a homogenous ODE with the same solution as in (26) by applying the pseudo-inverse

$$\ddot{q}(t) + \frac{k_v}{1+\mu^2}\dot{q}(t) + \frac{k_p}{1+\mu^2}q(t) = 0 \tag{28}$$

The influence of the damping μ on the critically damped task gains k_p and k_v is clearly exposed. Critical damping can be achieved with

$$k_v(k_p, \mu) = 2\sqrt{m(1+\mu^2)k_p} \tag{29}$$

Some convergence curves for $q(0) = 1$, $\dot{q}(0) = 0$, $\Delta t = 5$ ms are plotted in Fig. 2. Note 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 complicated 3D robot with more and especially coupled DoF's, and a varying $R(q, \mu)$, it seems cumbersome to find the expression for critical damping $k_v(k_p, R)$ such that overshooting behavior can be prevented. Therefore, we favour to shift the whole problem into the velocity domain and emulate acceleration-based control by formulating an appropriate controller.

B. Acceleration control expressed in the velocity domain

We propose a controller $\dot{e}^{\text{ctrl}}_{\text{PD}}$ that is able to emulate acceleration based PD control in the velocity domain

$$\dot{e}^{\text{ctrl}}_{\text{PD},k} \coloneqq -\dot{e}_k - \Delta t(\ddot{e}^{\text{ctrl}}_{\text{PD},k} + \dot{J}_k\dot{q}_k) \tag{30}$$

If we use this controller in the unconstrained Newton's method of control (20), by application of the pseudo-inverse we get (note that $\dot{e}^{\text{ctrl}}_{\text{PD},k}$ already incorporates the Euler integration $\dot{q}_{k+1} = \dot{q}_k + \Delta t\ddot{q}_k$; the new decision variable in (20) is therefore $\dot{q}_{k+1}$ instead of $\dot{q}_k$)

$$\begin{aligned}\dot{q}_{k+1} &= (J_k^T J_k + R_k^T R_k)^{-1} \\ &\quad ((1-\Delta t k_v)J_k^T J_k\dot{q}_k - \Delta t J_k^T(k_p e_k + \dot{J}_k\dot{q}_k))\end{aligned} \tag{31}$$

$$q_{k+1} = q_k + \Delta t\dot{q}_k \tag{32}$$

In contrast to (24) we can now see that the derivative term $-\Delta t k_v J_k^T J_k$ is not influenced by the augmentation $R_k^T R_k$. As long as the system is critically damped $k_v(k_p) = 2\sqrt{mk_p}$ and $k_v > 1/\Delta t$ exponential convergence is ensured. If there is no augmentation ($R_k = 0$, GN algorithm) we get the original acceleration based PD control dynamics

$$\dot{q}_{k+1} = \dot{q}_k + \Delta t J_k^+(-k_v J_k\dot{q}_k - k_p e_k - \dot{J}_k\dot{q}_k) \tag{33}$$

$$q_{k+1} = q_k + \Delta t\dot{q}_k \tag{34}$$

It can be seen that the augmented system (31) exhibits slower convergence than the equivalent undamped acceleration based PD controller (33) depending on the magnitude of $R_k^T R_k \geq 0$ since $\|(J_k^T J_k + R_k^T R_k)^{-1} J_k^T J_k\dot{q}_k\| \leq \|\dot{q}_k\|$. However, we can argue that we only use Newton's method when slow joint movements in singular configurations are desirable anyways and the particular type of critically damped PD motions is not further relevant.

To come back to the 1-D robot example from the previous section V-A we get

$$\begin{aligned}\dot{q}_{k+1} &= \frac{1}{1+\mu^2}(1 - \Delta t k_v\dot{q}_k - \Delta t k_p e_k) \\ q_{k+1} &= q_k + \Delta t\dot{q}_k\end{aligned} \tag{35}$$

Some convergence curves for $q(0) = 1$, $\dot{q}(0) = 0$, $\Delta t = 5$ ms are plotted in Fig. 3. With increasing damping μ the robot convergence becomes slower and diverges from the undamped control generated by the acceleration based PD controller $\ddot{e}^{\text{ctrl}}$. Unlike the acceleration based controller $\ddot{e}^{\text{ctrl}}$ however, exponential convergence is achieved by $\dot{e}^{\text{ctrl}}_{\text{PD}}$ for any μ as long as the system is critically damped with $k_v = 2\sqrt{mk_p}$.

At the optimum of each instantaneous control cycle we solve an equality only least-squares problem projected into the null-space basis of the active constraints of the higher priority levels [39]. Therefore the same argumentation as above using the pseudo-inverse (32) holds for the constrained Newton's method of instantaneous control.

The above scheme uses linear integration. This forces to adopt Euler angles to represent 3D rotations such as the robot base orientation. To avoid gimbal lock, we use them as a

local parametrization: a rotation Q is written $Q_0 Q(\theta)$ with Q_0 fixed and $Q(\theta)$ the Euler parametrization. After each control iteration Q_0 is set to $Q_0 Q(\theta)$ and θ is set to 0. We assume small changes of θ between iterations.

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 (6) are replaced by finite differences

$$\ddot{q}_k = \frac{\dot{q}_{k+1} - \dot{q}_k}{\Delta t} \tag{36}$$

such that we get

$$\begin{bmatrix} M(q_k) & -S^T & -J_c^T(q_k) \end{bmatrix} \begin{bmatrix} \dot{q}_{k+1} \\ \Delta t \tau_k \\ \Delta t \gamma_k \end{bmatrix} = M \dot{q}_k - \Delta t N(q_k, \dot{q}_k) \tag{37}$$

For numerical robustness it is desirable to keep the conditioning of the system matrix $\begin{bmatrix} M(q_k) & -S^T & -J_c^T(q_k) \end{bmatrix}$ so we compute $\dot{q}_{k+1}$, $\Delta t \tau_k$ and $\Delta t \gamma_k$.

The equation of motion can be considered full rank if the inertia matrix M is physically consistent and therefore positive definite [47], [48]. This means that the system matrix of the equation of motion $\begin{bmatrix} M(q_k) & -S^T & -J_c^T(q_k) \end{bmatrix}$ is not concerned with kinematic singularities of the contact Jacobians $J_c(q_k)$ due to the full-rank property of $M(q_k)$ ('row rank equals column rank'). Furthermore, the equation of motion is already linear in the accelerations $\ddot{q}_k$ (or velocities $\dot{q}_{k+1}$ in case of the forward integration and only non-linear in $\dot{q}_k$), joint torques τ_k and generalized contact forces γ_k. Hence, a Taylor expansion for the purpose of linearization is not necessary. We therefore do not consider the equation of motion in the Hessian calculation of the lower level linearized constraints.

VI. Validation

Here, we assess our proposed method named LexDynAH (acronym for **Lex**icographic Augmentation for **Dyn**amics with **A**nalytic **H**essian). We aim to validate that we can achieve: (i) agreement between the velocity and acceleration based controllers $\dot{e}_{\text{PD}}^{\text{ctrl}}$ (30) and $\ddot{e}_{\text{PD}}^{\text{ctrl}}$ (3); and (ii) numerical stability in singular robot configurations. First, we conduct three simulations with simple 2D stick robots (sec. VI-A) to confirm our derivations. Then, we apply our method in two real HRP-2Kai robot experiments (sec. VI-B) that is position controlled. HRP-2Kai has 32 actuated DoF and a 6 DoF un-actuated free-flyer. The control frequency is 200 Hz ($\Delta t = 5$ ms). Equation (20) is solved with the hierarchical least-squares warm-started active set solver LexLSI [39]. The trust region limit (Sec. IV) is chosen as 0.1 m or rad with an adaptive heuristic for real-time hierarchical control as in [34].

For comparison, we use the hierarchical Quasi-Newton method presented in [34] (referred to as LexDynBFGS). It was originally designed for optimization problems of the form (17) but including the modifications presented in this paper it is identical with LexDynAH except that the hierarchical Hessian (15) is approximated by the BFGS algorithm. Furthermore, for regularized acceleration based control (2) we use the QP solver LSSOL [49] ($p = 2$). Inequality constraints are only allowed on level 1. 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. On both levels a soft hierarchy can be established by weighting tasks against each other (therefore we call this solver Weighted Least Squares - WLS). Such a control setup is commonly found in the literature [6]–[11].

A. Three simulation toy examples

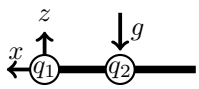

Fig. 4: **Example 1**: initial configuration of the robot.

1) Example 1 – Freely swinging pendulum: consists of a fixed-base planar robot with two links (both having unit length and unit mass) and two revolute joints. The joint torques of both joints are set to zero so that it swings freely. The initial configuration is set to $[-\pi/2, 0]$ rad with zero velocity (see Fig. 4). The motion of the pendulum is determined by solving the instantaneous equation of motion (6) in a least squares sense. The hierarchical problem is described in Fig. 5.

Results of Example 1 show that we can reproduce the behavior of the acceleration equation of motion (6) by the use of the velocity domain based one (37), with identical joint trajectories $q(t)$ (see Fig. 6).

2) Example 2 – In reach end-effector task: steers the robot from the previous Example 1 to reach the desired Cartesian task-point $[1, 1]$ m. The initial robot posture corresponds to an arbitrary non-singular configuration, e.g., $[-\pi, 1]$ rad, see Fig. 7. We define the acceleration- or velocity-based hierarchy in Fig. 8.

The problem is solved by the GN algorithm ((20) with $\hat{H}_l = 0$). The joint positions, joint velocities, joint torques and task error norms are given in Fig. 9. It can be seen that both the velocity and acceleration based control lead to an identical behavior. The converged robot posture is given in Fig. 10.

3) Example 3 – Conflict between linearized task constraint and dynamics constraint: In this simulation we want to test the behavior of the robot if a linearized end-effector task is in conflict with a (non-linearized) dynamics constraint. We use a robot with 3-DoF's. We set its initial configuration to $[-\pi, 1, 0]$ rad, see Fig. 11. The hierarchy is given in Fig. 12. Constraints in bold have to be considered in the calculation of the hierarchical Hessian. Note that for bound constraints we have $H = 0$. The commanded joint 1 torque is set to zero and hence conflicts with the end-effector task of reaching the point $[1, 1]$ m. The trust region constraint is put on the second level in order to not interfere with the equation of motion.

The norm of the distance of the end-effector from the target is given in Fig. 13. As discussed in Sec. V-A, damping in the acceleration domain leads to slowly oscillating behavior that can be observed from the error curve 'acc. based, damping'. The damping value is set to $\mu = 0.1$. The error norm of the other methods gets minimized fairly well but is disturbed by the slow oscillations of the freely swinging joint 1.

Hierarchy for example 1	
L.1	• Equation of motion, acceleration-based (6) or velocity-based (37) • $\tau = 0$
L.2	• Regularization term $\dot{q} = 0$ (vel. based), expressed in accelerations by using finite differences (36) in case of acceleration-based control

Fig. 5: **Example 1**. Hierarchy for swinging pendulum.

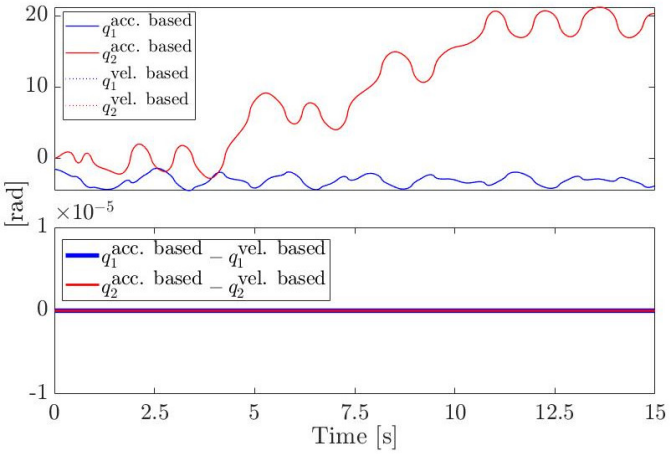

Fig. 6: **Example 1**. The upper graph shows the joint positions of the swinging pendulum for the acceleration-based ('acc. based') equation of motion and the velocity-based ('vel. based') ones (full superposition). The lower graph shows the difference between the values.

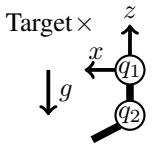

Fig. 7: **Example 2**, initial configuration of the robot. The desired end-effector position is at the cross.

Hierarchy for example 2	
L.1	Equation of motion, acceleration-based or velocity-based
L.2	Joint velocity limit (corresponds to the trust region constraint), expressed with accelerations by using finite differences (36) in the case of acc. based control
L.3	End-effector task with $\ddot{e}^{\text{ctrl}}$ (acc. based) or with $\dot{e}^{\text{ctrl}}_{\text{PD}}$ (vel. based)
L.4	$\dot{q} = 0$ (vel. based), expressed with accelerations by using forward differences (36) in case of acceleration-based control
L.5	$\tau = 0$

Fig. 8: **Example 2**. Control hierarchy.

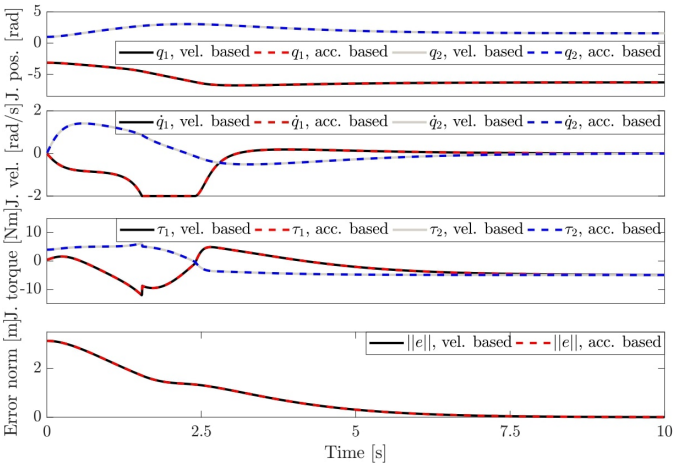

Fig. 9: **Example 2**, joint (J.) positions, velocities, torques and task error norm. They are identical for the acceleration- and velocity-based equation of motion and PD motion controllers $\ddot{e}^{\text{ctrl}}$ and $\dot{e}^{\text{ctrl}}_{\text{PD}}$.

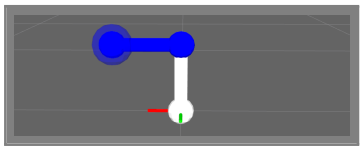

Fig. 10: **Example 2**, converged robot posture.

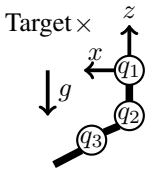

Fig. 11: **Example 3**, initial configuration of the robot. The desired end-effector position is at the cross.

Hierarchy for example 3	
L.1	• Equation of motion, acceleration-based or velocity-based with accelerations integrated by (36) • $\tau_1 = 0$
L.2	Trust region constraint, expressed with accelerations by using forward differences (36) in the case of acc. based control
L.3	**End-effector task with $\ddot{e}^{\text{ctrl}}$ (acc. based) or with $\dot{e}^{\text{ctrl}}_{\text{PD}}$ (vel. based), solved by the GN algorithm, Newton's method (vel. based) or the LM algorithm (acc. based)**
L.4	$\dot{q} = 0$ (vel. based), expressed with accelerations by using forward differences (36) in case of acceleration-based control
L.5	$\tau = 0$

Fig. 12: **Example 3**. Control hierarchy.

Figure 14 shows the joint velocities of the robot. For the GN algorithm (both acc. and vel. based) numerical instabilities can be observed. This is due to the zero commanded joint 1 torque that conflicts with the aim of getting closer to the target

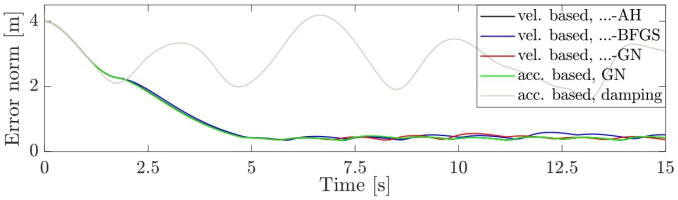

Fig. 13: **Example 3**, task error norm for the end-effector task. Damping in the acceleration domain leads to overshooting behavior and the error is barely minimized (dark gray curve). The velocity-based PD controller $\dot{e}_{\text{PD}}^{\text{ctrl}}$ prevents this behavior.

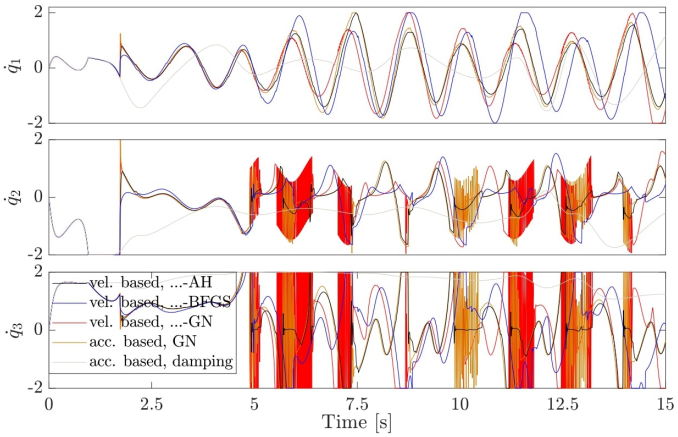

Fig. 14: **Example 3**, joint velocities. The acc. and vel. based GN algorithm leads to numerical instabilities in the joint velocities $\dot{q}_2$ and $\dot{q}_3$. They occur whenever the kinematic subchain consisting of link 2 and link 3 is close to kinematic singularity. LexDynAH (AH) and LexDynBFGS (BFGS) are numerically stable. Damping in the acceleration domain is numerically stable but leads to overshooting behavior.

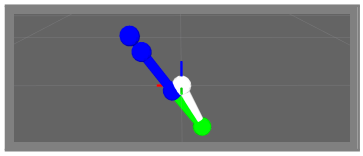

Fig. 15: **Example 3**. The robot managed to reach the target as close as possible (LexDynAH). However, some swaying motion remains due to the freely swinging joint 1 (white ball) with zero commanded torque. Note how the kinematic subchain of the green and blue link is in kinematic singularity.

with the end-effector. In order to minimize the task error as much as possible the kinematic subchain consisting of both links 2 and 3 is forced into kinematic singularity.

Newton's method with second order information from both LexDynAH and LexDynBFGS leads to numerically stable joint velocity behavior without overshooting the end-effector as seen for the damping in acceleration-based control. The robot configuration with low error norm but with remaining some swaying motion is given in Fig. 15.

Hierarchy for LexDynAH and LexDynBFGS

- L.1 • $4n_c$ (n_c: number of contacts) bounds on generalized contact wrenches: $\gamma > 0$
  • 32 joint limits using a velocity damper [50]
- L.2 • 38 integrated equations of motion
  • 38 torque limits
- L.3 38 trust region limits
- L.4 3 **inequality constraints for self-collision avoidance**
- L.5 18 **geometric contact constraints**
- L.6 1 equality constraint on the head yaw joint to put the vision marker into the field of view
- L.7 3 **inequality constraints on the CoM**
- L.8 • 6 **end-effector equality constraints for left and right hand**
  • 3 **equality constraints to keep the chest orientation upright**
- L.9 3 **stricter inequality constraints on the CoM**
- L.10 38 constraints to minimize joint velocities: $\dot{q} = 0$
- L.11 $4n_c$ constraints to minimize the generalized contact wrenches: $\gamma = 0$

Fig. 16: Control hierarchy for HRP-2Kai experiments.

Hierarchy for WLS

- L.1 • $4n_c$ bounds on generalized contact wrenches: $\gamma > 0$
  • 32 joint limits using an acceleration damper [50]
  • 38 equations of motion
  • 38 torque limits
  • 3 inequality constraints for self collision avoidance
  • 18 geometric contact constraints
  • 3 inequality constraints on the CoM
- L.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)
  • $4n_c$ constraints to minimize the generalized contact wrenches: $\gamma = 0$

Fig. 17: Weighted least-squares control hierarchy

B. Experiments with a humanoid robot

Here, we assess our approach with a real humanoid robot HRP-2Kai. We devised a control hierarchy given in Fig. 16. The hierarchical separation of the bound constraints on the generalized contact wrenches and the joint limits ($l = 1$) from the equation of motion ($l = 2$) allows LexLSI to cheaply handle the variable bounds on the first level.

For comparison purpose, we use the acceleration-based weighted control hierarchy given in Fig. 17, and solved by LSSOL. The hierarchy for WLS contains the dynamic constraints (equation of motion and bounds on γ and τ), joint limits, self-collision avoidance, center of mass (CoM) task and contact tasks as constraints on priority level 1. All these tasks have the same priority without weighting. Although the self-collision avoidance, the contact and the CoM constraints

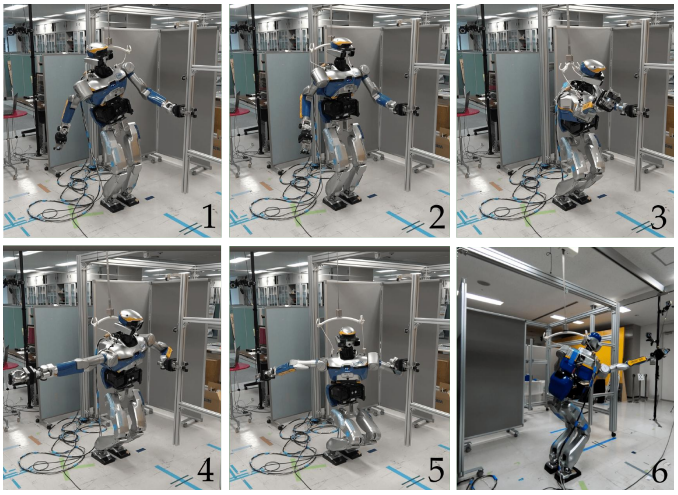

Fig. 18: Pictures of HRP-2Kai performing **exp. 1**, LexDynAH, from left to right: **1.:** The left hand has grabbed the pole while the right hand task on level 9 is in conflict with the CoM task on level 8. **2.:** The CoM on level 8 switched from box 1 to box 2 and is not in conflict with the right hand task anymore. The right hand task on level 9 is not augmented anymore. **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.

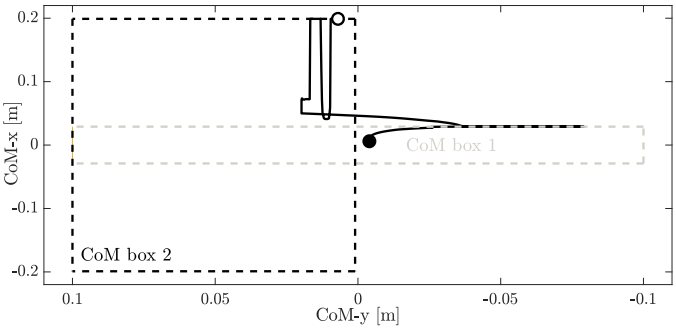

Fig. 19: **Exp. 1**, LexDynAH, CoM movement. The CoM starts at the black dot and moves until it arrives at the white dot, first being constrained in box 1 and then in box 2.

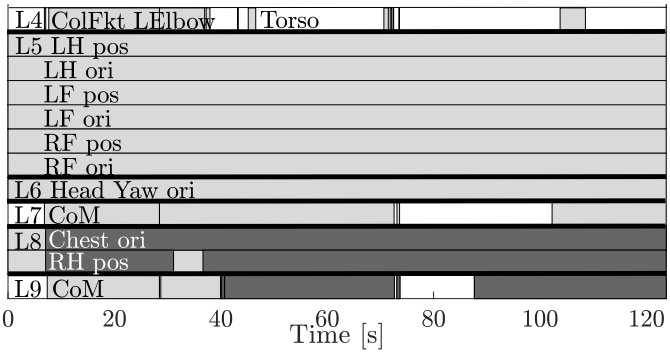

Fig. 20: **Exp. 1**, LexDynAH, map of activity (light gray) and Newton's method (dark gray)

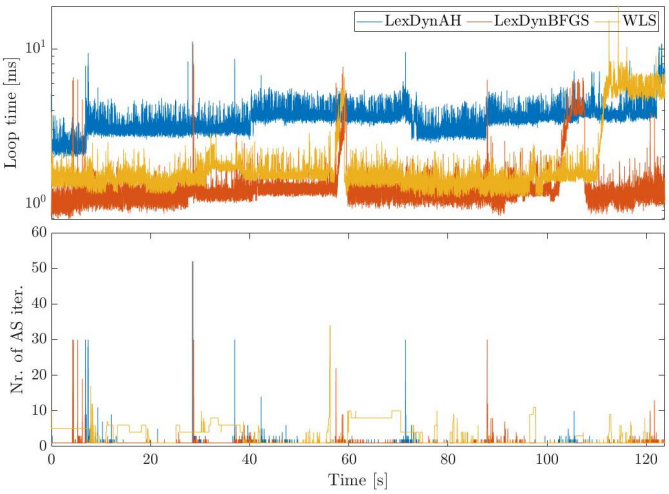

Fig. 21: **Exp. 1**, computation times and active set iterations. LexDynAH peaks at 28 s with 2.63 ms and a maximum number of active set iterations of 52. LexDynBFGS peaks at 28 s with 1.11 ms and a maximum number of active set iterations of 52. WLS peaks at 56 s with 2.94 ms and a maximum number of active set iterations of 34.

can be the source of potential conflict or even (unresolved) kinematic singularities, there are no regularization tasks in order to avoid interaction with the equation of motion. This highlights the significance of being able to easily design safe hierarchical robot control problems with LexDynAH or LexDynBFGS. The reaching task is defined as an objective on priority level 2. A posture reference task is also added at the objective level. Similarly to the LM algorithm, it acts as a velocity damper to approach singular configurations; it has a low weight ($5 \cdot 10^{-2}$) that is tuned depending on the task to be performed. For example reaching for very far away targets requires a higher weight. A further regularization task $\gamma = 0$ on the contact forces is added to yield a fully determined and full-rank problem. It is weighted with a small value 10^{-4}.

For both solvers we substitute the torques τ with the equation of motion which reduces the number of variables from 112 to 74.

1) Experiment 1: The first experiment (exp. 1, see Fig. 18) 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 is a fixed bilateral contact that allows pulling or pushing forces.

In the following we describe the experiment for LexDynAH. At first, the robot moves its left hand to the vertical metal pole and grabs it. The pole position is determined by vision thanks to a marker provided by the `whycon` library [51]. During the movement, the CoM task on level 7 is constrained by the bounding box 1 $[\pm 0.03, \pm 0.1, \pm\infty]$ m and gets activated, see Fig. 19. The right hand on the next level 8, which must remain at its current position, is therefore in conflict and moves slightly backwards. This is a purely algorithmic singularity and triggers the switch to Newton's method. Figure 20 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.

Box 1 is widened from $[\pm 0.03, \pm 0.1, \pm\infty]$ m to

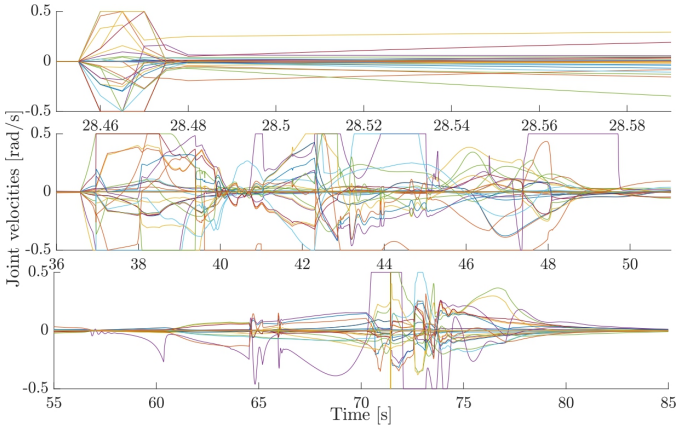

Fig. 22: **Exp. 1**, LexDynAH, joint velocities. The upper graph shows the moment when the CoM is released and the 52 active set iterations occur (followed by 5 occurrences of 30 iterations until 32.7 s). The middle one shows the moment when the right hand stretches to the right and starts being augmented. The lower graph shows the joint velocities when the robot stretches to the back and crouches.

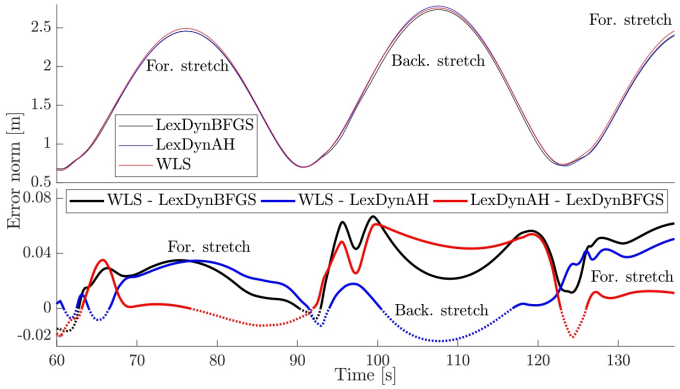

Fig. 23: **Exp. 1**, comparison of the error norm of the right hand tracking the swinging target during forward (for.) and backward (back.) stretch. The lower graph shows the differences of the error norms of the different methods.

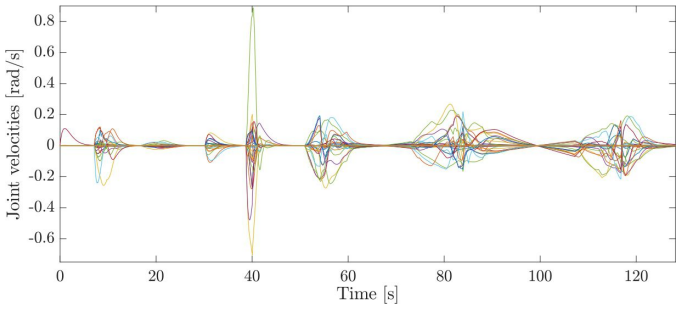

Fig. 24: **Exp. 1**, WLS, joint velocities.

$[\pm 0.2, 0.05 \pm 0.05, \pm\infty]$ m since we increase the support area of the robot in the sagittal plane due to the additional contact. However, several control iterations with a high number of active set is made to adjust the robot state; in Fig. 21 at 28 s notice the active set iterations peak (52). Figure 20 shows that at the instant of CoM release (around 28 s), both CoM tasks at level 7 and 9 are activated and shortly after deactivated again. Similar behavior is seen for the collision constraint between the left elbow and the torso. The upper graph of Fig. 22 shows how the velocity suddenly increases from zero to maximum velocity 0.5 s during the CoM release. The corresponding dynamic effects are also adjusted (the trust region and the contact force constraints, with possible interplay), resulting in a large number of active set iterations.

At around 36 s the augmentation of the level 8 position task stops as the right hand reached its prescribed position, see Fig. 20. The right hand then follows a target swinging from the front $[3, -1.5, 2]$ m to the back $[-3, -1.5, 0]$ m of the robot (for reference, HRP-2Kai is 1.71 m tall with a 2.11 m arm span). The target is always out of reach but numerically stable robot control is obtained thanks to the switch to the Newton's method. As can be seen from the middle and bottom graph of Fig. 22, the joint velocities are numerically stable and do not show signs of high frequency oscillations as for example observed in Fig. 14 for the GN algorithm. The joint velocities thereby exhibit a bang-bang type of behavior (generating joint velocities at the limits) for as fast as possible task convergence [52]. During the stretch motions, the CoM is well outside the support polygon of the feet (approximately at $[\pm 0.1, \pm 0.2, \pm\infty]$ m) which stresses the importance of the left hand supporting contact (see Fig. 19).

Figure 23 shows the norms of the tracking error of the right hand during the stretch motions. During forward stretches LexDynAH performs better than WLS ($\text{Err}_{\text{WLS}} - \text{Err}_{\text{LexDynAH}} > 0$). Especially during the second forward stretch (130s+) LexDynAH gets over 0.05 m closer to the target than WLS due to fully stretching its arm leading to a kinematic singularity. LexDynAH has a higher error norm than WLS during the backward stretch (see Fig. 23, 100 s to 117 s). This is despite the fact that the robot crouches more than seen for WLS (see video). This can be explained by local minima of the non-linear kinematic optimization problem created by joint limits.

Figure 21 shows that the computation time of LexDynAH, which includes the calculation time of the analytic Hessian, increases by about 1.5 ms as the right hand task gets augmented at time 8 s. An additional increase can be observed at 40 s when the CoM task on level 9 also requires augmentation (see Fig. 20). Now two expensive symmetric Schur decompositions of the hierarchical Hessian on level 8 and 9 are required.

LexDynBFGS is very similar to LexDynAH and shows the capabilities of the BFGS algorithm to provide a valid approximation of the analytic hierarchical Hessian. Computation times (Fig. 21) are lower than for LexDynAH since the hierarchical Hessian approximation is already positive definite.

WLS exhibits very smooth joint velocities (see Fig. 24) due to the conservatively chosen damping factor. This comes at the cost of worse convergence, especially compared to LexDynBFGS, see Fig. 23. 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 LexDynAH and LexDynBFGS (see video).

2) Experiment 2: With the second experiment (exp. 2) we show the importance of handling kinematic and algorithmic

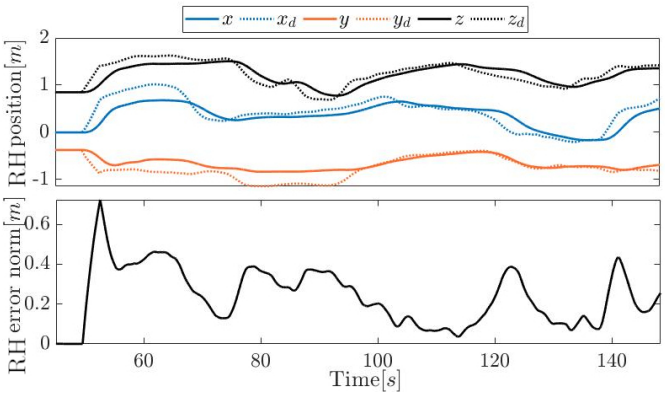

Fig. 25: **Exp. 2**, LexDynAH, right hand task error. The upper graph shows the actual right hand position in x, y and z direction and the desired ones x_d, y_d and z_d. The lower graph shows the error norm.

singularities without the reliance on heuristic damping tuning. This experiment is set up similarly to exp. 1. This time however the robot's right hand tracks a target held by a human operator which is sampled by a motion capture system. In such situations, and especially for a humanoid robot with multi-level prioritized constraints, it is difficult to determine the robot workspace beforehand. Our proposed methods LexDynAH and LexDynBFGS allow to only have approximate knowledge or no knowledge at all of the robot workspace while still enabling safe control of the robot. The target could be at the border of the workspace with the end-effector being in kinematic singularity or being in algorithmic conflict with a stable-balance task like the CoM task on a higher priority level. At the same time, the whole possible workspace is used. This might be in contrast to a conservatively tuned regularization term in order to ensure non-singular behavior over a range of robot configurations which might not be known in advance.

The right hand tracking error of LexDynAH is given in Fig. 25. From around 50 s onwards, the right hand is tracking the position of a handheld wand with markers detected 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. 25). Over all the course of the experiment the robot exhibits a numerically stable control. The marker is moved outside, on the border and inside of the robot's workspace with a left hand (LH) error norm of up to 70 cm. Note that in order to prevent too fast and unsafe motions, especially when the marker is far away, we set the right hand proportional task gain to $k_p = 0.5$ instead of $k_p = 1$ in the previous experiment. This leads to rather slow robot movements with relatively slow convergence behavior.

C. Performance comparison

The overall performance of LexDynBFGS and LexDynAH in the experiments is compiled comprehensively in Table I.

LexDynAH and LexDynBFGS both performed best in the category 'Error convergence'. The performance of WLS depends highly on the chosen damping. Its behavior can only be considered acceptable at best since determining 'perfect' damping can not be achieved with adaptive heuristics. The same holds for the 'stability' and 'smoothness' of the joint trajectories. If the damping weights for the posture reference task of WLS are too low, the numerical behavior is unstable and therefore not smooth. If the damping weights are too high we get very smooth joint trajectories but bad error convergence. LexDynAH and LexDynBFGS are numerically stable alternatives with relatively smooth joint trajectories. Especially, there is no need for damping tuning which makes it very easy to use ('Easiness of use').

	LexDyn AH	**LexDyn BFGS**	**WLS**
Error convergence	+	+	o / –
Joint stability	+	+	+ / –
Joint smoothness	o	o	+ / –
Easiness of use	+	+	o

TABLE I: Comprehensive overview of the performance of LexDynAH, LexDynBFGS and WLS in the evaluation. The symbols +, o, – indicate best, acceptable and worst performance in the corresponding evaluation criteria.

VII. Conclusion

In this paper we formulated the hierarchical Newton's method of control which enables numerically stable computations in case of singularities of kinematic tasks. We made the link to control and formulated second order PD controllers in the velocity domain with exponential convergence properties even in the case of regularization. Our simulations on toy robotic examples confirmed that such an approach is viable. Experiments with the HRP-2Kai, a complex humanoid robot, showed that the reliance on accurate representations of second order information –that is in contrast to approximations like classical damping methods– leads to better robot workspace occupancy and consequently better task error reduction. At the same time we confirmed that the notion of hierarchy enables formulating problems with strict safety prioritization.

Computation times of the hierarchical least squares solver are a limiting factor. 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 proposed in [39] or a more effective active set search.

VIII. Acknowledgment

We deeply thank Pierre-Brice Wieber and Dimitar Dimitrov for providing us with the code for LexLSI [39] which was indispensable for this work; and 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] 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.

[3] 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.

[4] F. Chaumette and E. Marchand, "A new redundancy-based iterative scheme for avoiding joint limits. application to visual servoing," in *IEEE International Conference on Robotics and Automation*, vol. 2, 2000, pp. 1720–1725.

[5] N. Mansard and F. Chaumette, "A new redundancy formalism for avoidance in visual servoing," in *IEEE/RSJ International Conference on Intelligent Robots and Systems*, 2005, pp. 468–474.

[6] Y. Abe, M. Da Silva, and J. Popović, "Multiobjective control with frictional contacts," *ACM SIGGRAPH Symposium on Computer Animation*, vol. 1, pp. 249–258, 2007.

[7] 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, Nov. 29 - Dec. 1 2007, pp. 81–88.

[8] S. Feng, X. Xinjilefu, W. Huang, and C. G. Atkeson, "3D walking based on online optimization," in *IEEE-RAS International Conference on Humanoid Robots*, 2013.

[9] S. Kuindersma, F. Permenter, and R. Tedrake, "An efficiently solvable quadratic program for stabilizing dynamic locomotion," in *IEEE International Conference on Robotics and Automation*, HK, China, 2014.

[10] 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.

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

[12] 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, 2019.

[13] 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.

[14] 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.

[15] G. Arechavaleta, A. Morales-Díaz, H. M. Pérez-Villeda, and M. Castelán, "Hierarchical task-based control of multirobot systems with terminal attractors," *IEEE Transactions on Control Systems Technology*, vol. 25, no. 1, pp. 334–341, 2017.

[16] T. Yoshikawa, "Manipulability of robotic mechanisms," *The International Journal of Robotics Research*, vol. 4, no. 2, pp. 3–9, 1985.

[17] T. Johansen, T. Fossen, and S. Berge, "Constrained nonlinear control allocation with singularity avoidance using sequential quadratic programming," *IEEE Transactions on Control Systems Technology*, vol. 12, no. 1, pp. 211–216, 2004.

[18] H. Leeghim, I.-H. Lee, D.-H. Lee, H. Bang, and J.-O. Park, "Singularity avoidance of control moment gyros by predicted singularity robustness: Ground experiment," *IEEE Transactions on Control Systems Technology*, vol. 17, no. 4, pp. 884–891, 2009.

[19] 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.

[20] 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. Butterworth-Heinemann, 2002, pp. 85 – 115.

[21] C. Guarino Lo Bianco and M. Raineri, "An experimentally validated technique for the real-time management of wrist singularities in nonredundant anthropomorphic manipulators," *IEEE Transactions on Control Systems Technology*, vol. 28, no. 4, pp. 1611–1620, 2020.

[22] 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.

[23] A. Kheddar, S. Caron, P. Gergondet, A. Comport, A. Tanguy, C. Ott, B. Henze, G. Mesesan, J. Englsberger, M. A. Roa, P.-B. Wieber, F. Chaumette, F. Spindler, G. Oriolo, L. Lanari, A. Escande, K. Chappellet, F. Kanehiro, and P. Rabaté, "Humanoid robots in aircraft manufacturing: The airbus use cases," *IEEE Robotics Automation Magazine*, vol. 26, no. 4, pp. 30–45, 2019.

[24] I. Kumagai, M. Morisawa, T. Sakaguchi, S. Nakaoka, K. Kaneko, H. Kaminaga, S. Kajita, M. Benallegue, R. Cisneros, and F. Kanehiro, "Toward industrialization of humanoid robots: Autonomous plasterboard installation to improve safety and efficiency," *IEEE Robotics Automation Magazine*, vol. 26, no. 4, pp. 20–29, 2019.

[25] 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.

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

[27] 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.

[28] 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.

[29] 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.

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

[31] P. Harish, M. Mahmudi, B. L. Callennec, and R. Boulic, "Parallel inverse kinematics for multithreaded architectures," *ACM Transactions on Graphics*, vol. 35, no. 2, pp. 1–13, Feb. 2016.

[32] 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.

[33] 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.

[34] 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.

[35] 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.

[36] 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.

[37] J. Wang, Y. Li, and X. Zhao, "Inverse kinematics and control of a 7-dof redundant manipulator based on the closed-loop algorithm," *Int. Journal of Advanced Robotic Systems*, vol. 7, no. 4, p. 37, 2010.

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

[39] D. Dimitrov, A. Sherikov, and P.-B. Wieber, "Efficient resolution of potentially conflicting linear constraints in robotics," Aug. 2015, v1. [Online]. Available: https://hal.inria.fr/hal-01183003

[40] T. Yenamandra, F. Bernard, J. Wang, F. Mueller, and C. Theobalt, "Convex optimisation for inverse kinematics," in *International Conference on 3D Vision*, Los Alamitos, CA, USA, sep 2019, pp. 318–327.

[41] H. Dai, G. Izatt, and R. Tedrake, "Global inverse kinematics via mixed-integer convex optimization," *The International Journal of Robotics Research*, vol. 38, no. 12-13, pp. 1420–1441, 2019.

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

[43] 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.

[44] K. Erleben and S. Andrews, "Inverse kinematics problems with exact hessian matrices," in *International Conference on Motion in Games*, 2017, pp. 14:1–14:6.

[45] G. H. Golub and C. F. Van Loan, *Matrix Computations (3rd Ed.)*. Baltimore, MD, USA: Johns Hopkins University Press, 1996.

[46] N. Higham, "Computing the polar decomposition with applications," *SIAM Journal on Scientific and Statistical Computing*, vol. 7, no. 4, pp. 1160–1174, 1986.

[47] 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.

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

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

[50] 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

[51] 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.

[52] Q. C. Pham, "A general, fast, and robust implementation of the time-optimal path parameterization algorithm," *IEEE Transactions on Robotics*, vol. 30, pp. 1533–1540, 2013.