Multi-task closed-loop inverse kinematics stability through semidefinite programming

Josep Marti-Saumell, Angel Santamaria-Navarro, Carlos Ocampo-Martinez and Juan Andrade-Cetto

***Abstract*— Today's complex robotic designs comprise in some cases a large number of degrees of freedom, enabling for multi-objective task resolution (*e.g.,* humanoid robots or aerial manipulators). This paper tackles the local stability problem of a hierarchical closed-loop inverse kinematics algorithm for such highly redundant robots. We present a method to guarantee this system stability by performing an online tuning of the closed-loop control gains. We define a semi-definite programming problem (SDP) with these gains as decision variables and a discrete-time Lyapunov stability condition as a linear matrix inequality, constraining the SDP optimization problem and guaranteeing the local stability of the prioritized tasks. To the best of authors' knowledge, this work represents the first mathematical development of an SDP formulation that introduces these stability conditions for a multi-objective closed-loop inverse kinematic problem for highly redundant robots. The validity of the proposed approach is demonstrated through simulation case studies, including didactic examples and a Matlab toolbox for the benefit of the community.**

I. Introduction

Modern robotic devices, such as humanoid robots [1], or unmanned aerial manipulators [2], share the characteristic of being highly kinematically redundant. This kinematic redundancy exists when a robot has more degrees of freedom (DOFs) than those required to fulfill a particular task, hence a robot controller may exploit the unused DOFs to simultaneously achieve other compatible objectives, which may be secondary. Solving a set of tasks, expressed in their respective task spaces, requires to find the appropriate joint commands, expressed in the robot joint space (also known as configuration space or C-space). This procedure is known as solving the inverse kinematics (IK) problem and is usually done at a differential level (*i.e.,* outputting joint velocities to perform trajectory tracking). These IK approaches generally suffer from drift as they do not receive any feedback from the actions executed by the robot. To overcome this issue, one can resort to closed-loop inverse kinematic (CLIK) schemes [3], [4], which consist in finding proper joint values such that the task errors (comparison of desired and actual values of a given task cost) are driven towards zero. Although most IK algorithms can be easily modified to become CLIK methods, there exist some implications when solving for several tasks.

To solve multiple tasks simultaneously for a redundant robot, a common approach is to introduce task priorities while combining them in a single control law. Hence, if the robot cannot fulfill all tasks, it can prioritize the solution of those placed at the top of the hierarchy. A method to solve the IK problem using a hierarchy of tasks was early introduced in [5]. This technique satisfies lower priority tasks only in the null space of the higher priority ones. A similar approach is taken in [6], this time using task-augmented Jacobians. By using these approaches, when two tasks are not independent (*i.e.,* when they share their corresponding null-space), the algorithm suffers from algorithmic singularities, leading to unstable joint velocities [7]. In [7], an algorithmic singularity robust method is proposed to solve for two tasks separately using a classical least-squares method. The conflict between tasks is filtered out by projecting the second task solution into the null-space of the first one. Even though the algorithmic singularity disappears, this is a suboptimal solution and hence, a greater tracking error appears. This approach is analyzed and extended for several tasks in [8] where the projection is done in the augmented null space. In this paper, we have drawn inspiration from [8] to formulate a CLIK problem. Although this null-space technique has been used in recent works like [2], [9] or [10], they usually lack of a rigorous formulation development. For instance, all these methods are usually developed in continuous time while afterward they are implemented in discrete time, hence bypassing the influence of the sampling time into the overall system's performance. Besides, the overall control law stability has not been analyzed and just analysis for individual tasks is presented, without a mention of the instability behaviors that can arise while combining them.

In closed-loop schemes, the analysis of the system stability is a must even though, as in this case, a low-level controller able to handle the computed velocities is assumed for each actuator. In task-priority CLIK problems, this stability might be affected by several factors. First, the completion of a particular task might prohibit the fulfillment of another eventual task due to their antagonistic nature (*e.g.,* the convergence of all errors might be compromised since, by construction, not all errors will converge towards zero). However, there exists the case where all task errors behave as desired if

J. Marti-Saumell, C. Ocampo-Martinez and J. Andrade-Cetto are with the Institut de Robòtica i Informàtica Industrial, CSIC-UPC, Llorens Artigas 4-6, Barcelona 08028, Spain (e-mail: jmarti, cocampo, cetto@iri.upc.edu).

A. Santamaria-Navarro is with NASA-JPL, California Institute of Technology, Pasadena, CA 91109 USA (e-mail: angel.santamaria.navarro@jpl.nasa.gov).

This work was partially supported by the EU H2020 project GAUSS (H2020-Galileo-2017-1-776293), project EB-SLAM (DPI2017-89564-P) and by the Spanish State Research Agency through the María de Maeztu Seal of Excellence to IRI (MDM-2016-0656). Part of this research was carried out at the Jet Propulsion Laboratory, California Institute of Technology, under a contract with the National Aeronautics and Space Administration (NASA, USA).

This paper has a supplementary downloadable video, available at http://ieeexplore.ieee.org, showing experimental results of the presented ap proach.

the proper closed-loop control gains are selected. Manually tuning these gains can easily lead to undesired behaviors as the error dynamics also depend on the robot configuration. In that sense, [11] presents the dependency between tasks and gains together with a way to measure it. Besides, [11] studies the local stability of a task-priority CLIK problem taking advantage of the Lyapunov theory, which allows finding conservative upper and lower bounds for the task gains. Unfortunately, the analytical developments presented in [11] are focused on three tasks and become intractable when extended to N tasks. A similar problem appears in [12] where stability conditions are only provided for a single task. An extension of [11] is [13], where the local stability of systems dealing with inequality tasks is analyzed, however, they assume a manually chosen set of gains and there is no insight provided on how to compute them. The gain-tuning solution is one of the main novelties presented hereafter in this paper.

Most of the existing methods that analyze the use of multiple tasks in a hierarchy are presented in continuous-time formulations [11], [13]. Discrete-time CLIK schemes add another factor to consider when proving the stability of the system: the sampling time selection. This effect is studied in [14] in the sense of Lyapunov theory, and in [12] without resorting to such theory. However, in both cases, the analysis refers to single tasks.

The novel contribution of this paper is a method to find optimal task feedback gains for the discrete hierarchical CLIK regulation problem, guaranteeing local stability of all tasks in the hierarchy. We choose a singularity free approach [8], describe it as a discrete-time CLIK system and, taking advantage of an SDP problem definition, we find the optimal gains which guarantee system stability. With the SDP approach defined hereafter, local stability can be guaranteed by adding a constraint to the optimization problem. This constraint is formulated in the sense of Lyapunov and as a linear matrix inequality (LMI), where the desired gains are the optimization variables. Apart from guaranteeing local stability, this constraint allows us to modify the error dynamics, *i.e.,* to get faster or slower error convergence towards zero. Besides, in the SDP we can account for the sampling time and also add further conditions, such as to limit the joint velocities. To the best of our knowledge, this is the first work that uses an SDP formulation to introduce local stability conditions in the solution of a hierarchical CLIK problem for highly redundant robots. We stress that this work addresses the *local* stability of the system as the global stability for multiple task hierarchical resolution is still an open question for discrete-time systems and we consider it out of the paper scope. For the sake of simplicity, in the rest of the paper we refer as stability to this local stability.

The remainder of this article is structured as follows. In Section II, we describe the required background, including the hierarchical IK and CLIK formulations. The system stability developments are presented in Section III. Section IV develops the formulation of the SDP problem to perform an online tuning of the task gains while guaranteeing closed-loop stability, together with additional constraints. The validation of the proposed method is illustrated in Section V. Finally, discussion and conclusions are drawn in Section VI.

II. Background

This section introduces the required background formulation related to IK and CLIK algorithms.

A. Hierarchical inverse kinematics

Our formulation has drawn inspiration from the algorithmic singularity-free IK presented in [8]. Let us define an i-th task $\boldsymbol{\sigma}_i(t) \in \mathbb{R}^{n_i}$, with n_i dimensions, as a function of the robot joints,

$$\boldsymbol{\sigma}_i(t) = \mathbf{f}_i(\mathbf{q}(t)) \,, \tag{1}$$

being $\mathbf{q} \in \mathbb{R}^{\nu}$ the joint values, *i.e.,* the robot configuration. Solving the IK problem consists of finding the appropriate values of $\mathbf{q}(t)$ such that $\boldsymbol{\sigma}_i(t)$ reaches some desired values (for the sake of simplicity, the dependence on time t will be omitted hereafter). This problem is typically tackled by differentiating (1) with respect to time, such as

$$\dot{\boldsymbol{\sigma}}_i = \mathbf{J}_i \dot{\mathbf{q}} \,, \tag{2}$$

where $\mathbf{J}_i \in \mathbb{R}^{n_i \times \nu}$ is the Jacobian matrix of the function $\mathbf{f}_i$ in (1). Then, the IK problem can be solved by inverting the Jacobian matrix $\mathbf{J}_i$

$$\dot{\mathbf{q}} = \mathbf{J}_i^{\dagger} \dot{\boldsymbol{\sigma}}_i \,, \tag{3}$$

being $\mathbf{J}_i^{\dagger} = \mathbf{J}_i^{\top} (\mathbf{J}_i \mathbf{J}_i^{\top})^{-1} \in \mathbb{R}^{\nu \times n_i}$ the expression of the Jacobian Moore-Penrose pseudo-inverse. Here, we assume to be working in a region free from kinematic singularities, hence $\mathbf{J}_i$ will be full rank and (3) does not make use of the damped pseudo-inverse as in [8].

In order to accomplish a secondary task simultaneously while imposing a hierarchy, one can take advantage of the motions residing in the null space of the primary task. That is, all motion belonging to the i-th task null space does not effect the variable $\dot{\boldsymbol{\sigma}}_i$. A first work presenting this technique is [5] where joint velocities for the secondary task are computed so as not to modify the primary task. However, as analyzed in [7], this method suffers from algorithmic singularities. Even though separate Jacobians from different hierarchy levels are full-rank, when joined together their respective null-spaces become linearly dependent. With the aim of overcoming these algorithmic singularities, [7] proposes a solution in which tasks at two different hierarchy levels are solved separately. Then, the low priority task solution is projected onto the null space of the task higher in the hierarchy. This technique is analyzed and generalized to more than two priority levels in [8]. Thus, in the case of having h hierarchy levels, the solution to the IK problem results in

$$\dot{\mathbf{q}} = \mathbf{J}_1^{\dagger} \dot{\boldsymbol{\sigma}}_1 + \overline{\mathbf{N}}_1 \mathbf{J}_2^{\dagger} \dot{\boldsymbol{\sigma}}_2 + \cdots + \overline{\mathbf{N}}_{h-1} \mathbf{J}_h^{\dagger} \dot{\boldsymbol{\sigma}}_h \,, \tag{4}$$

where $\overline{\mathbf{N}}_{h-1} = \mathbf{I}_n - \mathbf{J}_{1\ldots h-1}^{\dagger} \mathbf{J}_{1\ldots h-1}$ is the null-space projector of the augmented Jacobian matrix

$$\mathbf{J}_{1\ldots h-1} = [\mathbf{J}_1^{\top} \quad \mathbf{J}_2^{\top} \ldots \mathbf{J}_{h-1}^{\top}]^{\top} \,, \tag{5}$$

with $\mathbf{J}_{1\dots h-1} \in \mathbb{R}^{(n_1+\dots+n_{h-1})\times\nu}$. Note that here we consider one single task for each hierarchy level. In case of multiple tasks with the same priority, we can stack their vectors and Jacobian matrices and treat them as a single task. Projecting joint velocities into the null space of other tasks can be done as long as the robot is redundant in those dimensions (DOFs) required by the tasks higher up in the hierarchy; and that its configuration is in a region free from singularities.

B. Closed-loop inverse kinematics

The aforementioned IK solution has to be computed in the discrete-time domain. Thus, given a trajectory in the task-space we obtain its analogous in the joint space by numerical integration, *e.g.,* by using a first order Euler integration

$$\mathbf{q}^{(k+1)} = \mathbf{q}^{(k)} + \dot{\mathbf{q}}^{(k)}\Delta t\,, \tag{6}$$

where $\mathbf{q}^{(k)} = \mathbf{q}(t_k)$ and t_k is the time at integration step k (*i.e.,* $t_k = k\Delta t$, being Δt the sampling time). Notice that we abuse of the notation by using $\dot{\square}^{(k)}$ to express the velocity of a variable evaluated at time t_k. This IK discrete implementation entails a drifting problem provoked by numerical integration. To overcome it, we can formulate a closed-loop version of the IK problem (CLIK) with the task error defined as the difference between a desired and the actual values of the task variable.

The continuous-time version of the closed-loop solution defines the task error as

$$\tilde{\boldsymbol{\sigma}}_i = \boldsymbol{\sigma}_i^* - \boldsymbol{\sigma}_i\,, \tag{7}$$

and assigns to it an error dynamics such that

$$\dot{\tilde{\boldsymbol{\sigma}}}_i = -\boldsymbol{\Lambda}_i\tilde{\boldsymbol{\sigma}}_i\,, \tag{8}$$

where $\boldsymbol{\sigma}_i^* \in \mathbb{R}^{n_i}$ is the desired task value. In order to decrease the error towards zero, $\boldsymbol{\Lambda}_i \in \mathbb{R}^{n_i\times n_i}$ is a positive-definite diagonal matrix of suitable gains. Differentiating (7) with respect to time, combining it with (8) and isolating $\dot{\boldsymbol{\sigma}}_i$, we can directly substitute $\dot{\boldsymbol{\sigma}}_i = \dot{\boldsymbol{\sigma}}_i^* + \boldsymbol{\Lambda}_i\tilde{\boldsymbol{\sigma}}_i$ into (4). Hence, the analogous equation for a CLIK problem with h priority levels becomes

$$\dot{\mathbf{q}} = \mathbf{J}_1^\dagger\boldsymbol{\Lambda}_1\tilde{\boldsymbol{\sigma}}_1 + \overline{\mathbf{N}}_1\mathbf{J}_2^\dagger\boldsymbol{\Lambda}_2\tilde{\boldsymbol{\sigma}}_2 + \dots + \overline{\mathbf{N}}_{h-1}\mathbf{J}_h^\dagger\boldsymbol{\Lambda}_h\tilde{\boldsymbol{\sigma}}_h\,. \tag{9}$$

Notice how (9) includes different desired values and gain matrices for each task. Finally, when considering the discrete-time system, we can obtain $\dot{\mathbf{q}}^{(k)}$ from (6) by evaluating the multiple Jacobian matrices and task errors in (9) at time t_k.

This discrete-time CLIK formulation is considered in sections III and IV to state an SDP problem that, when solved, outputs the gains that render all tasks stable. Although all equations stated hereafter consider the discrete-time domain (*i.e.,* matrices and vectors must be evaluated at the corresponding t_k), for the sake of conciseness, the super-index (k) indicating the evaluation time will be only written when confusion may occur.

III. Stability analysis

In this section, we establish the conditions to guarantee the stability of the CLIK algorithm described above. By definition, a single task i is stable if its error decreases asymptotically to zero. However, when a hierarchy is applied to solve several tasks simultaneously for a redundant robot, the interaction between tasks affects the overall control law stability (*i.e.,* solving a task in the null space of tasks higher up in the hierarchy can also affect its own stability). As stated in the Introduction section with reference to [11], the stability of the closed-loop controlled scheme can be guaranteed if we assume independent tasks, as done in this work, and with proper tuning of the task gains $\boldsymbol{\Lambda}$. In the following, we present one of the key novelties of this paper: the condition to guarantee the overall stability considering a discrete-time system, which depends on these task gains. As the analytical computation of the gains for N tasks is unfeasible [11], the condition presented here will be later on introduced as a constraint in an SDP optimization procedure.

In order to analyze the stability of the whole system, let us consider an augmented vector containing all task errors:

$$\tilde{\boldsymbol{\sigma}}^\top = [\tilde{\boldsymbol{\sigma}}_1^\top \dots \tilde{\boldsymbol{\sigma}}_h^\top]\,, \tag{10}$$

being $\tilde{\boldsymbol{\sigma}} \in \mathbb{R}^n$ the augmented error, with $n = n_1 + \dots + n_h$ at time t_k. We can assess the stability of a system by resorting to the Lyapunov theory for discrete systems. Given a Lyapunov candidate function $V(\tilde{\boldsymbol{\sigma}}^{(k)}) = V^{(k)} > 0\,,$ $\forall\tilde{\boldsymbol{\sigma}}^{(k)} \neq \mathbf{0}$, the error will decrease towards zero if $V^{(k+1)} - V^{(k)} < 0$ holds, *i.e.,* if the Lyapunov candidate decreases with time [15]. We choose it to be

$$V(\tilde{\boldsymbol{\sigma}}) = \frac{1}{2}\tilde{\boldsymbol{\sigma}}^\top\tilde{\boldsymbol{\sigma}}\,. \tag{11}$$

Hence, to guarantee the stability of the system we must ensure that

$$\frac{1}{2}\tilde{\boldsymbol{\sigma}}^{(k+1)\top}\tilde{\boldsymbol{\sigma}}^{(k+1)} - \frac{1}{2}\tilde{\boldsymbol{\sigma}}^{(k)\top}\tilde{\boldsymbol{\sigma}}^{(k)} < 0\,, \tag{12}$$

where the error $\tilde{\boldsymbol{\sigma}}^{(k+1)}$ can be approximated by a Taylor series expansion of $\tilde{\boldsymbol{\sigma}}(t)$ around t_k and evaluating it at t_{k+1} up to the first term (first order Euler integration). Thus,

$$\tilde{\boldsymbol{\sigma}}^{(k+1)} \approx \tilde{\boldsymbol{\sigma}}^{(k)} + \dot{\tilde{\boldsymbol{\sigma}}}^{(k)}\Delta t\,. \tag{13}$$

As shown in [14], this approximation is valid as long as the higher-order terms remain small. According to this work, the higher-order terms can be neglected if $||\dot{\mathbf{q}}||\Delta t$ is below a certain bound. To fulfill this statement, we add another constraint to our SDP problem, as explained in the following section. Notice that guaranteeing (12) implies local stability depending on the tasks gains, the sampling time and the initial value of the error. In this paper, we propose a solution considering the first two factors (tasks gains and sampling time) and assume, as commonly done in the literature, an initial value of the error that can keep the problem feasible (*i.e.,* limiting the initial error by keeping the robot and tasks within proper operative conditions). In [12], it is proposed a method to estimate the region of attraction for a single task,

depending on the task gain selected, the sampling time and several parameters related to the derivatives of the $\mathbf{f}(\mathbf{q}(t))$. However, an estimation of the region of attraction for several tasks still remains as an open problem, hence we limit the scope of the paper to guarantee local stability.

Now we can substitute (13) in (12) obtaining

$$V^{(k+1)} - V^{(k)} \approx \frac{1}{2}(\dot{\tilde{\boldsymbol{\sigma}}}^{(k)\top}\tilde{\boldsymbol{\sigma}}^{(k)}\Delta t + \tilde{\boldsymbol{\sigma}}^{(k)\top}\dot{\tilde{\boldsymbol{\sigma}}}^{(k)}\Delta t + + \dot{\tilde{\boldsymbol{\sigma}}}^{(k)\top}\dot{\tilde{\boldsymbol{\sigma}}}^{(k)}\Delta t^2)\,, \tag{14}$$

which no longer depends on $\tilde{\boldsymbol{\sigma}}^{(k+1)}$ and therefore, from now on, the super-index k will be omitted. In order to keep developing (14), we use the following expression for the error velocity evaluated at time t_k,

$$\dot{\tilde{\boldsymbol{\sigma}}} = \begin{bmatrix} \dot{\tilde{\boldsymbol{\sigma}}}_1 \\ \vdots \\ \dot{\tilde{\boldsymbol{\sigma}}}_h \end{bmatrix} = - \begin{bmatrix} \mathbf{J}_1 \\ \vdots \\ \mathbf{J}_h \end{bmatrix} \dot{\mathbf{q}}\,. \tag{15}$$

Notice how this expression can be obtained by differentiating (7) and combining it with (2). This can be further expanded by using the CLIK solution in (9) resulting in the following linear mapping

$$\begin{aligned} \dot{\tilde{\boldsymbol{\sigma}}} &= \begin{bmatrix} -\mathbf{J}_1\mathbf{J}_1^{\dagger}\boldsymbol{\Lambda}_1 & \cdots & -\mathbf{J}_1\overline{\mathbf{N}}_{h-1}\mathbf{J}_h^{\dagger}\boldsymbol{\Lambda}_h \\ \vdots & \ddots & \vdots \\ -\mathbf{J}_h\mathbf{J}_1^{\dagger}\boldsymbol{\Lambda}_1 & \cdots & -\mathbf{J}_h\overline{\mathbf{N}}_{h-1}\mathbf{J}_h^{\dagger}\boldsymbol{\Lambda}_h \end{bmatrix} \tilde{\boldsymbol{\sigma}}\,, \\ \dot{\tilde{\boldsymbol{\sigma}}} &= \mathbf{A}\tilde{\boldsymbol{\sigma}}\,. \end{aligned} \tag{16}$$

Now, by substituting (16) into (14), we obtain the following quadratic equation

$$V^{(k+1)} - V^{(k)} \approx \frac{1}{2}\tilde{\boldsymbol{\sigma}}^{\top}(\mathbf{A}^{\top}\Delta t + \mathbf{A}\Delta t + \mathbf{A}^{\top}\mathbf{A}\Delta t^2)\tilde{\boldsymbol{\sigma}}\,. \tag{17}$$

Therefore, the stability of the stack of tasks will be guaranteed if we can assure the positive definiteness of the expression

$$-\mathbf{A}^{\top}\Delta t - \mathbf{A}\Delta t - \mathbf{A}^{\top}\mathbf{A}\Delta t^2 \succ 0\,, \tag{18}$$

leading the candidate Lyapunov function to become an actual Lyapunov function. The symbol "$\succ$" stands for positive definite matrix. Notice how for convenience, we expressed (18) as a positive definite matrix condition by multiplying the sum in (17) by -1.

System stability is guaranteed if we can find proper gain values $\boldsymbol{\Lambda}_i$ so that condition (18) holds at every time step. In this paper, we choose these gains as the decision variables in the SDP optimization problem and, by performing an online gain tuning, we account for the error dynamic change at every specific robot configuration. However, notice how (18) depends quadratically on the different $\boldsymbol{\Lambda}_i$. As an SDP formulation requires linear matrix inequalities (LMIs), further developments are detailed hereafter.

IV. SDP-BASED GAIN SCHEDULING

We want to formulate an SDP problem with the stability condition (18) as an LMI constraint. So, this SDP will be used to find the optimal closed-loop control gains that guarantee the stability of the error in (10). An SDP problem is a convex optimization problem whose feasible set is a cone formed by positive semidefinite symmetric matrices [16], [17]. Making use of LMIs, this kind of problems allows us to impose constraints on the definiteness of matrices. In this case, we choose a specific form of SDP problem to solve, expressed as

$$\begin{aligned} \min_{\mathbf{x}} \quad & \mathbf{c}^{\top}\mathbf{x}\,, \\ \text{s.t.} \quad & \mathbf{F}(\mathbf{x}) \succeq 0\,, \end{aligned} \tag{19}$$

where $\mathbf{x} = [x_1, \ldots, x_r]^{\top} \in \mathbb{R}^r$ is the vector of decision variables, $\mathbf{c} \in \mathbb{R}^r$ is a vector of coefficients and $\mathbf{F}(\mathbf{x}) \in \mathbb{R}^{s\times s}$ is a positive semi-definite LMI (noted with "$\succeq$"). The dimensions r and s will be defined hereafter.

Our goals in specifying the elements of (19) are to impose closed-loop stability and limit the resulting joint velocities, so the approximation in (13) holds, while trying to impose a convergence speed. Each of these constraints will be defined as single LMIs $\mathbf{F}_j(\mathbf{x})$, described in the following subsections. Afterwards, all these single LMIs will be formulated as an LMI of the form $\mathbf{F}(\mathbf{x})$ by placing them into a block diagonal matrix

$$\mathbf{F}(\mathbf{x}) = \mathsf{diag}(\mathbf{F}_1(\mathbf{x}), \ldots, \mathbf{F}_m(\mathbf{x})) \succeq 0\,, \tag{20}$$

with m the number of single LMIs.

The optimized outputs of the SDP problem are the $\boldsymbol{\Lambda}_i$ gain matrices, which have the form

$$\boldsymbol{\Lambda}_i = \mathsf{diag}(\lambda_{i,1}, \ldots, \lambda_{i,n_i})\,, \text{ for } i = 1, \ldots, h\,, \tag{21}$$

thus the $\boldsymbol{\lambda}_i = [\lambda_{i,1}, ..., \lambda_{i,n_i}]^{\top}$ vectors will be part of the decision variable $\mathbf{x}$ in (19). Therefore, each single LMI $\mathbf{F}_j(\mathbf{x})$ will have the form

$$\mathbf{F}_j(\mathbf{x}) = \mathbf{F}_{j,0} + \mathbf{F}_{j,1}\lambda_{i,1} + \cdots + \mathbf{F}_{j,n_i}\lambda_{i,n_i} \succeq 0\,, \quad \text{for } i = 1, \ldots, h\,. \tag{22}$$

In the following subsections, we describe in detail the form of each of these single LMIs $\mathbf{F}_j(\mathbf{x})$ that form the SDP constraint $\mathbf{F}(\mathbf{x})$ in (20). For the sake of simplicity, we join all task gains in a single vector of n elements, *i.e.,* $\boldsymbol{\lambda} = [\lambda_{1,1}, ..., \lambda_{1,n_1}, ..., \lambda_{h,1}, ...\lambda_{h,n_h}]^{\top} \in \mathbb{R}^n$ with $n = \sum_{s=1}^{h} n_s$, being the sum of task dimensions. Besides, notice that online gain tuning consists of solving the SDP problem at each iteration, obtaining different $\boldsymbol{\lambda}$ at each t_k.

A. $\mathbf{F}_1$: *Stability*

In order to express (18) as an LMI, we require some mathematical manipulations. On the one hand, notice that (18) imposes the strict definiteness ($\succ$) in contrast to the semidefiniteness ($\succeq$) required by LMIs in (19). To accomplish with this restriction, we can add a scalar factor $\alpha > 0$ that when

multiplied by the identity of suitable dimensions, will impose the strict positive definitness on (18), *i.e.,*

$$-\mathbf{A}^\top \Delta t - \mathbf{A}\Delta t - \mathbf{A}^\top \mathbf{A}\Delta t^2 \succeq \alpha \mathbf{I}\,. \tag{23}$$

Notice that, if we consider the scalar $\beta \triangleq \frac{\alpha}{\Delta t}$, (23) can be further simplified, obtaining

$$-\mathbf{A}^\top - \mathbf{A} - \mathbf{A}^\top \mathbf{A}\Delta t \succeq \beta \mathbf{I}\,. \tag{24}$$

Note also how β can be used to modify the error dynamics, so the higher the value of β, the faster the errors will convergence. However, setting β too high can jeoparadize the SDP feasibility as it is contradictory with limiting the joints velocities. To overcome this issue, we set a soft equality constraint on β in the SDP problem (see Section IV-C for implementation details).

Notice how (24) depends quadratically on the task gains. To express it as an LMI (linear dependence on gains), we take advantage of the Schur complement for symmetric matrices. First, let us define a symmetric matrix of suitable dimensions as

$$\mathbf{M} = \begin{bmatrix} \mathbf{B} & \mathbf{C} \\ \mathbf{C}^\top & \mathbf{D} \end{bmatrix}\,. \tag{25}$$

The Schur complement condition for positive definiteness states that, considering $\mathbf{D}$ is positive definite, $\mathbf{M}$ is positive semi-definite *if and only if* its Schur complement $\mathbf{M}|\mathbf{D}$ is positive semi-definite. Hence,

$$\text{If } \mathbf{D} \succ 0, \text{ then } \mathbf{M} \succeq 0 \Leftrightarrow \mathbf{M}|\mathbf{D} = \mathbf{B} - \mathbf{C}\mathbf{D}^{-1}\mathbf{C}^\top \succeq 0\,. \tag{26}$$

Then, doing the proper assignments, the expression (24) becomes the Schur complement of the matrix

$$\mathbf{M} = \begin{bmatrix} -(\mathbf{A}^\top + \mathbf{A}) - \beta\mathbf{I} & \mathbf{A}^\top \Delta t^{1/2} \\ \mathbf{A}\Delta t^{1/2} & \mathbf{I} \end{bmatrix} \succeq 0\,, \tag{27}$$

which finally, depends linearly on the gains $\boldsymbol{\lambda}_i$.

The details on how to express $\mathbf{M}$ in terms of $\boldsymbol{\Lambda}$, $\mathbf{M}(\boldsymbol{\Lambda})$, are presented in the following. This procedure is only detailed for the block $(\mathbf{A}^\top + \mathbf{A})$. The developments for the rest of the blocks are straight-forward and are here omitted for the sake of brevity.

We can express the matrix $\mathbf{A}$ in terms of the different gain matrices $\boldsymbol{\Lambda}_i$ as

$$\mathbf{A}(\boldsymbol{\Lambda}) = \begin{bmatrix} \mathbf{A}_{1,1} & \cdots & \mathbf{A}_{1,h} \\ \vdots & \ddots & \vdots \\ \mathbf{A}_{h,1} & \cdots & \mathbf{A}_{h,h} \end{bmatrix} \begin{bmatrix} \boldsymbol{\Lambda}_1 & & \mathbf{0} \\ & \ddots & \\ \mathbf{0} & & \boldsymbol{\Lambda}_h \end{bmatrix}\,, \tag{28}$$

being $\mathbf{A}_{i,\rho} = -\mathbf{J}_i \overline{\mathbf{N}}_{\rho-1} \mathbf{J}_\rho^\dagger$ for $i, \rho = 1, \ldots, h$ from (16), with $\overline{\mathbf{N}}_0 = \mathbf{I}$. Then, using (28) we obtain the symmetric matrix

$$\mathbf{A}^\top(\boldsymbol{\Lambda}) + \mathbf{A}(\boldsymbol{\Lambda}) = \begin{bmatrix} \mathbf{A}_{1,1}\boldsymbol{\Lambda}_1 + \boldsymbol{\Lambda}_1\mathbf{A}_{1,1}^\top & \cdots & \mathbf{A}_{1,h}\boldsymbol{\Lambda}_h + \boldsymbol{\Lambda}_1\mathbf{A}_{h,1}^\top \\ \vdots & \ddots & \vdots \\ \mathbf{A}_{h,1}\boldsymbol{\Lambda}_1 + \boldsymbol{\Lambda}_h\mathbf{A}_{1,h}^\top & \cdots & \mathbf{A}_{h,h}\boldsymbol{\Lambda}_h + \boldsymbol{\Lambda}_h\mathbf{A}_{h,h}^\top \end{bmatrix}\,. \tag{29}$$

The stability LMI requires $\mathbf{M}(\boldsymbol{\Lambda})$ to be positive definite, *i.e.,* $\mathbf{F}_1(\boldsymbol{\lambda}) = \mathbf{M}(\boldsymbol{\Lambda})$. To express $\mathbf{M}$ in terms of the corresponding vector $\boldsymbol{\lambda}$, which is part of the decision variables, the matrices $\mathbf{F}_{1,l}$ with $l \in [1, n]$ in (22) must be the elements of $\mathbf{M}$ that are multiplied by the corresponding gains in $\boldsymbol{\lambda}$. Notice that this LMI also takes into account the sampling time Δt, whose effect is shown in Section V.

B. $\mathbf{F}_2$*: Joint velocity limits*

In order to make the approximation in (13) hold, we must bound the joint velocity so second-order terms of the Taylor expansion do not become significant. With that aim, we resort to (9) to express every component of $\dot{\mathbf{q}} = [\dot{q}_1, \ldots, \dot{q}_n]^\top$ as a linear combination of the gains, hence we can transform (9) into

$$\dot{\mathbf{q}} = \mathbf{J}_1^\dagger \boldsymbol{\Sigma}_1 \boldsymbol{\lambda}_1 + \cdots + \overline{\mathbf{N}}_{h-1} \mathbf{J}_h^\dagger \boldsymbol{\Sigma}_h \boldsymbol{\lambda}_h\,, \tag{30}$$

where we express every $\boldsymbol{\Lambda}_i$ in its vector form and the error vector of every task is replaced by a diagonal matrix, *i.e.,* $\boldsymbol{\Sigma}_h = \mathsf{diag}(\dot{\tilde{\boldsymbol{\sigma}}}_h)$. This expression can be rearranged as

$$\dot{\mathbf{q}} = \mathbf{S}\boldsymbol{\lambda}\,, \tag{31}$$

with

$$\mathbf{S} \triangleq \left[\mathbf{J}_1^\dagger \boldsymbol{\Sigma}_1 | \cdots | \overline{\mathbf{N}}_{h-1} \mathbf{J}_h^\dagger \boldsymbol{\Sigma}_h\right]\,. \tag{32}$$

Then, we can add upper bounds to the joints velocities considering

$$\bar{\dot{\mathbf{q}}} - \mathbf{S}\boldsymbol{\lambda} \geq 0\,, \tag{33}$$

where $\bar{\dot{\mathbf{q}}}$ is the vector containing the upper bounds, and the symbol $\geq$ stands for the element-wise operand $\geq$. Finally, we can convert (33) into an LMI of the form $\mathbf{F} \succeq 0$ by specifying

$$\mathbf{F}_{2,0} = \mathsf{diag}(\bar{\dot{\mathbf{q}}})\,, \tag{34a}$$

$$\mathbf{F}_{2,j} = -\mathsf{diag}(\mathbf{s}_j) \text{ for } j = 1, \ldots, n\,, \tag{34b}$$

being $\mathbf{s}_j$ the j-th column of $\mathbf{S}$. Notice that $\mathbf{F}_{2,j} \in \mathbb{R}^{\nu \times \nu}$.

The addition of lower bounds is done analogously to the upper bounds procedure and has been omitted here for the sake of brevity.

C. $\mathbf{F}_3$*: Soft constraint on* β

The variable β will direct the velocity of the error convergence in (24). Although high values of β may lead to fast convergence, this behavior might go against limiting the joints' velocities with the constraint described above, thus it difficults the convergence of the SDP problem. To avoid this, we can define a soft constraint on β with an initial desired value $\tilde{\beta}$, such that

$$\begin{aligned} &\min\ ||\beta - \tilde{\beta}||^2 + \delta||\boldsymbol{\lambda}||^2\,, \\ &\text{s.t. } \mathbf{F}_1, \mathbf{F}_2 \succeq 0\,. \end{aligned} \tag{35}$$

Now, the error will converge with a speed imposed by $\tilde{\beta}$, if possible. Otherwise, the constraint will be relaxed so the system is stable ($\mathbf{F}_1$) and the Euler approximation in (13) holds ($\mathbf{F}_2$). Setting a high value for $\tilde{\beta}$ might be seen as maximizing the speed convergence while keeping the stability and the joints' velocity limits. Notice that, for the

sake of the problem solvability, it is also necessary to add a regularization term δ related to the gains.

In order to convert (35) into an LMI we must provide a linear objective function. This can be done by using its epigraph form, *i.e.,* upper-bounding the quadratic expression with an additional optimization variable $\gamma \in \mathbb{R}$. Thus, we will minimize γ subject to the following constraint

$$\gamma - (\beta - \tilde{\beta})^2 - \delta \boldsymbol{\lambda}^\top \boldsymbol{\lambda} \geq 0\,. \tag{36}$$

Again, we can express the constraint (36) as an LMI by taking advantage of the Schur complement

$$\mathbf{F}_3(\boldsymbol{\lambda}, \beta, \gamma) = \begin{bmatrix} \gamma & \boldsymbol{\lambda}^\top & \beta - \tilde{\beta} \\ \boldsymbol{\lambda} & \delta^{-1}\mathbf{I} & \mathbf{0} \\ \beta - \tilde{\beta} & \mathbf{0} & 1 \end{bmatrix} \succeq \mathbf{0}\,. \tag{37}$$

Notice that an extra constraint to guarantee $\beta > 0$ is also necessary. For the sake of conciseness and due to its simplicity, it has not been detailed here.

By introducing β and γ in the problem, we are adding two new components to the vector of decision variables, hence $\mathbf{x} = [\boldsymbol{\lambda}^\top, \beta, \gamma]^\top$. Therefore, two extra matrices $\mathbf{F}_{3,n+1}$ and $\mathbf{F}_{3,n+2}$ should be added in the computation of $\mathbf{F}_3$ in (22), which will be multiplied by β and γ, respectively. Notice how the previous LMIs ($\mathbf{F}_1, \mathbf{F}_2$) do not depend on γ nor β and their corresponding $\mathbf{F}_{\{j,n+1\}}$ matrices can be omitted since they are null. With γ the variable to be minimized, we can define the coefficient vector $\mathbf{c}$ of the cost function in (19) as a vector with zeros in the positions related to each component of $\boldsymbol{\lambda}$ and β (*i.e.,* $n+1$ elements with 0) and a one in the position of the upper bound γ, hence $\mathbf{c} = [\mathbf{0}_{1\times n+1}, 1]^\top \in \mathbb{R}^{n+2}$.

D. Final SDP

In summary, considering the described LMIs *i.e.,* $\mathbf{F}_j$ with $j \in [1, 2, 3]$, the SDP problem formulation shown in (19) results in

$$\begin{aligned} \min_{\mathbf{x}} \quad & \mathbf{c}^\top \mathbf{x} \\ \text{s.t.} \quad & \mathsf{blockdiag}(\mathbf{F}_1(\mathbf{x}), \mathbf{F}_2(\mathbf{x}), \mathbf{F}_3(\mathbf{x})) \succeq 0\,. \end{aligned} \tag{38}$$

V. Validation

Task stabilization through the optimization procedure presented in this work is of use when dealing with highly redundant robots that must perform several tasks. In these cases, the analytical study to choose the right gains, hence to guarantee stability, becomes unfeasible. When the number of tasks and required DOFs are not high (*e.g.,* two tasks and three DOFs), this method is also of use as it eases the gains search that will render all tasks stable, without the need for simplifications as in other existing methods (*e.g.,* it allows to consider different gains for each task dimension).

The effectiveness of the mathematical developments proposed in this work can be better explained with robots with a low number of DOFs. Hence, without loss of generality, we present a numerical experiment with the commercial UR5[1] (6 DOFs) robotic arm performing with the on-line task gain tuning approach. These examples suffice to validate that:

a) In all simulations the stability condition in (18) holds. This can be checked by looking at the highest eigenvalue of the sum in (17), which should be negative as sign for stability.
b) Variations on $\tilde{\beta}$ truly affect the convergence speed. For higher values of $\tilde{\beta}$ the tasks should converge faster.
c) Joint velocity bounds $\bar{\dot{\mathbf{q}}}, \underline{\dot{\mathbf{q}}}$ are respected and the system still manages to converge by relaxing β.
d) The method can handle different values of Δt without affecting the system stability.

We have performed the simulations by taking advantage of `Matlab` and the toolbox presented in [18] to simulate a robot manipulator. Moreover, we use the already existing SDP solver `Sedumi` [19]. All code related with this paper is made publicly available for the benefit of the community[2].

We have set a use case with the aim of reproducing a hand-writing action done by a human arm. Thus, we set a primary task to follow a desired position path with the robot's end effector, whose error is described by $\tilde{\boldsymbol{\sigma}}_1 \in \mathbb{R}^3$. Besides, we impose the wrist to be close to the writing surface as a secondary task, *i.e.,* we impose the y coordinate of the 4th joint to have a specific value ($\tilde{\sigma}_2 \in \mathbb{R}$).

The parameters for this case study are $\mathbf{q}_0 = [135, 0, -90, 0, 90, 0]^\top$deg, $\boldsymbol{\sigma}_1^* = [-0.5, -0.4, 0.6]^\top$m and $\sigma_2^* = -0.3$m. The regularization parameter is $\delta = 5 \times 10^{-5}$.

We have performed several experiments to validate the items (a)-(d) stated above. Although the system stability (a) has been confirmed in all simulations, for the sake of conciseness, it is only reported here the figure that shows the stability of the experiments where we want to show the effectiveness of $\tilde{\beta}$. In those cases, we have set $\Delta t = 0.01$s and $\bar{\dot{\mathbf{q}}} = -\underline{\dot{\mathbf{q}}} = \mathbf{6}$ rad/s (we have used a boldface number to indicate the same limit for all the robot's joints) and ran several simulations to compare our method with both $\tilde{\beta} = 2$ and $\tilde{\beta} = 8$ against the tasks deployment with $\boldsymbol{\lambda} = [2, 2, 2, 1]^\top$ as fixed gains. For the sake of readability, all plots for this robot type are time-cropped to 4s.

The corresponding results are shown in Fig. 1, where the stability of the CLIK method is not guaranteed when using constant $\boldsymbol{\lambda}$ gains (notice the positive maximum eigenvalue indicated by the solid blue line). The same fact can be seen in Fig. 2. It shows the Lyapunov function (11). Notice that, while for the case of using the presented method the Lyapunov function decreases monotonously whereas for the case with constant gains the Lyapunov function remains stable once the first task has converged. The direct consequence of not having a proper gain tuning method is that tasks with low priority are less prone to converge (see Fig. 3b).

In the example, we see that a high $\tilde{\beta}$ translates to faster error convergence for all tasks. This effect can be seen in Fig. 2 and Fig. 3 for the primary and secondary tasks.

[1] `https://www.universal-robots.com/products/ur5-robot/`

[2] `https://gitlab.iri.upc.edu/jmarti/SDP_HierarchicalTaskStability`

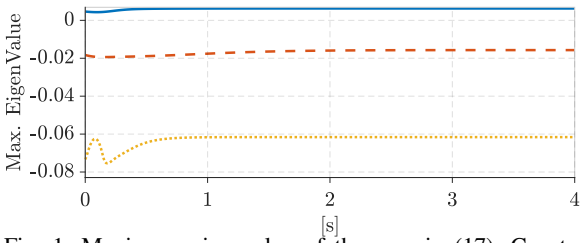

Fig. 1: Maximum eigenvalue of the sum in (17). Constant gains (solid blue), $\tilde{\beta} = 2$ (dashed red), $\tilde{\beta} = 8$ (dotted yellow). The reported values must negative to guarantee stability.

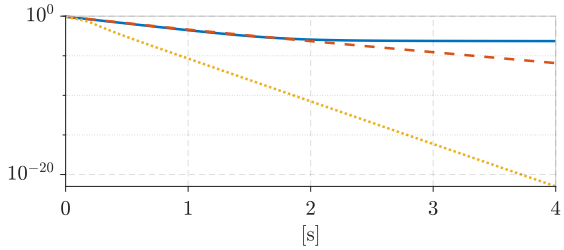

Fig. 2: Lyapunov function considering different values of $\tilde{\beta}$. Constant gains (solid blue), $\tilde{\beta} = 2$ (dashed red) and $\tilde{\beta} = 8$ (dotted yellow).

In this experiment, the manipulator is driven towards a configuration where the two tasks are close to be dependant, *i.e.,* where the rank of the augmented Jacobian is smaller than the sum of ranks of the respective Jacobians. In such a situation, the null-space of the first task becomes smaller, hence the joint velocities devoted to fulfill the low priority task become smaller. When operating in a close-loop manner, this vanishing joint velocity can be tackled by increasing the gains, a benefit of imposing (23) within our SDP approach. This effect is shown in Fig. 4, where the orientation task gain is increased as the configuration approaches the singularity.

As previously noted, imposing faster error dynamics by increasing $\tilde{\beta}$ results in increasing the joint velocities. This fact might violate the joint velocity constraint ($\mathbf{F_2}$) if β was not properly relaxed ($\mathbf{F_3}$). This case is shown in Fig. 5, where the limits for the joints velocities are respected while the β value is moved away from its desired $\tilde{\beta}$ (Fig. 6), in order to keep the joints velocity limits while guaranteeing the system stability. Fig. 6 shows the impact on β while imposing different joint velocity limits, *i.e.,* the more restricting joint velocities (solid blue line), the further β has to be moved from its desired value.

The effect of the sampling time in the system performance is depicted in Fig. 7 and Fig. 8. We have set the desired speed to $\tilde{\beta} = 8$ and the joint velocity limits to $\overline{\dot{\mathbf{q}}} = -\underline{\dot{\mathbf{q}}} = \mathbf{6}$ rad/s across all different Δt. As with previous experiments, the method manages to stabilize the two tasks for all the tested Δt, *i.e.,* in all cases the maximum eigenvalue remains negative. Therefore, as shown in Fig. 7, the different task gains must be adapted (we could not notice a remarkable difference for the second task's gain). Besides, the speed convergence of the second task improves with smaller sampling time (see Fig. 8). An important remark is that for a sufficiently small Δt, the quadratic term vanishes and (24) becomes the stability condition of the continuous-time CLIK algorithm. This fact is shown in both Fig. 7 and Fig. 8, where the

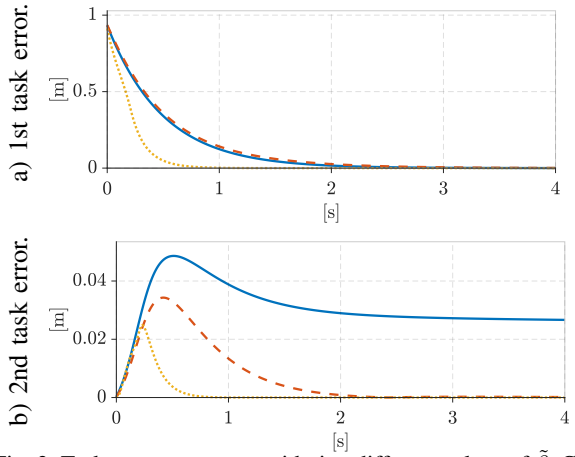

Fig. 3: Tasks error norms considering different values of $\tilde{\beta}$. Constant gains (solid blue), $\tilde{\beta} = 2$ (dashed red), $\tilde{\beta} = 8$ (dotted yellow).

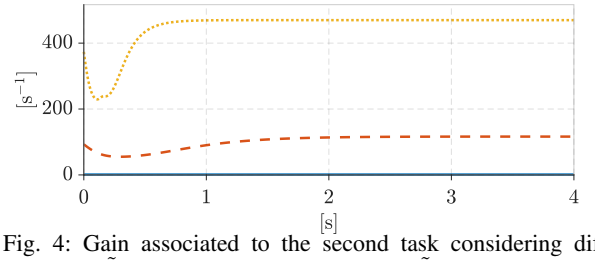

Fig. 4: Gain associated to the second task considering different values of $\tilde{\beta}$. Constant gains (solid blue), $\tilde{\beta} = 2$ (dashed red) and $\tilde{\beta} = 8$ (dotted yellow).

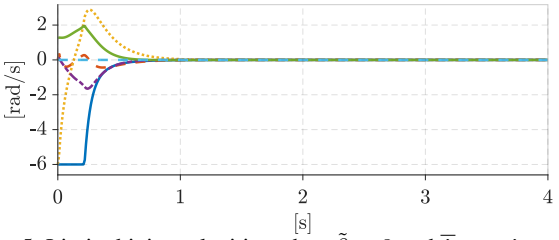

Fig. 5: Limited joint velocities when $\tilde{\beta} = 8$ and $\overline{\dot{\mathbf{q}}} = -\underline{\dot{\mathbf{q}}} = \mathbf{6}$ rad/s. Colors related to each joint in increasing order: solid blue, dashed red, dotted yellow, dashed magenta, solid green, dashed cyan.

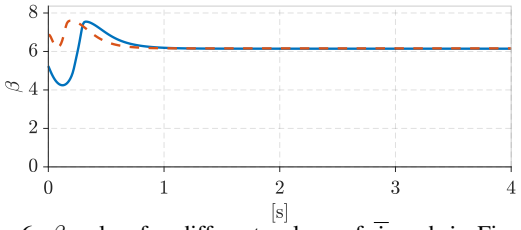

Fig. 6: β value for different values of $\overline{\dot{\mathbf{q}}}$ and $\underline{\dot{\mathbf{q}}}$. Fixed value of $\tilde{\beta} = 8$. $\overline{\dot{\mathbf{q}}} = -\underline{\dot{\mathbf{q}}} = \mathbf{4}$ rad/s (solid blue), $\overline{\dot{\mathbf{q}}} = -\underline{\dot{\mathbf{q}}} = \mathbf{6}$ rad/s (dashed red).

trajectories converge to a specific solution as Δt decreases.

VI. Discussion and Conclusions

In this paper, we presented a novel approach to guarantee the stability of a hierarchical multi-task CLIK scheme. Differently from other existing works, this optimization-based approach allow us to extend the proper gain tuning that guarantees such stability for N number of tasks. Besides, our method allows to modify the error convergence speed. It also considers the effects of sampling time (discrete-time)

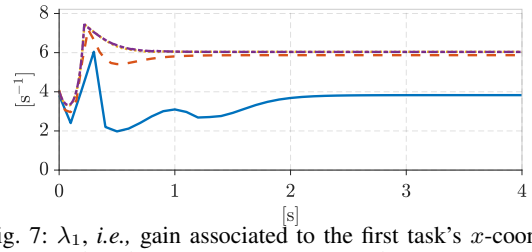

Fig. 7: λ_1, *i.e.,* gain associated to the first task's x-coordinate, for different Δt. $\Delta t = 0.1$s (solid blue), $\Delta t = 0.05$s (dashed red), $\Delta t = 0.01$s (dotted yellow), $\Delta t = 0.005$s (dashed magenta).

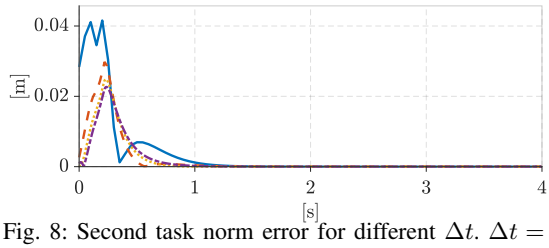

Fig. 8: Second task norm error for different Δt. $\Delta t = 0.1$s (solid blue), $\Delta t = 0.05$s (dashed red), $\Delta t = 0.01$s (dotted yellow), $\Delta t = 0.005$s (dashed magenta).

and allows us to add a constraint to limit the joint velocity limits.

The focus of this work has been the mathematical development to introduce the use of an SDP approach with its advantages with respect to state-of-art methods. However, some issues still remain open for further investigation. First, since this is a problem of local stability, the error has an associated region of attraction which should be estimated. In [12], a method is proposed to estimate this region for a single task, however, an estimation for multiple tasks still remains as an open problem. Second, the development of the stability condition is based upon an Euler approximation of the error (13), which is valid for a sufficiently small value of $||\dot{\mathbf{q}}||\Delta t$. An upper bound for this product should be found. Third, an open issue is how this method can be adapted to work in the acceleration domain so as to consider actuator torque limitations. Finally, even though SDP techniques using LMIs offer the possibility of using fast and dedicated solvers, our method is based on iterative procedures (*i.e.,* optimization), which in contrast to analytical solutions requires some extra effort in efficient programming to run it in real-time.

VII. Acknowledgements

The authors would like to thank Gianluca Antonelli for his advise in producing the final manuscript version as well as to the anonymous reviewers for their valuable comments.

References

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

[2] A. Santamaria-Navarro, V. Lippiello, and J. Andrade-Cetto, "Task priority control for aerial manipulation," in *2014 IEEE International Symposium on Safety, Security, and Rescue Robotics (2014)*, Hokkaido, Japan, Oct. 2014, pp. 1–6.

[3] B. Siciliano, "A closed-loop inverse kinematic scheme for on-line joint-based robot control*," *Robotica*, vol. 8, no. 3, pp. 231–243, July 1990.

[4] P. Chiacchio, S. Chiaverini, L. Sciavicco, and B. Siciliano, "Closed-loop inverse kinematics schemes for constrained redundant manipulators with task space augmentation and task priority strategy," *The International Journal of Robotics Research*, vol. 10, no. 4, pp. 410–425, Aug. 1991.

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

[6] B. Siciliano and J. J. E. Slotine, "A general framework for managing multiple tasks in highly redundant robotic systems," in , *5th International Conference on Advanced Robotics*, Pisa, Italy, June 1991, pp. 1211–1216 vol.2.

[7] 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, June 1997.

[8] P. Baerlocher and R. Boulic, "Task-priority formulations for the kinematic control of highly redundant articulated structures," in *IEEE/RSJ International Conference on Intelligent Robots and Systems.*, vol. 1, Victoria, Canada, Oct. 1998, pp. 323–329 vol.1.

[9] K. Baizid, G. Giglio, F. Pierri, M. A. Trujillo, G. Antonelli, F. Caccavale, A. Viguria, S. Chiaverini, and A. Ollero, "Behavioral control of unmanned aerial vehicle manipulator systems," *Autonomous Robots*, vol. 41, no. 5, pp. 1203–1220, June 2017.

[10] A. Santamaria-Navarro, P. Grosch, V. Lippiello, J. Sola, and J. Andrade-Cetto, "Uncalibrated visual servo for unmanned aerial manipulation," *IEEE/ASME Transactions on Mechatronics*, vol. 22, no. 4, pp. 1610–1621, Aug. 2017.

[11] G. Antonelli, "Stability analysis for prioritized closed-loop inverse kinematic algorithms for redundant robotic systems," *IEEE Transactions on Robotics*, vol. 25, no. 5, pp. 985–994, Oct. 2009.

[12] P. Falco and C. Natale, "On the stability of closed-loop inverse kinematics algorithms for redundant robots," *IEEE Transactions on Robotics*, vol. 27, no. 4, pp. 780–784, Aug. 2011.

[13] S. Moe, A. R. Teel, G. Antonelli, and K. Y. Pettersen, "Stability analysis for set-based control within the singularity-robust multiple task-priority inverse kinematics framework," in *54th IEEE Conference on Decision and Control (CDC)*, Osaka, Japan, Dec. 2015, pp. 171–178.

[14] H. Das, J. Slotine, and T. B. Sheridan, "Inverse kinematic algorithms for redundant systems," in *IEEE International Conference on Robotics and Automation*, Philadelphia, USA, Apr. 1988, pp. 43–48 vol.1.

[15] S. P. Boyd, Ed., *Linear matrix inequalities in system and control theory*, ser. SIAM studies in applied mathematics. Philadelphia: Society for Industrial and Applied Mathematics, 1994, no. vol. 15.

[16] S. Boyd and L. Vandenberghe, *Convex Optimization*. Cambridge University Press, Mar. 2004.

[17] H. Wolkowicz, R. Saigal, and L. Vandenberghe, Eds., *Handbook of Semidefinite Programming: Theory, Algorithms, and Applications*, ser. International Series in Operations Research & Management Science. Springer US, 2000.

[18] P. Corke, *Robotics, Vision and Control: Fundamental Algorithms In MATLAB*, 2nd ed., ser. Springer Tracts in Advanced Robotics. Springer International Publishing, 2017.

[19] J. F. Sturm, "Using SeDuMi 1.02, A Matlab toolbox for optimization over symmetric cones," *Optimization Methods and Software*, vol. 11, no. 1-4, pp. 625–653, Jan. 1999.