# Stability Analysis for Set-based Control within the Singularity-robust Multiple Task-priority Inverse Kinematics Framework

Signe Moe[1], Andrew R. Teel[2], Gianluca Antonelli[3] and Kristin Y. Pettersen[1]

***Abstract*— Inverse kinematics algorithms are commonly used in robotic systems to accomplish desired behavior, and several methods exist to ensure the achievement of several tasks simultaneously. The multiple task-priority inverse kinematics framework allows tasks to be considered in a prioritized order by projecting task velocities through the nullspaces of higher priority tasks. This paper extends this framework to handle set-based tasks, i.e. tasks with a range of valid values, in addition to equality tasks, which have a specific desired value. Examples of such tasks are joint limit and obstacle avoidance. The proposed method is proven to ensure asymptotic convergence of the equality task errors and the satisfaction of all high-priority set-based tasks. Simulations results confirm the effectiveness of the proposed approach.**

## I. INTRODUCTION

Robotic systems with a large number of Degrees of Freedom (DOFs) are commonly used for industrial purposes [1] and are becoming increasingly important within a variety of fields, including unmanned vehicles such as underwater [2], [3] and aerial [4] systems.

Traditionally, robotic systems are controlled in their joint space. However, the tasks they are required to perform are often given in the operational space, for instance given by the desired end effector position or orientation. As such, a variety of inverse kinematics and dynamics algorithms have been developed to map tasks from the operational space to the joint space and thus generate reference trajectories for the controllers. By limiting our attention to the kinematic level, the most common approach is to use a Jacobian-based method [5]-[7]. In particular, the pseudo-inverse Jacobian is defined for systems that are not square nor have full rank and is a widely used solution to the inverse kinematics problem [8]-[10]. Examples of various tasks and their corresponding kinematics and Jacobian matrices for the underwater case are given in [2].

A robotic system is said to be kinematically redundant if it possesses more DOFs than those required to perform a certain task [11]. In this case, the "excess" DOFs can be utilized in order to perform several tasks using Null-Space-Based (NSB) behavioral control, also known as multiple task-priority inverse kinematics [12]. This framework, which is described in more detail in Section II, has been developed for *equality tasks*. Equality tasks specify exactly one desired value for given states of the system, for instance the position and orientation of the end effector. However, for a general robotic system, several goals may not be described as equality tasks, but rather as *set-based tasks*, which are tasks that have a desired interval of values rather than one exact desired value. Examples of such tasks are staying within joint limits [13] and collision/obstacle avoidance [14]. As recognized in [15], the multiple task-priority inverse kinematics algorithm is not suitable to directly handle set-based tasks, and these tasks are therefore usually transformed into more restrictive equality constraints through potential fields or cost functions [16], [17].

An approach to systematically include set-based tasks in a prioritized task-regulation framework is proposed in [15] and further improved in [18]. To handle the set-based tasks, the algorithms in [15], [18] transform the inverse kinematics problem into a Quadratic Programming (QP) problem, and therefore they can not be utilized directly into the multiple task-priority inverse kinematics algorithm. In [19], set-based tasks are handled by resorting to proper activation and regularization functions. Furthermore, no analytical framework to prove the convergence/satisfaction of the tasks is provided. In [20], [21] set-based tasks are considered in a prioritized order and an algorithm is developed to ensure a smooth control law when (de)activating a set-based task. However, during transitions, the strict priority of the tasks is not respected.

The work in [22] and [23] aims to develop a method to include set-based tasks directly in the NSB framework by considering the set-based tasks within the given priority. Here, a set-based task is considered satisfied within an interval of valid values (for instance a joint $q$ with angle limits of $\pm 90^\circ$), and the goal is to prevent it from being violated while simultaneously fulfilling the equality tasks of the system. As such, a set-based task is defined as active within a certain region close to its limit (for instance $|q| \geq 85^\circ$). This is illustrated in Figure 1. An inactive set-based task can be ignored as the system is not close to its limits and thereby it is not in danger of being violated. However, when a set-based task is active, it is necessary to avoid that the velocities required by the other tasks push it towards the task limit. This is done by either 1) freezing the mentioned task at its current value, or 2) actively pushing the task away from its limit through feedback. The task is then deactivated once the other tasks ask for velocities that would push the set-based task away from its limit. Simulations validate the proposed method.

[1]S.Moe and K.Y.Pettersen are with the Center for Autonomous Marine Operations and Systems (AMOS), at The Department of Engineering Cybernetics, Norwegian University of Science and Technology (NTNU), Trondheim, Norway {signe.moe, kristin.y.pettersen}@itk.ntnu.no

[2]A.R.Teel is with the Department of Electrical Engineering, University of California Santa Barbara, Santa Barbara, USA teel@ece.ucsb.edu

[3]G.Antonelli is with the Department of Electrical and Information Engineering, University of Cassino and Southern Lazio, Cassino, Italy antonelli@unicas.it

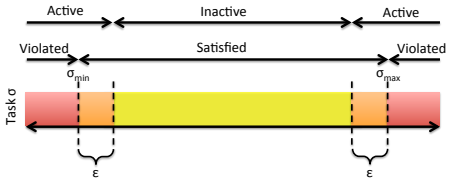



Fig. 1: Definition of main regions in set-based control for a task with an upper and lower bound. The task is considered active when it is within an interval $\varepsilon$ from the limits.

The contribution of this paper is to provide an analytical framework of convergence of equality task errors and satisfaction of set-based tasks based on the preliminary idea given in [22] and [23] described above. The proposed approach involves switching between several solutions, and the resulting closed-loop dynamic system can be described as a discontinuous differential equation. In order to investigate the stability properties of the system, this is first rewritten into a constrained differential inclusion and then the stability is investigated using Lyapunov analysis [24]. Sufficient conditions are provided such that the task errors of the equality tasks converge asymptotically to zero when including set-based tasks into the framework, and the higher-priority set-based tasks are satisfied at all times. For lower-priority set-based tasks, i.e. when one or more equality tasks have higher priority, it can not be guaranteed that the set-based tasks are not violated due to the influence of the higher-priority tasks. Thus, set-based tasks such as avoiding joint limits and obstacles should be made high-priority, whereas set-based tasks that are not crucial for operation can be made lower-priority. For compactness, the equations in this paper describe systems with high-priority set-based tasks.

This paper is organized as follows: Section II gives a brief description of the traditional multiple task-priority inverse kinematics algorithm used for equality tasks. Section III presents the new method for incorporating set-based tasks in the NSB framework. The stability proof is presented in Section IV, before simulation results validating the proposed algorithm are given in Section V. Conclusions are given in Section VI.

## II. SINGULARITY-ROBUST MULTIPLE TASK-PRIORITY INVERSE KINEMATICS

This section gives a brief presentation of the singularity-robust multiple task-priority inverse kinematics solution, also known as the NSB method, which is suitable for generating a reference trajectory for a general robotic system to fulfill equality tasks in a prioritized order. "Singularity-robust" is with respect to algorithmic singularities [25]. The method is presented for completeness, and in order to define concepts and notation that are needed to present the method proposed in this paper. For more details, the reader is referred to [12].

A general robotic system has $n$ DOFs. Its state is described by the joint values $\boldsymbol{q} = [q_1, q_2, ..., q_n]^T$. It is then possible to define tasks and task velocities in the operational space through forward kinematics and the task Jacobian matrix:

$$\boldsymbol{\sigma}(\boldsymbol{q}) = \boldsymbol{f}(\boldsymbol{q}) \tag{1}$$

$$\dot{\boldsymbol{\sigma}}(\boldsymbol{q}) = \frac{\delta \boldsymbol{f}(\boldsymbol{q})}{\delta \boldsymbol{q}}\dot{\boldsymbol{q}} = \boldsymbol{J}(\boldsymbol{q})\dot{\boldsymbol{q}}. \tag{2}$$

For compactness, the argument $\boldsymbol{q}$ of tasks, task errors, Jacobians and null-space matrices are omitted from the equations below. $\boldsymbol{J}$ is the configuration-dependent task Jacobian matrix and $\dot{\boldsymbol{q}}$ is the system joint velocities. To track a desired continuous, differentiable trajectory $\boldsymbol{\sigma}_{\mathrm{d}}(t)$, the corresponding desired joint angles $\dot{\boldsymbol{q}}_{\mathrm{d}}$ are computed using the Moore-Penrose pseudoinverse of the Jacobian.

$$\dot{\boldsymbol{q}}_{\mathrm{d}} = \boldsymbol{J}^{\dagger}(\dot{\boldsymbol{\sigma}}_{\mathrm{d}} + \boldsymbol{\Lambda}\tilde{\boldsymbol{\sigma}}). \tag{3}$$

$\tilde{\boldsymbol{\sigma}} \triangleq \boldsymbol{\sigma}_{\mathrm{d}} - \boldsymbol{\sigma}$ is the task error and $\boldsymbol{\Lambda}$ is a positive gain matrix. This feedback approach reduces the error dynamics to

$$\begin{aligned}\dot{\tilde{\boldsymbol{\sigma}}} &= \dot{\boldsymbol{\sigma}}_{\mathrm{d}} - \dot{\boldsymbol{\sigma}} = \dot{\boldsymbol{\sigma}}_{\mathrm{d}} - \boldsymbol{J}\dot{\boldsymbol{q}} \\ &= \dot{\boldsymbol{\sigma}}_{\mathrm{d}} - \boldsymbol{J}\boldsymbol{J}^{\dagger}(\dot{\boldsymbol{\sigma}}_{\mathrm{d}} + \boldsymbol{\Lambda}\tilde{\boldsymbol{\sigma}}) \\ &= -\boldsymbol{\Lambda}\tilde{\boldsymbol{\sigma}},\end{aligned} \tag{4}$$

if $\dot{\boldsymbol{q}} = \dot{\boldsymbol{q}}_{\mathrm{d}}$ and $\boldsymbol{J}$ has full rank, implying that $\boldsymbol{J}\boldsymbol{J}^{\dagger} = \boldsymbol{I}$. Equation (4) describes a linear system with an exponentially stable equilibrium point at the $\tilde{\boldsymbol{\sigma}} = \boldsymbol{0}$. It is worth noticing that the assumption $\dot{\boldsymbol{q}} = \dot{\boldsymbol{q}}_{\mathrm{d}}$ is common to all inverse kinematics algorithms. For practical applications, it requires that the low level dynamic control loop is faster than the kinematic one.

Kinematically redundant systems can execute more than a single task using the NSB method:

$$\dot{\boldsymbol{q}}_{\mathrm{d}} = \boldsymbol{J}^{\dagger}(\dot{\boldsymbol{\sigma}}_{\mathrm{d}} + \boldsymbol{\Lambda}\tilde{\boldsymbol{\sigma}}) + \underbrace{(\boldsymbol{I} - \boldsymbol{J}^{\dagger}\boldsymbol{J})}_{\triangleq \boldsymbol{N}}\dot{\boldsymbol{q}}_{\mathrm{d,sec}}. \tag{5}$$

$\boldsymbol{N}$ is the null-space of the original task $\boldsymbol{\sigma}$, and filters out joint velocities from a secondary task $\dot{\boldsymbol{q}}_{\mathrm{d,sec}}$ that would interfere with those of the first task. It is straightforward to show that $\boldsymbol{J}\boldsymbol{N} \equiv \boldsymbol{0}$. As such, multiple tasks can be arranged in a prioritized order to fulfill several goals at once. $\boldsymbol{\sigma}_k$ denotes task $k$ and $\boldsymbol{J}_k$ is the corresponding Jacobian of that task:

$$\dot{\boldsymbol{q}}_{1,\mathrm{d}} = \boldsymbol{J}_1^{\dagger}(\dot{\boldsymbol{\sigma}}_{1,\mathrm{d}} + \boldsymbol{\Lambda}_1\tilde{\boldsymbol{\sigma}}_1) \tag{6}$$

$$\vdots$$

$$\dot{\boldsymbol{q}}_{i,\mathrm{d}} = \boldsymbol{J}_i^{\dagger}(\dot{\boldsymbol{\sigma}}_{i,\mathrm{d}} + \boldsymbol{\Lambda}_i\tilde{\boldsymbol{\sigma}}_i) \tag{7}$$

$$\dot{\boldsymbol{q}}_{\mathrm{d}} = \dot{\boldsymbol{q}}_{1,\mathrm{d}} + \boldsymbol{N}_1\dot{\boldsymbol{q}}_{2,\mathrm{d}} + \boldsymbol{N}_{12}\dot{\boldsymbol{q}}_{3,\mathrm{d}} + ... + \boldsymbol{N}_{12..i-1}\dot{\boldsymbol{q}}_{i,\mathrm{d}}. \tag{8}$$

Here, $\dot{\boldsymbol{q}}_{k,\mathrm{d}}$ is the desired joint velocities for solving task $k$ alone. The total desired system velocity $\dot{\boldsymbol{q}}_{\mathrm{d}}$ is then found as the sum of the independent desired task velocities filtered through the null-spaces of the higher-priority tasks:

$$\boldsymbol{J}_{12..k} = \begin{bmatrix}\boldsymbol{J}_1 \\ \vdots \\ \boldsymbol{J}_k\end{bmatrix} \tag{9}$$

$$\boldsymbol{N}_{12..k} = \boldsymbol{I} - \boldsymbol{J}_{12..k}^{\dagger}\boldsymbol{J}_{12..k} \tag{10}$$

In theory it is possible to include an infinite number of tasks. However, a robotic system with finite DOFs will not be able to fulfill all of them. Once the DOFs of the system are "spent" on the highest priority tasks, the null-spaces will simply be reduced to the null-matrix and the lower-priority tasks will not be satisfied or influence the generated system trajectories.

## III. SET-BASED TASK CONTROL

The previous section introduced the concept of multiple task-priority inverse kinematic control for a robotic system as a method to generate reference trajectories for the system joints that, if fulfilled, will result in the successful achievement of several equality tasks. However, for a general robotic system, several goals might be described by *set-based tasks*. Set-based tasks can not be implemented directly using the method in the previous section, as this algorithm depends on the desired task velocity $\dot{\boldsymbol{\sigma}}_{\mathrm{d}}$ and the task error $\tilde{\boldsymbol{\sigma}}$, which are not defined when the desired task is a set rather than a value. This section presents a method to handle scalar set-based tasks in the multiple task-priority inverse kinematics framework.

The method proposed in this paper only allows set-based tasks to remain in the closed set $D = [\sigma_{\min} + \varepsilon, \sigma_{\max} - \varepsilon]$ for some $\varepsilon > 0$ (the yellow set in Figure 1). Choosing $\varepsilon > 0$ ensures robustness for practical applications. The proposed algorithm considers only the system's equality tasks as long as the resulting solution stays within this desired set. If this is not the case, the set-based task is frozen on the border of $D$ ensuring that the set-based task is not violated. The task is then then unfrozen once the resulting solution of considering only the equality tasks will make the system stay in $D$ once again.

From here on, equality tasks are denoted with number subscripts and set-based tasks with letters, e.g. $\boldsymbol{\sigma}_1$ and $\sigma_{\mathrm{a}}$. Furthermore, only regulation equality tasks are considered, that is tasks to guide the system to a stationary value ($\dot{\boldsymbol{\sigma}}_{\mathrm{d}} \equiv 0$). Finally, it is assumed that the desired joint velocities $\dot{\boldsymbol{q}}_{\mathrm{d}}$ are tracked perfectly by the system, so $\dot{\boldsymbol{q}} = \dot{\boldsymbol{q}}_{\mathrm{d}}$. This is a common assumption in Closed Loop Inverse Kinematics [26].

### A. One set-based task, k equality tasks

For simplicity we first consider a robotic system with $n$ DOFs and $k$ equality tasks of $m_i$ DOFs for $i \in \{1, \ldots, k\}$. Task $i$ is denoted $\boldsymbol{\sigma}_i$, and $\tilde{\boldsymbol{\sigma}}_i \triangleq \boldsymbol{\sigma}_{i,\mathrm{d}} - \boldsymbol{\sigma}_i$. Furthermore, the system has a set-based task $\sigma_{\mathrm{a}} \in \mathbb{R}$ of avoiding a circular obstacle at a constant position $\boldsymbol{p}_{\mathrm{o}}$ with radius $r > 0$. The task is defined as the distance between the end effector and the obstacle center. The kinematics and Jacobian are given in [2]:

$$\sigma_{\mathrm{a}} = \sqrt{(\boldsymbol{p}_{\mathrm{o}} - \boldsymbol{p}_{\mathrm{e}})^T (\boldsymbol{p}_{\mathrm{o}} - \boldsymbol{p}_{\mathrm{e}})} \tag{11}$$

$$\dot{\sigma}_{\mathrm{a}} = \boldsymbol{J}_{\mathrm{a}} \dot{\boldsymbol{q}} = -\frac{(\boldsymbol{p}_{\mathrm{o}} - \boldsymbol{p}_{\mathrm{e}})^T}{||\boldsymbol{p}_{\mathrm{o}} - \boldsymbol{p}_{\mathrm{e}}||} \boldsymbol{J} \dot{\boldsymbol{q}}. \tag{12}$$

Here, $\boldsymbol{p}_{\mathrm{e}}$ denotes the position of the end effector and $\boldsymbol{J}$ is the corresponding position Jacobian. Consider the state-vector $\boldsymbol{z} \in \mathbb{R}^l$ and the closed set $C$, where

$$l = n + 1 + \sum_{i=1}^{k} m_i, \tag{13}$$

and

$$\boldsymbol{z} = \begin{bmatrix} \boldsymbol{q} \\ \boldsymbol{\sigma}_{\mathrm{sb}} \\ \tilde{\boldsymbol{\sigma}}_{\mathrm{eb}} \end{bmatrix} = \begin{bmatrix} q_1 \\ \vdots \\ q_n \\ \sigma_{\mathrm{a}} \\ \tilde{\boldsymbol{\sigma}}_1 \\ \vdots \\ \tilde{\boldsymbol{\sigma}}_k \end{bmatrix} \tag{14}$$

$$C := \mathbb{R}^n \times [\varepsilon, \infty) \times \mathbb{R}^{l-n-1} \tag{15}$$

for an $\varepsilon > r$ and $\boldsymbol{q} \in \mathbb{D}$, where $\mathbb{D} = \{\boldsymbol{q} \mid \sigma_{\mathrm{a}}(\boldsymbol{q}) \in \mathbb{R} \setminus \{0\}\}$. In $C$, the distance between the end effector and the obstacle center is equal to or greater than $\varepsilon$, which in turn is greater than the obstacle radius $r$. Thus, the set-based task is always satisfied for $\boldsymbol{z} \in C$. $\boldsymbol{\sigma}_{\mathrm{sb}}$ and $\tilde{\boldsymbol{\sigma}}_{\mathrm{eb}}$ are vectors containing the set-based tasks and the equality task errors of the system, respectively.

For a system with one set-based task, two modes must be considered:

1) Ignoring the set-based task and consider only the equality tasks.
2) Freezing the set-based task as first priority and consider the equality tasks as second priority.

Mode 1 is the "default" solution, whereas mode 2 should be activated only when it is necessary to prevent the set-based task from being violated. Mode 1 results in the following system:

$$\dot{\boldsymbol{q}} = \boldsymbol{J}_1^{\dagger} \boldsymbol{\Lambda}_1 \tilde{\boldsymbol{\sigma}}_1 + \boldsymbol{N}_1 \boldsymbol{J}_2^{\dagger} \boldsymbol{\Lambda}_2 \tilde{\boldsymbol{\sigma}}_2 + \boldsymbol{N}_{12} \boldsymbol{J}_3^{\dagger} \boldsymbol{\Lambda}_3 \tilde{\boldsymbol{\sigma}}_3 + \ldots + \boldsymbol{N}_{12..(k-1)} \boldsymbol{J}_k^{\dagger} \boldsymbol{\Lambda}_k \tilde{\boldsymbol{\sigma}}_k \tag{16}$$

$$\Downarrow$$

$$\dot{\sigma}_{\mathrm{a}} = \boldsymbol{J}_{\mathrm{a}} \dot{\boldsymbol{q}} = \boldsymbol{J}_{\mathrm{a}} (\boldsymbol{J}_1^{\dagger} \boldsymbol{\Lambda}_1 \tilde{\boldsymbol{\sigma}}_1 + \ldots + \boldsymbol{N}_{12..(k-1)} \boldsymbol{J}_k^{\dagger} \boldsymbol{\Lambda}_k \tilde{\boldsymbol{\sigma}}_k) \tag{17}$$

$$\begin{aligned}
\dot{\tilde{\boldsymbol{\sigma}}}_{\mathrm{eb}} = -\dot{\boldsymbol{\sigma}}_{\mathrm{eb}} &= -\begin{bmatrix} \dot{\boldsymbol{\sigma}}_1 \\ \dot{\boldsymbol{\sigma}}_2 \\ \vdots \\ \dot{\boldsymbol{\sigma}}_k \end{bmatrix} = -\begin{bmatrix} \boldsymbol{J}_1 \\ \boldsymbol{J}_2 \\ \vdots \\ \boldsymbol{J}_k \end{bmatrix} \dot{\boldsymbol{q}} \\
&= -\begin{bmatrix} \boldsymbol{J}_1 (\boldsymbol{J}_1^{\dagger} \boldsymbol{\Lambda}_1 \tilde{\boldsymbol{\sigma}}_1 + \boldsymbol{N}_1 \boldsymbol{J}_2^{\dagger} \boldsymbol{\Lambda}_2 \tilde{\boldsymbol{\sigma}}_2 + \ldots + \boldsymbol{N}_{12..(k-1)} \boldsymbol{J}_k^{\dagger} \boldsymbol{\Lambda}_k \tilde{\boldsymbol{\sigma}}_k) \\ \boldsymbol{J}_2 (\boldsymbol{J}_1^{\dagger} \boldsymbol{\Lambda}_1 \tilde{\boldsymbol{\sigma}}_1 + \boldsymbol{N}_1 \boldsymbol{J}_2^{\dagger} \boldsymbol{\Lambda}_2 \tilde{\boldsymbol{\sigma}}_2 + \ldots + \boldsymbol{N}_{12..(k-1)} \boldsymbol{J}_k^{\dagger} \boldsymbol{\Lambda}_k \tilde{\boldsymbol{\sigma}}_k) \\ \vdots \\ \boldsymbol{J}_k (\boldsymbol{J}_1^{\dagger} \boldsymbol{\Lambda}_1 \tilde{\boldsymbol{\sigma}}_1 + \boldsymbol{N}_1 \boldsymbol{J}_2^{\dagger} \boldsymbol{\Lambda}_2 \tilde{\boldsymbol{\sigma}}_2 + \ldots + \boldsymbol{N}_{12..(k-1)} \boldsymbol{J}_k^{\dagger} \boldsymbol{\Lambda}_k \tilde{\boldsymbol{\sigma}}_k) \end{bmatrix} \\
&= -\begin{bmatrix} \boldsymbol{\Lambda}_1 \tilde{\boldsymbol{\sigma}}_1 \\ \boldsymbol{J}_2 \boldsymbol{J}_1^{\dagger} \boldsymbol{\Lambda}_1 \tilde{\boldsymbol{\sigma}}_1 + \boldsymbol{J}_2 \boldsymbol{N}_1 \boldsymbol{J}_2^{\dagger} \boldsymbol{\Lambda}_2 \tilde{\boldsymbol{\sigma}}_2 \\ \vdots \\ \boldsymbol{J}_k \boldsymbol{J}_1^{\dagger} \boldsymbol{\Lambda}_1 \tilde{\boldsymbol{\sigma}}_1 + \boldsymbol{J}_k \boldsymbol{N}_1 \boldsymbol{J}_2^{\dagger} \boldsymbol{\Lambda}_2 \tilde{\boldsymbol{\sigma}}_2 + \ldots + \boldsymbol{J}_k \boldsymbol{N}_{12..(k-1)} \boldsymbol{J}_k^{\dagger} \boldsymbol{\Lambda}_k \tilde{\boldsymbol{\sigma}}_k \end{bmatrix} \\
&= -\begin{bmatrix} \boldsymbol{\Lambda}_1 & \boldsymbol{0}_{m_1 \times m_2} & \cdots & \boldsymbol{0}_{m_1 \times m_k} \\ \boldsymbol{J}_2 \boldsymbol{J}_1^{\dagger} \boldsymbol{\Lambda}_1 & \boldsymbol{J}_2 \boldsymbol{N}_1 \boldsymbol{J}_2^{\dagger} \boldsymbol{\Lambda}_2 & \cdots & \boldsymbol{0}_{m_2 \times m_k} \\ & & \ddots & \\ \vdots & \vdots & & \vdots \\ \boldsymbol{J}_k \boldsymbol{J}_1^{\dagger} \boldsymbol{\Lambda}_1 & \boldsymbol{J}_k \boldsymbol{N}_1 \boldsymbol{J}_2^{\dagger} \boldsymbol{\Lambda}_2 & \cdots & \boldsymbol{J}_k \boldsymbol{N}_{12..(k-1)} \boldsymbol{J}_k^{\dagger} \boldsymbol{\Lambda}_k \end{bmatrix} \begin{bmatrix} \tilde{\boldsymbol{\sigma}}_1 \\ \tilde{\boldsymbol{\sigma}}_2 \\ \vdots \\ \tilde{\boldsymbol{\sigma}}_k \end{bmatrix} \\
&= -\boldsymbol{M}_1 \tilde{\boldsymbol{\sigma}}_{\mathrm{eb}}
\end{aligned} \tag{18}$$

$$\dot{\boldsymbol{z}} = \begin{bmatrix} \boldsymbol{J}_1^{\dagger} \boldsymbol{\Lambda}_1 \tilde{\boldsymbol{\sigma}}_1 + \ldots + \boldsymbol{N}_{12..(k-1)} \boldsymbol{J}_k^{\dagger} \boldsymbol{\Lambda}_k \tilde{\boldsymbol{\sigma}}_k \\ \boldsymbol{J}_{\mathrm{a}} (\boldsymbol{J}_1^{\dagger} \boldsymbol{\Lambda}_1 \tilde{\boldsymbol{\sigma}}_1 + \ldots + \boldsymbol{N}_{12..(k-1)} \boldsymbol{J}_k^{\dagger} \boldsymbol{\Lambda}_k \tilde{\boldsymbol{\sigma}}_k) \\ -\boldsymbol{M}_1 \tilde{\boldsymbol{\sigma}}_{\mathrm{eb}} \end{bmatrix} \triangleq \begin{bmatrix} \boldsymbol{f}_{11}(\boldsymbol{z}) \\ f_{12}(\boldsymbol{z}) \\ \boldsymbol{f}_{13}(\boldsymbol{z}) \end{bmatrix} = \boldsymbol{f}_1(\boldsymbol{z}). \tag{19}$$

The matrix $\boldsymbol{M}_1$ is positive definite given certain assumptions: when an additional task is considered, the task Jacobian must be independent with respect to the Jacobian obtained by stacking all the higher priority tasks, and a proper choice of the gains is required (for details, see [26]).

The time evolution of $\boldsymbol{z}$ follows the vector field $\boldsymbol{f}_1(\boldsymbol{z})$ as long as the solution $\boldsymbol{z}$ stays in $C$. In mode 1, the set-based task evolves freely, $\dot{\sigma}_{\mathrm{a}} = f_{12}(\boldsymbol{z})$. If following $\boldsymbol{f}_1(\boldsymbol{z})$ would result

in $\boldsymbol{z}$ leaving the set $C$, avoiding the obstacle is considered the first priority task and mode 2 is activated. This can only occur on the border of $C$, that is $\sigma_{\mathrm{a}} = \varepsilon$, and $\dot{\sigma}_{\mathrm{a}} = f_{12}(\boldsymbol{z}) < 0$. To avoid the obstacle, an equality task with the goal of keeping the current distance to the obstacle is added as the highest priority task. In this case, the desired task value $\sigma_{a,\mathrm{d}}$ is equal to the current task value $\sigma_{\mathrm{a}} = \varepsilon$. Thus, the task error $\tilde{\sigma}_{\mathrm{a}} = \sigma_{a,\mathrm{d}} - \sigma_{\mathrm{a}} \equiv 0$. The system is then defined by the following equations:

$$\begin{aligned}\dot{\boldsymbol{q}} &= \boldsymbol{J}_{\mathrm{a}}^{\dagger}\dot{\tilde{\sigma}}_{\mathrm{a}} + \boldsymbol{N}_{\mathrm{a}}\boldsymbol{J}_1^{\dagger}\boldsymbol{\Lambda}_1\tilde{\boldsymbol{\sigma}}_1 + \boldsymbol{N}_{\mathrm{a}1}\boldsymbol{J}_2^{\dagger}\boldsymbol{\Lambda}_2\tilde{\boldsymbol{\sigma}}_2 + \boldsymbol{N}_{\mathrm{a}12}\boldsymbol{J}_3^{\dagger}\boldsymbol{\Lambda}_3\tilde{\boldsymbol{\sigma}}_3 \\ &\quad + \ldots + \boldsymbol{N}_{\mathrm{a}12..(k-1)}\boldsymbol{J}_k^{\dagger}\boldsymbol{\Lambda}_k\tilde{\boldsymbol{\sigma}}_k \\ &= \boldsymbol{N}_{\mathrm{a}}\boldsymbol{J}_1^{\dagger}\boldsymbol{\Lambda}_1\tilde{\boldsymbol{\sigma}}_1 + \boldsymbol{N}_{\mathrm{a}1}\boldsymbol{J}_2^{\dagger}\boldsymbol{\Lambda}_2\tilde{\boldsymbol{\sigma}}_2 + \ldots + \boldsymbol{N}_{\mathrm{a}12..(k-1)}\boldsymbol{J}_k^{\dagger}\boldsymbol{\Lambda}_k\tilde{\boldsymbol{\sigma}}_k \qquad (20)\end{aligned}$$

$$\Downarrow$$

$$\dot{\sigma}_{\mathrm{a}} = \boldsymbol{J}_{\mathrm{a}}\dot{\boldsymbol{q}} = \boldsymbol{J}_{\mathrm{a}}(\boldsymbol{N}_{\mathrm{a}}\boldsymbol{J}_1^{\dagger}\boldsymbol{\Lambda}_1\tilde{\boldsymbol{\sigma}}_1 \ldots + \boldsymbol{N}_{\mathrm{a}12..(k-1)}\boldsymbol{J}_k^{\dagger}\boldsymbol{\Lambda}_k\tilde{\boldsymbol{\sigma}}_k) = 0 \qquad (21)$$

$$\begin{aligned}\dot{\tilde{\boldsymbol{\sigma}}}_{\mathrm{eb}} &= -\dot{\boldsymbol{\sigma}}_{\mathrm{eb}} = -\begin{bmatrix}\dot{\boldsymbol{\sigma}}_1 \\ \dot{\boldsymbol{\sigma}}_2 \\ \vdots \\ \dot{\boldsymbol{\sigma}}_k\end{bmatrix} = -\begin{bmatrix}\boldsymbol{J}_1 \\ \boldsymbol{J}_2 \\ \vdots \\ \boldsymbol{J}_k\end{bmatrix}\dot{\boldsymbol{q}} \\ &= -\begin{bmatrix}\boldsymbol{J}_1(\boldsymbol{N}_{\mathrm{a}}\boldsymbol{J}_1^{\dagger}\boldsymbol{\Lambda}_1\tilde{\boldsymbol{\sigma}}_1 + \boldsymbol{N}_{\mathrm{a}1}\boldsymbol{J}_2^{\dagger}\boldsymbol{\Lambda}_2\tilde{\boldsymbol{\sigma}}_2 + \ldots + \boldsymbol{N}_{\mathrm{a}12..(k-1)}\boldsymbol{J}_k^{\dagger}\boldsymbol{\Lambda}_k\tilde{\boldsymbol{\sigma}}_k) \\ \boldsymbol{J}_2(\boldsymbol{N}_{\mathrm{a}}\boldsymbol{J}_1^{\dagger}\boldsymbol{\Lambda}_1\tilde{\boldsymbol{\sigma}}_1 + \boldsymbol{N}_{\mathrm{a}1}\boldsymbol{J}_2^{\dagger}\boldsymbol{\Lambda}_2\tilde{\boldsymbol{\sigma}}_2 + \ldots + \boldsymbol{N}_{\mathrm{a}12..(k-1)}\boldsymbol{J}_k^{\dagger}\boldsymbol{\Lambda}_k\tilde{\boldsymbol{\sigma}}_k) \\ \vdots \\ \boldsymbol{J}_k(\boldsymbol{N}_{\mathrm{a}}\boldsymbol{J}_1^{\dagger}\boldsymbol{\Lambda}_1\tilde{\boldsymbol{\sigma}}_1 + \boldsymbol{N}_{\mathrm{a}1}\boldsymbol{J}_2^{\dagger}\boldsymbol{\Lambda}_2\tilde{\boldsymbol{\sigma}}_2 + \ldots + \boldsymbol{N}_{\mathrm{a}12..(k-1)}\boldsymbol{J}_k^{\dagger}\boldsymbol{\Lambda}_k\tilde{\boldsymbol{\sigma}}_k)\end{bmatrix} \\ &= -\begin{bmatrix}\boldsymbol{J}_1\boldsymbol{N}_{\mathrm{a}}\boldsymbol{J}_1^{\dagger}\boldsymbol{\Lambda}_1\tilde{\boldsymbol{\sigma}}_1 \\ \boldsymbol{J}_2\boldsymbol{N}_{\mathrm{a}}\boldsymbol{J}_1^{\dagger}\boldsymbol{\Lambda}_1\tilde{\boldsymbol{\sigma}}_1 + \boldsymbol{J}_2\boldsymbol{N}_{\mathrm{a}1}\boldsymbol{J}_2^{\dagger}\boldsymbol{\Lambda}_2\tilde{\boldsymbol{\sigma}}_2 \\ \vdots \\ \boldsymbol{J}_k\boldsymbol{N}_{\mathrm{a}}\boldsymbol{J}_1^{\dagger}\boldsymbol{\Lambda}_1\tilde{\boldsymbol{\sigma}}_1 + \boldsymbol{J}_k\boldsymbol{N}_{\mathrm{a}1}\boldsymbol{J}_2^{\dagger}\boldsymbol{\Lambda}_2\tilde{\boldsymbol{\sigma}}_2 + \ldots + \boldsymbol{J}_k\boldsymbol{N}_{\mathrm{a}12..(k-1)}\boldsymbol{J}_k^{\dagger}\boldsymbol{\Lambda}_k\tilde{\boldsymbol{\sigma}}_k\end{bmatrix} \\ &= -\begin{bmatrix}\boldsymbol{J}_1\boldsymbol{N}_{\mathrm{a}}\boldsymbol{J}_1^{\dagger}\boldsymbol{\Lambda}_1 & \boldsymbol{0}_{m_1\times m_2} & \cdots & \boldsymbol{0}_{m_1\times m_k} \\ \boldsymbol{J}_2\boldsymbol{N}_{\mathrm{a}}\boldsymbol{J}_1^{\dagger}\boldsymbol{\Lambda}_1 & \boldsymbol{J}_2\boldsymbol{N}_{\mathrm{a}1}\boldsymbol{J}_2^{\dagger}\boldsymbol{\Lambda}_2 & \cdots & \boldsymbol{0}_{m_2\times m_k} \\ \vdots & \vdots & \ddots & \vdots \\ \boldsymbol{J}_k\boldsymbol{N}_{\mathrm{a}}\boldsymbol{J}_1^{\dagger}\boldsymbol{\Lambda}_1 & \boldsymbol{J}_k\boldsymbol{N}_{\mathrm{a}1}\boldsymbol{J}_2^{\dagger}\boldsymbol{\Lambda}_2 & \cdots & \boldsymbol{J}_k\boldsymbol{N}_{\mathrm{a}12..(k-1)}\boldsymbol{J}_k^{\dagger}\boldsymbol{\Lambda}_k\end{bmatrix}\begin{bmatrix}\tilde{\boldsymbol{\sigma}}_1 \\ \tilde{\boldsymbol{\sigma}}_2 \\ \vdots \\ \tilde{\boldsymbol{\sigma}}_k\end{bmatrix} \\ &= -\boldsymbol{M}_2\tilde{\boldsymbol{\sigma}}_{\mathrm{eb}} \qquad (22)\end{aligned}$$

$$\dot{\boldsymbol{z}} = \begin{bmatrix}\boldsymbol{N}_{\mathrm{a}}\boldsymbol{J}_1^{\dagger}\boldsymbol{\Lambda}_1\tilde{\boldsymbol{\sigma}}_1 \ldots + \boldsymbol{N}_{\mathrm{a}12..(k-1)}\boldsymbol{J}_k^{\dagger}\boldsymbol{\Lambda}_k\tilde{\boldsymbol{\sigma}}_k \\ \boldsymbol{J}_{\mathrm{a}}(\boldsymbol{N}_{\mathrm{a}}\boldsymbol{J}_1^{\dagger}\boldsymbol{\Lambda}_1\tilde{\boldsymbol{\sigma}}_1 \ldots + \boldsymbol{N}_{\mathrm{a}12..(k-1)}\boldsymbol{J}_k^{\dagger}\boldsymbol{\Lambda}_k\tilde{\boldsymbol{\sigma}}_k) \\ -\boldsymbol{M}_2\tilde{\boldsymbol{\sigma}}_{\mathrm{eb}}\end{bmatrix} \triangleq \begin{bmatrix}f_{21}(\boldsymbol{z}) \\ 0 \\ f_{23}(\boldsymbol{z})\end{bmatrix} = \boldsymbol{f}_2(\boldsymbol{z}). \qquad (23)$$

The matrix $\boldsymbol{M}_2$ can be seen as a principal submatrix of the general matrix $\boldsymbol{M}$ in Equation (60) in [26], which is shown to be positive definite given the same assumptions as for $\boldsymbol{M}_1$. Thus, $\boldsymbol{M}_2$ is also positive definite. Furthermore, as can be seen in (21), the solution (20) ensures that the distance to the obstacle is kept constant ($\dot{\sigma}_{\mathrm{a}} = 0$).

Let $\boldsymbol{T}_C(\boldsymbol{z})$ denote the tangent cone to $C$ at the point $\boldsymbol{z} \in C$ and define the set $P$ as the interior of $C$.

$$\boldsymbol{T}_C(\boldsymbol{z}) = \begin{cases}\mathbb{R}^l & \boldsymbol{z} \in P \\ \mathbb{R}^n \times [\,0, \infty\,) \times \mathbb{R}^{l-n-1} & \boldsymbol{z} = C \backslash P\end{cases} \qquad (24)$$

Consider the continuous functions $\boldsymbol{f}_1, \boldsymbol{f}_2 : C \to \mathbb{R}^l$ as defined in (19) and (23). $\dot{\boldsymbol{z}}$ follows the vector field $\boldsymbol{f}_1(\boldsymbol{z})$ as long as the solution $\boldsymbol{z}$ stays in $C$ (mode 1). Using Lemma 5.26 [24] on the system $\dot{\boldsymbol{z}} = \boldsymbol{f}_1(\boldsymbol{z})$, we know that such a solution exists when $\boldsymbol{f}_1(\boldsymbol{x}) \cap \boldsymbol{T}_C(\boldsymbol{x}) \neq \emptyset$ for all $\boldsymbol{x}$ near $\boldsymbol{z}$ (restricting $\boldsymbol{x}$ to $C$). Hence, we define

$$S := \{\boldsymbol{z} \in C : \exists \text{ a neighborhood U of } \boldsymbol{z} : \boldsymbol{f}_1(\boldsymbol{x}) \in \boldsymbol{T}_C(\boldsymbol{x}) \forall \boldsymbol{x} \in C \cap U\}. \qquad (25)$$

The discontinuous function $\boldsymbol{f} : C \to \mathbb{R}^l$

$$\boldsymbol{f}(\boldsymbol{z}) := \begin{cases}\boldsymbol{f}_1(\boldsymbol{z}) & \boldsymbol{z} \in S \\ \boldsymbol{f}_2(\boldsymbol{z}) & \boldsymbol{z} \in C \backslash S\end{cases} \qquad (26)$$

then describes our system. The differential equation $\dot{\boldsymbol{z}} = \boldsymbol{f}(\boldsymbol{z})$ then corresponds to following $\boldsymbol{f}_1$ (mode 1) as long as $\boldsymbol{z}$ stays in $C$ and following $\boldsymbol{f}_2$ (mode 2) otherwise. In mode 2, the set-based task is frozen, so the $(n+1)$th element in $\boldsymbol{f}_2(\boldsymbol{z}) \equiv 0$. Consequently, $\boldsymbol{f}_2(\boldsymbol{z}) \in \boldsymbol{T}_C(\boldsymbol{z}) \forall \boldsymbol{z} \in C$. This implies that $C$ is strongly forward invariant for $\dot{\boldsymbol{z}} = \boldsymbol{f}_2(\boldsymbol{z})$, so $\boldsymbol{z}(t_0) \in C \Rightarrow \boldsymbol{z}(t) \in C \ \forall \ t \geq t_0$.

More specifically, the set $S$ contains the points $\boldsymbol{z}$ in $C$ such that $\boldsymbol{f}_1(\boldsymbol{x}) \in \boldsymbol{T}_C(\boldsymbol{x})$ for $\boldsymbol{x}$ in $C$ that are near $\boldsymbol{z}$. At the border of $C$, $\sigma_{\mathrm{a}} = \varepsilon$. As such, the $(n+1)$th element of $\boldsymbol{f}_1(\boldsymbol{z})$ must be zero or positive for $\boldsymbol{z}$ to stay in $S$. If it is not, mode 2 is activated, which freezes the distance to the obstacle at the border of $C$. This remains the solution until following $\boldsymbol{f}_1(\boldsymbol{z})$ will result in $\dot{\sigma}_{\mathrm{a}} \geq 0$, i.e. the distance between the end effector and obstacle will remain constant or increase. The following pseudocode illustrates the system:

```
IF in the interior of C (sigma_a > epsilon)
     q_dot = f11;
ELSE on the border of C (sigma_a == epsilon)
     IF f12 >= 0 (eq. 18)
          q_dot = f11;
     ELSE
          q_dot = f21;
     END
END
```

### *B. Two set-based tasks, k equality tasks*

Consider a $n$ DOF manipulator with an upper and lower limit $q_{1,\max}$ and $q_{1,\min}$ on joint 1 and $q_{2,\max}$ and $q_{2,\min}$ on joint 2. The manipulator is tasked with $k$ equality tasks of $m_i$ DOFs for $i \in \{1, \ldots, k\}$. Task $i$ is denoted $\boldsymbol{\sigma}_i$, and $\tilde{\boldsymbol{\sigma}}_i \triangleq \boldsymbol{\sigma}_{i,\mathrm{d}} - \boldsymbol{\sigma}_i$. The set-based tasks $\sigma_{\mathrm{a}} = q_1 \in \mathbb{R}$ and $\sigma_{\mathrm{b}} = q_2 \in \mathbb{R}$ aim to avoid the joint limits. Consider the state-vector $\boldsymbol{z} \in \mathbb{R}^l$, where

$$l = n + 2 + \sum_{i=1}^{k} m_i \qquad (27)$$

and

$$\boldsymbol{z} = \begin{bmatrix}\boldsymbol{q} \\ \boldsymbol{\sigma}_{\mathrm{sb}} \\ \tilde{\boldsymbol{\sigma}}_{\mathrm{eb}}\end{bmatrix} = \begin{bmatrix}q_1 \\ \vdots \\ q_n \\ \sigma_{\mathrm{a}} \\ \sigma_{\mathrm{b}} \\ \tilde{\boldsymbol{\sigma}}_1 \\ \vdots \\ \tilde{\boldsymbol{\sigma}}_k\end{bmatrix}. \qquad (28)$$

In the case of 2 set-based tasks, $2^2 = 4$ different modes are considered:

1) Ignoring the set-based tasks and consider only the equality tasks
2) Freezing joint 2
3) Freezing joint 1
4) Freezing both joint 1 and 2

Similar to the previous section, mode 1 is the default mode, whereas modes 2-4 are activated only when it is necessary to prevent the system from violating the set-based task limits. In mode 1, only the equality tasks are considered:

$$\begin{aligned}\dot{\boldsymbol{q}} &= \boldsymbol{f}_{11}(\boldsymbol{q}, \tilde{\boldsymbol{\sigma}}) \\ &= \boldsymbol{J}_1^{\dagger}\boldsymbol{\Lambda}_1\tilde{\boldsymbol{\sigma}}_1 + \boldsymbol{N}_1\boldsymbol{J}_2^{\dagger}\boldsymbol{\Lambda}_2\tilde{\boldsymbol{\sigma}}_2 + \boldsymbol{N}_{12}\boldsymbol{J}_3^{\dagger}\boldsymbol{\Lambda}_3\tilde{\boldsymbol{\sigma}}_3 + \ldots + \boldsymbol{N}_{12..(k-1)}\boldsymbol{J}_k^{\dagger}\boldsymbol{\Lambda}_k\tilde{\boldsymbol{\sigma}}_k. \qquad (29)\end{aligned}$$

In mode 2, joint 2 is frozen at its current value, so $\sigma_{b,\mathrm{d}} \equiv \sigma_\mathrm{b}$ and consequently, $\tilde{\sigma}_\mathrm{b} \equiv 0$. This is then the first priority task, followed by the equality tasks.

$$\begin{aligned}\dot{\boldsymbol{q}} &= \boldsymbol{f}_{21}(\boldsymbol{q},\tilde{\boldsymbol{\sigma}}) \\ &= \boldsymbol{J}_\mathrm{b}^\dagger \underbrace{\tilde{\sigma}_\mathrm{b}}_{\equiv 0} + \boldsymbol{N}_\mathrm{b}\boldsymbol{J}_1^\dagger\boldsymbol{\Lambda}_1\tilde{\boldsymbol{\sigma}}_1 + \boldsymbol{N}_{\mathrm{b}1}\boldsymbol{J}_2^\dagger\boldsymbol{\Lambda}_2\tilde{\boldsymbol{\sigma}}_2 + \boldsymbol{N}_{\mathrm{b}12}\boldsymbol{J}_3^\dagger\boldsymbol{\Lambda}_3\tilde{\boldsymbol{\sigma}}_3 \\ &\quad + \ldots + \boldsymbol{N}_{\mathrm{b}12..(k-1)}\boldsymbol{J}_k^\dagger\boldsymbol{\Lambda}_k\tilde{\boldsymbol{\sigma}}_k \\ &= \boldsymbol{N}_\mathrm{b}\boldsymbol{J}_1^\dagger\boldsymbol{\Lambda}_1\tilde{\boldsymbol{\sigma}}_1 + \boldsymbol{N}_{\mathrm{b}1}\boldsymbol{J}_2^\dagger\boldsymbol{\Lambda}_2\tilde{\boldsymbol{\sigma}}_2 + \ldots + \boldsymbol{N}_{\mathrm{b}12..(k-1)}\boldsymbol{J}_k^\dagger\boldsymbol{\Lambda}_k\tilde{\boldsymbol{\sigma}}_k.\end{aligned} \tag{30}$$

Similarly, in mode 3 joint 1 is frozen at its current value:

$$\begin{aligned}\dot{\boldsymbol{q}} &= \boldsymbol{f}_{31}(\boldsymbol{q},\tilde{\boldsymbol{\sigma}}) \\ &= \boldsymbol{J}_\mathrm{a}^\dagger \underbrace{\tilde{\sigma}_\mathrm{a}}_{\equiv 0} + \boldsymbol{N}_\mathrm{a}\boldsymbol{J}_1^\dagger\boldsymbol{\Lambda}_1\tilde{\boldsymbol{\sigma}}_1 + \boldsymbol{N}_{\mathrm{a}1}\boldsymbol{J}_2^\dagger\boldsymbol{\Lambda}_2\tilde{\boldsymbol{\sigma}}_2 + \boldsymbol{N}_{\mathrm{a}12}\boldsymbol{J}_3^\dagger\boldsymbol{\Lambda}_3\tilde{\boldsymbol{\sigma}}_3 \\ &\quad + \ldots + \boldsymbol{N}_{\mathrm{a}12..(k-1)}\boldsymbol{J}_k^\dagger\boldsymbol{\Lambda}_k\tilde{\boldsymbol{\sigma}}_k \\ &= \boldsymbol{N}_\mathrm{a}\boldsymbol{J}_1^\dagger\boldsymbol{\Lambda}_1\tilde{\boldsymbol{\sigma}}_1 + \boldsymbol{N}_{\mathrm{a}1}\boldsymbol{J}_2^\dagger\boldsymbol{\Lambda}_2\tilde{\boldsymbol{\sigma}}_2 + \ldots + \boldsymbol{N}_{\mathrm{a}12..(k-1)}\boldsymbol{J}_k^\dagger\boldsymbol{\Lambda}_k\tilde{\boldsymbol{\sigma}}_k.\end{aligned} \tag{31}$$

Finally, in mode 4, both joints are frozen:

$$\begin{aligned}\dot{\boldsymbol{q}} &= \boldsymbol{f}_{41}(\boldsymbol{q},\tilde{\boldsymbol{\sigma}}) \\ &= \boldsymbol{J}_\mathrm{a}^\dagger \underbrace{\tilde{\sigma}_\mathrm{a}}_{\equiv 0} + \boldsymbol{N}_\mathrm{a}\boldsymbol{J}_\mathrm{b}^\dagger \underbrace{\tilde{\sigma}_\mathrm{b}}_{\equiv 0} + \boldsymbol{N}_\mathrm{ab}\boldsymbol{J}_1^\dagger\boldsymbol{\Lambda}_1\tilde{\boldsymbol{\sigma}}_1 + \boldsymbol{N}_{\mathrm{ab}1}\boldsymbol{J}_2^\dagger\boldsymbol{\Lambda}_2\tilde{\boldsymbol{\sigma}}_2 + \boldsymbol{N}_{\mathrm{ab}12}\boldsymbol{J}_3^\dagger\boldsymbol{\Lambda}_3\tilde{\boldsymbol{\sigma}}_3 \\ &\quad + \ldots + \boldsymbol{N}_{\mathrm{ab}12..(k-1)}\boldsymbol{J}_k^\dagger\boldsymbol{\Lambda}_k\tilde{\boldsymbol{\sigma}}_k \\ &= \boldsymbol{N}_\mathrm{ab}\boldsymbol{J}_1^\dagger\boldsymbol{\Lambda}_1\tilde{\boldsymbol{\sigma}}_1 + \boldsymbol{N}_{\mathrm{ab}1}\boldsymbol{J}_2^\dagger\boldsymbol{\Lambda}_2\tilde{\boldsymbol{\sigma}}_2 + \boldsymbol{N}_{\mathrm{ab}12}\boldsymbol{J}_3^\dagger\boldsymbol{\Lambda}_3\tilde{\boldsymbol{\sigma}}_3 + \ldots + \boldsymbol{N}_{\mathrm{ab}12..(k-1)}\boldsymbol{J}_k^\dagger\boldsymbol{\Lambda}_k\tilde{\boldsymbol{\sigma}}_k.\end{aligned} \tag{32}$$

It can be shown that

$$\dot{\boldsymbol{q}} = \boldsymbol{f}_{11}(\boldsymbol{q},\tilde{\boldsymbol{\sigma}}) \Rightarrow \sigma_\mathrm{a} = \boldsymbol{J}_\mathrm{a}\boldsymbol{f}_{11} = f_{12}(\boldsymbol{q},\tilde{\boldsymbol{\sigma}}) \tag{33}$$
$$\sigma_\mathrm{b} = \boldsymbol{J}_\mathrm{b}\boldsymbol{f}_{11} = f_{13}(\boldsymbol{q},\tilde{\boldsymbol{\sigma}}) \tag{34}$$
$$\dot{\tilde{\boldsymbol{\sigma}}}_{\mathrm{eb}} = -\boldsymbol{M}_1\tilde{\boldsymbol{\sigma}}_{\mathrm{eb}} \tag{35}$$
$$\dot{\boldsymbol{q}} = \boldsymbol{f}_{21}(\boldsymbol{q},\tilde{\boldsymbol{\sigma}}) \Rightarrow \sigma_\mathrm{a} = \boldsymbol{J}_\mathrm{a}\boldsymbol{f}_{21} = f_{22}(\boldsymbol{q},\tilde{\boldsymbol{\sigma}}) \tag{36}$$
$$\sigma_\mathrm{b} = \boldsymbol{J}_\mathrm{b}\boldsymbol{f}_{21} = 0 \tag{37}$$
$$\dot{\tilde{\boldsymbol{\sigma}}}_{\mathrm{eb}} = -\boldsymbol{M}_2\tilde{\boldsymbol{\sigma}}_{\mathrm{eb}} \tag{38}$$
$$\dot{\boldsymbol{q}} = \boldsymbol{f}_{31}(\boldsymbol{q},\tilde{\boldsymbol{\sigma}}) \Rightarrow \sigma_\mathrm{a} = \boldsymbol{J}_\mathrm{a}\boldsymbol{f}_{31} = 0 \tag{39}$$
$$\sigma_\mathrm{b} = \boldsymbol{J}_\mathrm{b}\boldsymbol{f}_{31} = f_{33} \tag{40}$$
$$\dot{\tilde{\boldsymbol{\sigma}}}_{\mathrm{eb}} = -\boldsymbol{M}_3\tilde{\boldsymbol{\sigma}}_{\mathrm{eb}} \tag{41}$$
$$\dot{\boldsymbol{q}} = \boldsymbol{f}_{41}(\boldsymbol{q},\tilde{\boldsymbol{\sigma}}) \Rightarrow \sigma_\mathrm{a} = \boldsymbol{J}_\mathrm{a}\boldsymbol{f}_{41} = 0 \tag{42}$$
$$\sigma_\mathrm{b} = \boldsymbol{J}_\mathrm{b}\boldsymbol{f}_{41} = 0 \tag{43}$$
$$\dot{\tilde{\boldsymbol{\sigma}}}_{\mathrm{eb}} = -\boldsymbol{M}_4\tilde{\boldsymbol{\sigma}}_{\mathrm{eb}}, \tag{44}$$

where $\boldsymbol{M}_i$ for $i \in \{1,\ldots,4\}$ is either equal to or a principal submatrix of the general matrix $\boldsymbol{M}$ in Equation (60) in [26], which is shown to be positive definite given that the task Jacobians are linearly independent and the gain matrices $\boldsymbol{\Lambda}_j$ for $j \in \{1,\ldots,k\}$ are positive definite. Therefore, the matrices $\boldsymbol{M}_i$ are positive definite.

Consider the sets

$$C_1 := [q_{1,\min} + \varepsilon, q_{1,\max} - \varepsilon] \tag{45}$$
$$C_2 := [q_{2,\min} + \varepsilon, q_{2,\max} - \varepsilon] \tag{46}$$
$$C := \mathbb{R}^n \times C_1 \times C_2 \times \mathbb{R}^{l-n-2} \tag{47}$$

with the corresponding tangent cones

$$T_{C_1}(\boldsymbol{z}) = \begin{cases} [\,0,\infty\,) & \sigma_\mathrm{a} = q_{1,\min} + \varepsilon \\ \mathbb{R} & \sigma_\mathrm{a} \in (q_{1,\min} + \varepsilon, q_{1,\max} - \varepsilon)\,, \\ (\,-\infty, 0\,] & \sigma_\mathrm{a} = q_{1,\max} - \varepsilon \end{cases} \tag{48}$$

$$T_{C_2}(\boldsymbol{z}) = \begin{cases} [\,0,\infty\,) & \sigma_\mathrm{b} = q_{2,\min} + \varepsilon \\ \mathbb{R} & \sigma_\mathrm{b} \in (q_{2,\min} + \varepsilon, q_{2,\max} - \varepsilon)\,, \\ (\,-\infty, 0\,] & \sigma_\mathrm{b} = q_{2,\max} - \varepsilon \end{cases} \tag{49}$$

$$\boldsymbol{T}_C(\boldsymbol{z}) = \mathbb{R}^n \times T_{C_1}(\boldsymbol{z}) \times T_{C_2}(\boldsymbol{z}) \times \mathbb{R}^{l-n-2} \tag{50}$$

for an $\varepsilon > 0$. Let $\boldsymbol{f}_1, \boldsymbol{f}_2, \boldsymbol{f}_3, \boldsymbol{f}_4 : C \to \mathbb{R}^l$ be the continuous vector functions

$$\boldsymbol{f}_1(\boldsymbol{z}) = \begin{bmatrix} \boldsymbol{f}_{11}(\boldsymbol{z}) \\ f_{12}(\boldsymbol{z}) \\ f_{13}(\boldsymbol{z}) \\ -\boldsymbol{M}_1\tilde{\boldsymbol{\sigma}} \end{bmatrix} \tag{51}$$

$$\boldsymbol{f}_2(\boldsymbol{z}) = \begin{bmatrix} \boldsymbol{f}_{21}(\boldsymbol{z}) \\ f_{22}(\boldsymbol{z}) \\ 0 \\ -\boldsymbol{M}_2\tilde{\boldsymbol{\sigma}} \end{bmatrix} \tag{52}$$

$$\boldsymbol{f}_3(\boldsymbol{z}) = \begin{bmatrix} \boldsymbol{f}_{31}(\boldsymbol{z}) \\ 0 \\ f_{33}(\boldsymbol{z}) \\ -\boldsymbol{M}_3\tilde{\boldsymbol{\sigma}} \end{bmatrix} \tag{53}$$

$$\boldsymbol{f}_4(\boldsymbol{z}) = \begin{bmatrix} \boldsymbol{f}_{41}(\boldsymbol{z}) \\ 0 \\ 0 \\ -\boldsymbol{M}_4\tilde{\boldsymbol{\sigma}} \end{bmatrix} \tag{54}$$

and consider the sets defined below.

$$S_1 := \{\boldsymbol{z} \in C : \exists \text{ a neighborhood U of } \boldsymbol{z} : \boldsymbol{f}_1(\boldsymbol{x}) \in \boldsymbol{T}_C(\boldsymbol{x}) \forall \boldsymbol{x} \in C \cap U\}. \tag{55}$$

$$S_2 := \{\boldsymbol{z} \in C \backslash S_1 : \exists \text{ a neighborhood U of } \boldsymbol{z} : \boldsymbol{f}_2(\boldsymbol{x}) \in \boldsymbol{T}_C(\boldsymbol{x}) \forall \boldsymbol{x} \in C \cap U\}. \tag{56}$$

$$S_3 := \{\boldsymbol{z} \in C \backslash (S_1 \cup S_2) : \exists \text{ a neighborhood U of } \boldsymbol{z} : \boldsymbol{f}_3(\boldsymbol{x}) \in \boldsymbol{T}_C(\boldsymbol{x}) \forall \boldsymbol{x} \in C \cap U\}. \tag{57}$$

$$S_4 := C \backslash (S_1 \cup S_2 \cup S_3). \tag{58}$$

$S_1$ is the set where $\dot{\boldsymbol{z}}$ follows the vector field $\boldsymbol{f}_1$ and $\boldsymbol{z}$ stays in the set $C$. $S_2$, $S_3$ and $S_4$ make up the complement of $S_1$ and represent the sets where $\dot{\boldsymbol{z}}$ can follow $\boldsymbol{f}_2$, $\boldsymbol{f}_3$ and $\boldsymbol{f}_4$ respectively while the solution $\boldsymbol{z}$ stays in $C$. In mode 4, both set-based tasks are frozen, thus making the $(n+1)$th and $(n+2)$th elements in $\boldsymbol{f}_4(\boldsymbol{z}) \equiv 0$. Hence, $\boldsymbol{f}_4(\boldsymbol{z}) \in \boldsymbol{T}_C(\boldsymbol{z}) \forall \boldsymbol{z} \in C$, and $C$ is strongly forward invariant for $\dot{\boldsymbol{z}} = \boldsymbol{f}_4(\boldsymbol{z})$. The discontinuous equation $\dot{\boldsymbol{z}} = \boldsymbol{f}(\boldsymbol{z})$ with $\boldsymbol{f} : C \to \mathbb{R}^l$ defined as

$$\boldsymbol{f}(\boldsymbol{z}) := \begin{cases} \boldsymbol{f}_1(\boldsymbol{z}) & \boldsymbol{z} \in S_1 \\ \boldsymbol{f}_2(\boldsymbol{z}) & \boldsymbol{z} \in S_2 \\ \boldsymbol{f}_3(\boldsymbol{z}) & \boldsymbol{z} \in S_3 \\ \boldsymbol{f}_4(\boldsymbol{z}) & \boldsymbol{z} \in S_4 \end{cases} \tag{59}$$

then defines our system.

### C. j set-based tasks, k equality tasks

Consider a $n$ DOF robotic system tasked with $k$ equality tasks of $m_i$ DOFs for $i \in \{1,\ldots,k\}$ and $j$ set-based tasks $\in \mathbb{R}$. Denote the $j$th set-based task $\sigma_\mathrm{x}$. Consider the state-vector $\boldsymbol{z} \in \mathbb{R}^l$, where

$$l = n + j + \sum_{i=1}^{k} m_i, \tag{60}$$

and

$$\boldsymbol{z} = \begin{bmatrix} \boldsymbol{q} \\ \boldsymbol{\sigma}_\mathrm{sb} \\ \tilde{\boldsymbol{\sigma}}_{\mathrm{eb}} \end{bmatrix} = \begin{bmatrix} q_1 \\ \vdots \\ q_n \\ \sigma_\mathrm{a} \\ \vdots \\ \sigma_\mathrm{x} \\ \tilde{\boldsymbol{\sigma}}_1 \\ \vdots \\ \tilde{\boldsymbol{\sigma}}_k \end{bmatrix}. \tag{61}$$

With $j$ set-based tasks, it is necessary to consider the $2^j$ solutions resulting from all combinations of ignoring and freezing every set-based task. By analysis similar to the above sections:

| Mode | Description | Equations |
|---|---|---|
| 1 | No set-based tasks frozen | $\dot{\boldsymbol{q}} = \boldsymbol{J}_1^\dagger \boldsymbol{\Lambda}_1 \tilde{\boldsymbol{\sigma}}_1 + ... + \boldsymbol{N}_{12..(k-1)} \boldsymbol{J}_k^\dagger \boldsymbol{\Lambda}_k \tilde{\boldsymbol{\sigma}}_k$ <br> $\Rightarrow \dot{\tilde{\boldsymbol{\sigma}}}_{\text{eb}} = -\boldsymbol{M}_1 \tilde{\boldsymbol{\sigma}}_{\text{eb}}$ <br> $\dot{\boldsymbol{z}} = \boldsymbol{f}_1(\boldsymbol{z})$ |
| 2 | $\sigma_{\text{a}}$ frozen | $\dot{\boldsymbol{q}} = \boldsymbol{N}_{\text{a}} \boldsymbol{J}_1^\dagger \boldsymbol{\Lambda}_1 \tilde{\boldsymbol{\sigma}}_1 + ... + \boldsymbol{N}_{\text{a}12..(k-1)} \boldsymbol{J}_k^\dagger \boldsymbol{\Lambda}_k \tilde{\boldsymbol{\sigma}}_k$ <br> $\Rightarrow \dot{\sigma}_{\text{a}} = 0$ <br> $\dot{\tilde{\boldsymbol{\sigma}}}_{\text{eb}} = -\boldsymbol{M}_2 \tilde{\boldsymbol{\sigma}}_{\text{eb}}$ <br> $\dot{\boldsymbol{z}} = \boldsymbol{f}_2(\boldsymbol{z})$ |
| 3 | $\sigma_{\text{b}}$ frozen | $\dot{\boldsymbol{q}} = \boldsymbol{N}_{\text{b}} \boldsymbol{J}_1^\dagger \boldsymbol{\Lambda}_1 \tilde{\boldsymbol{\sigma}}_1 + ... + \boldsymbol{N}_{\text{b}12..(k-1)} \boldsymbol{J}_k^\dagger \boldsymbol{\Lambda}_k \tilde{\boldsymbol{\sigma}}_k$ <br> $\Rightarrow \dot{\sigma}_{\text{b}} = 0$ <br> $\dot{\tilde{\boldsymbol{\sigma}}}_{\text{eb}} = -\boldsymbol{M}_3 \tilde{\boldsymbol{\sigma}}_{\text{eb}}$ <br> $\dot{\boldsymbol{z}} = \boldsymbol{f}_3(\boldsymbol{z})$ |
| $\ddots$ | $\ddots$ | $\ddots$ |
| $2^j$ | All set-based tasks frozen | $\dot{\boldsymbol{q}} = \boldsymbol{N}_{\text{ab,.x}} \boldsymbol{J}_1^\dagger \boldsymbol{\Lambda}_1 \tilde{\boldsymbol{\sigma}}_1 + ... + \boldsymbol{N}_{\text{ab,.x}12..(k-1)} \boldsymbol{J}_k^\dagger \boldsymbol{\Lambda}_k \tilde{\boldsymbol{\sigma}}_k$ <br> $\Rightarrow \dot{\boldsymbol{\sigma}}_{\text{sb}} = \boldsymbol{0}$ <br> $\dot{\tilde{\boldsymbol{\sigma}}}_{\text{eb}} = -\boldsymbol{M}_{2^j} \tilde{\boldsymbol{\sigma}}_{\text{eb}}$ <br> $\dot{\boldsymbol{z}} = \boldsymbol{f}_{2^j}(\boldsymbol{z})$ |

TABLE I: System equations for the resulting $2^j$ modes for $j$ set-based tasks.

All modes have certain commonalities:

1) All active set-based tasks are frozen.
2) Inactive set-based tasks do not affect the behavior of the system.
3) All matrices $\boldsymbol{M}_i$ for $i \in \{1, \ldots, 2^j\}$ are either equal to or a principal submatrix to the general matrix $\boldsymbol{M}$ in Equation (60) in [26]. Consequently, given linearly independent tasks Jacobians and positive definite gain matrices $\boldsymbol{\Lambda}$, the matrices $\boldsymbol{M}_i$ are positive definite.

Consider the sets

$$C_1 := [\sigma_{a,\min} + \varepsilon_{\text{a}}, \sigma_{a,\max} - \varepsilon_{\text{a}}] \tag{62}$$
$$C_2 := [\sigma_{b,\min} + \varepsilon_{\text{b}}, \sigma_{b,\max} - \varepsilon_{\text{b}}] \tag{63}$$
$$\vdots$$
$$C_j := [\sigma_{\text{x},\min} + \varepsilon_{\text{x}}, \sigma_{\text{x},\max} - \varepsilon_{\text{x}}] \tag{64}$$
$$C := \mathbb{R}^n \times C_1 \times C_2 \times ... \times C_j \times \mathbb{R}^{l-n-j} \tag{65}$$

with the corresponding tangent cones

$$T_{C_1}(\boldsymbol{z}) = \begin{cases} [\,0, \infty\,) & \sigma_{\text{a}} = \sigma_{a,\min} + \varepsilon_{\text{a}} \\ \mathbb{R} & \sigma_{\text{a}} \in (\sigma_{a,\min} + \varepsilon_{\text{a}}, \sigma_{a,\max} - \varepsilon_{\text{a}}) \\ (\,-\infty, 0\,] & \sigma_{\text{a}} = \sigma_{a,\max} - \varepsilon_{\text{a}} \end{cases}, \tag{66}$$

$$T_{C_2}(\boldsymbol{z}) = \begin{cases} [\,0, \infty\,) & \sigma_{\text{b}} = \sigma_{b,\min} + \varepsilon_{\text{b}} \\ \mathbb{R} & \sigma_{\text{b}} \in (\sigma_{b,\min} + \varepsilon_{\text{b}}, \sigma_{b,\max} - \varepsilon_{\text{b}}) \\ (\,-\infty, 0\,] & \sigma_{\text{b}} = \sigma_{b,\max} - \varepsilon_{\text{b}} \end{cases}, \tag{67}$$

$$\vdots$$

$$T_{C_j}(\boldsymbol{z}) = \begin{cases} [\,0, \infty\,) & \sigma_{\text{x}} = \sigma_{\text{x},\min} + \varepsilon_{\text{x}} \\ \mathbb{R} & \sigma_{\text{x}} \in (\sigma_{\text{x},\min} + \varepsilon_{\text{x}}, \sigma_{\text{x},\max} - \varepsilon_{\text{x}}) \\ (\,-\infty, 0\,] & \sigma_{\text{x}} = \sigma_{\text{x},\max} - \varepsilon_{\text{x}} \end{cases}, \tag{68}$$

$$\boldsymbol{T}_C(\boldsymbol{z}) = \mathbb{R}^n \times T_{C_1}(\boldsymbol{z}) \times T_{C_2}(\boldsymbol{z}) \times ... \times T_{C_j}(\boldsymbol{z}) \times \mathbb{R}^{l-n-j}. \tag{69}$$

for some $\varepsilon_i > 0$ for $i \in \{1, \ldots, j\}$.

$$S_1 := \{\boldsymbol{z} \in C : \exists \text{ a neighborhood U of } \boldsymbol{z} : \boldsymbol{f}_1(\boldsymbol{x}) \in \boldsymbol{T}_C(\boldsymbol{x}) \forall \boldsymbol{x} \in C \cap U\} \tag{70}$$
$$S_2 := \{\boldsymbol{z} \in C \backslash S_1 : \exists \text{ a nbhd U of } \boldsymbol{z} : \boldsymbol{f}_2(\boldsymbol{x}) \in \boldsymbol{T}_C(\boldsymbol{x}) \forall \boldsymbol{x} \in C \cap U\} \tag{71}$$
$$\vdots$$
$$S_{2^j} := C \backslash \left(S_1 \cup S_2 \cup S_3 \cup ... \cup S_{2^j - 1}\right). \tag{72}$$

The discontinuous equation $\dot{\boldsymbol{z}} = \boldsymbol{f}(\boldsymbol{z})$ with $\boldsymbol{f} : C \to \mathbb{R}^l$ defined as

$$\boldsymbol{f}(\boldsymbol{z}) := \begin{cases} \boldsymbol{f}_1(\boldsymbol{z}) & \boldsymbol{z} \in S_1 \\ \boldsymbol{f}_2(\boldsymbol{z}) & \boldsymbol{z} \in S_2 \\ \boldsymbol{f}_3(\boldsymbol{z}) & \boldsymbol{z} \in S_3 \\ \vdots & \vdots \\ \boldsymbol{f}_{2^j}(\boldsymbol{z}) & \boldsymbol{z} \in S_{2^j} \end{cases} \tag{73}$$

then defines our system.

## IV. STABILITY ANALYSIS

To study the behavior of the discontinuous differential equation (73), we look at the corresponding constrained differential inclusion:

$$\boldsymbol{z} \in C \quad \dot{\boldsymbol{z}} \in \boldsymbol{F}(\boldsymbol{z}) \tag{74}$$

where

$$\boldsymbol{F}(\boldsymbol{z}) := \bigcap_{\delta > 0} \text{co}\overline{\boldsymbol{f}((\boldsymbol{z} + \delta \mathbb{B}) \cap C)} \quad \forall \boldsymbol{z} \in C. \tag{75}$$

$\mathbb{B}$ is a unit ball in $\mathbb{R}^{\dim(\boldsymbol{z})}$ centered at the origin. co$\overline{P}$ denotes the closed convex hull of the set $P$, in other words, the smallest closed convex set containing $P$. $\boldsymbol{z}$ is the Krasovskii solution of the discontinuous differential equation (73) (Definition 4.2 [24]).

### A. Convergence of equality tasks

The first part of the stability proof considers the convergence of the system equality tasks when switching between $2^j$ possible modes of the system.

If $V$ is a continuously differentiable Lyapunov function for $\dot{\boldsymbol{z}} = \boldsymbol{f}_i(\boldsymbol{z})$ for all $i \in \{1, \ldots, 2^j\}$, then $V$ is a Lyapunov function for $\dot{\boldsymbol{z}} \in \boldsymbol{F}(\boldsymbol{z})$. The following equation holds as all $\boldsymbol{f}_i(\boldsymbol{z})$ are continuous functions:

$$\boldsymbol{f} \in \boldsymbol{F}(\boldsymbol{z}) \Rightarrow \boldsymbol{f} = \lambda_1 \boldsymbol{f}_1(\boldsymbol{z}) + \lambda_2 \boldsymbol{f}_2(\boldsymbol{z}) + ... + \lambda_{2^j} \boldsymbol{f}_{2^j}(\boldsymbol{z}) \tag{76}$$

for some $\sum_{i=1}^{2^j} \lambda_i = 1, \lambda_i \geq 0$. Consider the Lyapunov function candidate for the equality task errors:

$$V(\tilde{\boldsymbol{\sigma}}_{\text{eb}}) = \frac{1}{2} \tilde{\boldsymbol{\sigma}}_{\text{eb}}^T \tilde{\boldsymbol{\sigma}}_{\text{eb}}. \tag{77}$$

Using (76) and the system equations given in Table I, we find that the time derivative of $V$ is given by

$$\begin{aligned} \dot{V} &= \tilde{\boldsymbol{\sigma}}_{\text{eb}}^T (\lambda_1 (-\boldsymbol{M}_1 \tilde{\boldsymbol{\sigma}}_{\text{eb}}) + \lambda_2 (-\boldsymbol{M}_2 \tilde{\boldsymbol{\sigma}}_{\text{eb}}) + ... + \lambda_{2^j} (-\boldsymbol{M}_{2^j} \tilde{\boldsymbol{\sigma}}_{\text{eb}})) \\ &= -\tilde{\boldsymbol{\sigma}}_{\text{eb}}^T (\lambda_1 \boldsymbol{M}_1 + \lambda_2 \boldsymbol{M}_2 + ... + \lambda_{2^j} \boldsymbol{M}_{2^j}) \tilde{\boldsymbol{\sigma}}_{\text{eb}} \\ &= -\tilde{\boldsymbol{\sigma}}_{\text{eb}}^T \boldsymbol{Q} \tilde{\boldsymbol{\sigma}}_{\text{eb}}. \end{aligned} \tag{78}$$

The convex combination $\boldsymbol{Q}$ of positive definite matrices is also positive definite. Therefore, $\dot{V}$ is negative definite and $\tilde{\boldsymbol{\sigma}}_{\text{eb}} = \boldsymbol{0}$ is a globally asymptotically stable equilibrium point in all modes. Thus, the equality task errors $\tilde{\boldsymbol{\sigma}}_{\text{eb}}$ asymptotically converge to zero when switching between modes. Furthermore, if $\boldsymbol{q}$ belongs to a compact set, the equilibrium point $\tilde{\boldsymbol{\sigma}}_{\text{eb}} = \boldsymbol{0}$ is exponentially stable.

### B. Satisfaction of set-based tasks and existence of solution

The second part of the stability proof considers the satisfaction of the set-based tasks and the existence of a valid solution.

We have defined a closed set $C$ in (65) within which all set-based tasks are satisfied at all times. As long as the system solution $\boldsymbol{z} \in C$, the set-based tasks are not violated.

For a system with only high-priority set-based tasks, (73) defines the system: $\dot{\boldsymbol{z}} = \boldsymbol{f}_1(\boldsymbol{z})$ as long as the solution $\boldsymbol{z}$ stays in $C$. If the system reaches the boundary of $C$ and remaining in mode 1 would cause $\boldsymbol{z}$ to leave $C$, another mode is activated. If neither of the vector fields $\boldsymbol{f}_1$-$\boldsymbol{f}_{2^j-1}$ will result in $\boldsymbol{z}$ staying in $C$, the chosen solution is $\dot{\boldsymbol{z}} = \boldsymbol{f}_{2^j}(\boldsymbol{z})$, for which it has been shown that $\dot{\boldsymbol{\sigma}}_{\text{sb}} \equiv \boldsymbol{0}$. Therefore, $\boldsymbol{f}_{2^j}(\boldsymbol{z}) \in \boldsymbol{T}_C(\boldsymbol{z}) \forall \boldsymbol{z} \in C$, and $C$ is strongly forward invariant for $\dot{\boldsymbol{z}} = \boldsymbol{f}_{2^j}(\boldsymbol{z})$. Thus, there will always exist a solution $\boldsymbol{z} \in C$ and the set-based tasks are consequently always satisfied.

## V. SIMULATION RESULTS

To illustrate the effectiveness of the proposed method, a simple example has been implemented in Matlab for a planar three-link manipulator. This section presents the simulations results.

The simulated example is the case presented in Section III-A: The manipulator has one set-based task $\sigma_{\text{a}}$, which is to avoid a circular obstacle with center in $(0, 2.6)$ m and radius $r = 0.65$ m. As described above, an $\varepsilon = 0.75$ m is chosen as a lower limit for the distance between the end effector and the obstacle center. Furthermore, the system is given one equality task $\boldsymbol{\sigma}_1 \in \mathbb{R}^3 = [x_{ee}, y_{ee}, \psi_{ee}]^T$ to guide the end effector to a certain position and orientation. In this example, this position is $(-2, 3)$ m and the desired orientation is $\frac{\pi}{2}$ radians. The equality task gain matrix $\boldsymbol{\Lambda}_1 = \boldsymbol{I}$. The manipulator links have length 1.75 m, 1.25 m and 1.0 m, respectively. Furthermore, the manipulator has been implemented with a saturation on the joint velocities of $10 \frac{\text{deg}}{\text{s}}$. The simulation results are shown in Figure 2. The system starts in mode 1, ignoring the obstacle and moving in a straight line towards the target (Figure 2a). However, by doing so the end effector approaches the obstacle and soon reaches the minimum allowed distance $\varepsilon$ to the obstacle center (Figure 2b). Obviously, by continuing to follow the straight line in mode 1, this distance would decrease further. Thus, the system switches to mode 2 and freezes the distance at $\sigma_{\text{a}} = \varepsilon$. The end effector then moves towards the desired end effector position along the circle with center in the obstacle and radius $\varepsilon$. Eventually it reaches a point where the shortest path between the end effector and desired position does not bring the manipulator closer to the obstacle (Figure 2c). The system then changes back to mode 1 and converges to the desired position and orientation (Figure 2d and 3). Figure 4 confirms that the set-based task is satisfied at all times.

The theoretical results in this paper are further confirmed by simulations and experiments in [27].

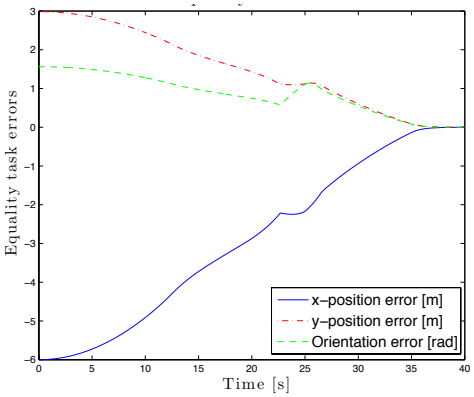



Fig. 3: Equality task errors for Example 1. Both the position and orientation errors converge to 0.

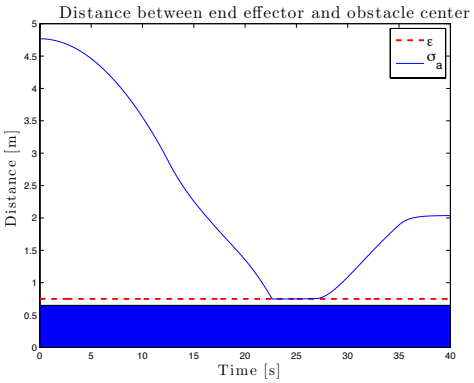
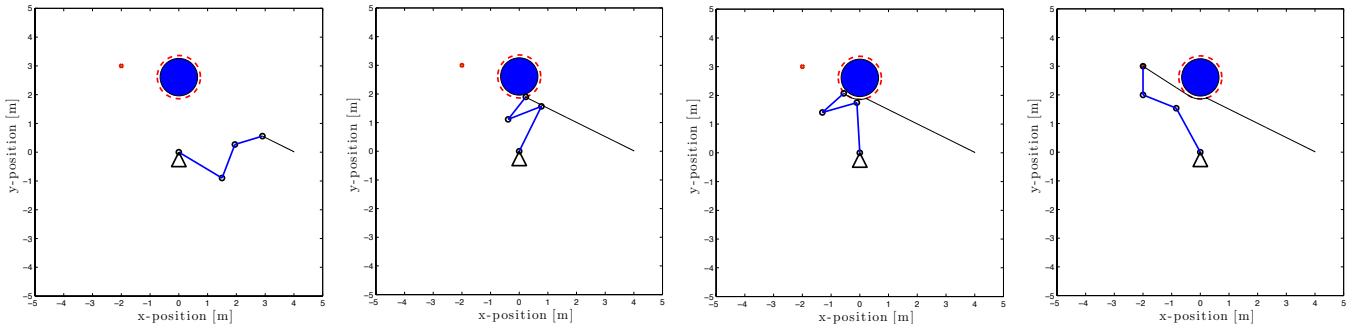



Fig. 4: Evolution of distance between end effector and obstacle center. The simulations confirm that the chosen task limit of $\varepsilon$ is not violated at any time.

## VI. CONCLUSIONS

This paper has presented an approach to include set-based tasks in the multiple task-priority inverse kinematics framework. As a default solution, only the system equality tasks are considered and implemented with the desired priority by projecting the low-priority task velocities through the nullspaces of the high-priority tasks. However, if this solution would result in a set-based task (e.g. a joint limit) being violated, this task is included in the task hierarchy with a certain priority with the goal of freezing the task at its current value. It is shown that set-based tasks given high priority, i.e. above the highest priority equality task, are fulfilled at all times. For lower-priority set-based tasks, this can not be guaranteed due to the influence of the higher-priority equality-based tasks. Due to the switching between set-based tasks being active/inactive, the resulting closed-loop dynamic system can be described as a discontinuous differential equation. Using switched control systems theory it has been proven that the equality task errors converge asymptotically to zero when including set-based tasks into the framework given that certain, specified conditions are fulfilled. Furthermore, presented simulation results illustrate the effectiveness of the proposed method.

Future work includes analyzing computational time and




(a) Time = 10 s (b) Time = 23 s (c) Time = 28 s (d) Time = 40 s

Fig. 2: Simulation results Example 1: The desired position is marked as a red cross at $(-2, 3)$ m. The obstacle is the blue circle, and the red dashed line around it marks the area within which the end effector is not allowed to ensure avoidance of the obstacle.

finding an optimal implementation, as well as considering some smoothing function to ensure continuous acceleration profiles for switches between modes.

## ACKNOWLEDGMENTS

This work was supported by the Research Council of Norway through the Center of Excellence funding scheme, project number 223254, the European Community through the projects ARCAS (FP7-287617), EuRoC (FP7-608849), DexROV (H2020-635491) and AEROARMS (H2020-644271), NSF grant ECCS-1232035 and AFOSR grant FA9550-15-1-0155.

## REFERENCES

[1] B. Siciliano and O. Khatib, Eds., *Springer Handbook of Robotics*. Springer Berlin Heidelberg, 2008.

[2] G. Antonelli, *Underwater Robots, Third Edition*. Springer International Publishing, 2014.

[3] R. Christ and R. Wernli, *The ROV Manual: A User Guide for Remotely Operated Vehicles*. Elsevier Science, 2013.

[4] K. Valavanis, P. Oh, and L. A. Piegla, *Unmanned Aircraft Systems*. Springer Netherlands, 2009.

[5] D. Nenchev, Y. Umetani, and K. Yoshida, "Analysis of a redundant free-flying spacecraft/manipulator system," *IEEE Transactions on Robotics and Automation*, vol. 8, no. 1, pp. 1–6, 1992.

[6] F. Caccavale and B. Siciliano, "Kinematic control of redundant free-floating robotic systems," *Advanced Robotics*, vol. 15, no. 4, pp. 429–448, 2001.

[7] O. Egeland and K. Pettersen, "Free-floating robotic systems," in *Control Problems in Robotics and Automation*. Springer Berlin Heidelberg, 1998, vol. 230, pp. 119–134.

[8] C. Klein and C.-H. Huang, "Review of pseudoinverse control for use with kinematically redundant manipulators," *IEEE Transactions on Systems, Man and Cybernetics*, vol. 13, no. 2, pp. 245–250, 1983.

[9] S. R. Buss, "Introduction to inverse kinematics with jacobian transpose, pseudoinverse and damped least squares methods," *IEEE Journal of Robotics and Automation*, vol. 17, pp. 1–19, 2004.

[10] S. Chiaverini, G. Oriolo, and I. D. Walker, *Springer Handbook of Robotics*. Heidelberg, D: B. Siciliano, O. Khatib, (Eds.), Springer-Verlag, 2008, ch. Kinematically Redundant Manipulators, pp. 245–268.

[11] B. Siciliano, L. Sciavicco, L. Villani, and G. Oriolo, *Robotics: modelling, planning and control*. Springer Verlag, 2009.

[12] G. Antonelli, F. Arrichiello, and S. Chiaverini, "The Null-Space-Based behavioral control for autonomous robotic systems," *Intelligent Service Robotics*, vol. 1, no. 1, pp. 27–39, 2008.

[13] E. Marchand, F. Chaumette, and A. Rizzo, "Using the task function approach to avoid robot joint limits and kinematic singularities in visual servoing," in *Proc. 1996 International Conference on Intelligent Robots and Systems*, Osaka, Japan, 1996.

[14] H. Hanafusa, T. Yoshikawa, and Y. Nakamura, "Analysis and control of articulated robot arms with redundancy," in *Proc. 8th ZFAC World Congress*, 1981.

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

[16] O. Khatib, "The potential field approach and operational space formulation in robot control," in *Adaptive and Learning Systems: Theory and Applications*, K. S. Narendra, Ed. Springer US, 1986.

[17] B. Faverjon and P. Tournassoud, "A local based approach for path planning of manipulators with a high number of degrees of freedom," in *IEEE International Conference on Robotics and Automation. Proceedings*, vol. 4, 1987, pp. 1152–1159.

[18] A. Escande, N. Mansard, and P.-B. Wieber, "Hierarchical quadratic programming: Fast online humanoid-robot motion generation," *International Journal of Robotics Research*, 2013.

[19] E. Simetti, G. Casalino, S. Torelli, A. Sperindé, and A. Turetta, "Floating underwater manipulation: Developed control methodology and experimental validation within the TRIDENT project," *Journal of Field Robotics*, vol. 31(3), pp. 364–385, 2013.

[20] N. Mansard and O. Khatib, "Continuous control law from unilateral constraints," in *IEEE International Conference on Robotics and Automation (ICRA 2008)*, 2008.

[21] N. Mansard, O. Khatib, and A. Kheddar, "A unified approach to integrate unilateral constraints in the stack of tasks," *IEEE Transaction on Robotics*, pp. 670–685, 2009.

[22] G. Antonelli, S. Moe, and K. Pettersen, "Incorporating set-based control within the singularity-robust multiple task-priority inverse kinematics," in *Proc. 2014 IROS Workshop on Real-time Motion Generation and Control*, Chicago, USA, 2014.

[23] ——, *Incorporating Set-based Control within the Singularity-robust Multiple Task-priority Inverse Kinematics*. Submitted to 23rd Mediterranean Conference on Control and Automation, 2015.

[24] R. Goebel, R. Sanfelice, and A. Teel, *Hybrid Dynamical Systems: Modeling, Stability, and Robustness*. Princeton University Press, 2012.

[25] 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.

[26] 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, October, 2009.

[27] S. Moe, G. Antonelli, K. Pettersen, and J. Schrimpf, *Experimental Results for Set-based Control within the Singularity-robust Multiple Task-priority Inverse Kinematics Framework*. Submitted to IEEE International Conference on Robotics and Biomimetics (ROBIO 2015), 2015.