Set-Based Task Priority Control for Articulated Intervention AUVs

Erlend Andreas Basso

Supervisor: Professor Kristin Ytterstad Pettersen
Co-supervisor: Dr. Anna Kohl

February 1, 2019

NTNU
Norwegian University of
Science and Technology

Abstract

The Articulated Intervention Autonomous Underwater Vehicle (AIAUV) has emerged from the Underwater Snake Robot (USR) by equipping it with longitudinal and tunnel thrusters. The AIAUV is an overactuated and highly redundant underwater floating base manipulator, where its entire articulated body serves as a floating manipulator arm, while its slender shape allows it to access narrow and confined spaces.

This thesis examines different schemes for redundancy resolution, which allow a robotic system to perform multiple task arranged in priority simultaneously. Redundancy resolution is investigated at the velocity, acceleration and torque level by resorting to kinematic and operational space control. Most existing frameworks can only handle equality tasks, where the goal is to bring a control objective towards a desired value. However, there has been an extensive research effort in recent years in order to extend these frameworks to handle set-based tasks, where the control objective should be kept within some interval.

In this thesis the set-based singularity-robust multiple task priority (SRMTP) framework is extended to support multidimensional set-based tasks and formalized for acceleration level redundancy resolution. Moreover, the same technique for activation and deactivation of set-based tasks is used in connection with a operational space task priority control frameworks, enabling set-based operational space control.

Contents

List of Tables

List of Figures

Chapter 1

Introduction

1.1 Motivation

As the number of subsea oil and gas installations continue to grow while ageing subsea infrastructure requires more preventive maintenance, the need for subsea inspection, maintenance and repair (IMR) solutions is increasing [1]. Historically, the remotely operated vehicle (ROV) has been the go-to solution for all subsea IMR operations. ROVs are operated by a human operator via a tethered telecommunications link from a submarine or surface ship, ROV operations are therefore both costly and time consuming. Increasing the autonomy of subsea IMR operations has the potential to significantly improve the safety and cost-effectiveness of operations [2]. While AUVs and smaller inspection class ROVs have gradually taken over subsea inspection operations, manipulation tasks and operations within narrow and confined areas still requires the flexibility of a robotic arm.

Articulated intervention autonomous underwater vehicles (AIAUVs) have emerged from underwater snake robots (USRs) by adding longitudinal and tunnel thrusters along the body. AIAUVs are a special class of underwater vehicle manipulator systems (UVMSs), where the system is both vehicle and manipulator

at the same time [3]. The small size of the AIAUV and its articulated body enable it to better access narrow and confined spaces compared to UVMSs with a large floating base. Moreover, because of its flexible shape, the AIAUV can assume a torpedo shape similar to a conventional AUV for transportation, while serving as a floating base manipulator for intervention tasks. These modes of operation are referred to as transport and work mode, respectively.

AIAUVs are highly redundant and overactuated robotic systems. Both of these properties lead to some kind of optimization problem to be solved by the control system. This thesis investigates task priority frameworks for redundancy resolution, wherein control objectives are prioritized according to their respective importance. Safety related tasks such as collision avoidance and joint limit avoidance are inherently set-based tasks, which do not fit naturally into the original formulations of most of these frameworks. Therefore, methods for inclusion of set-based tasks within these frameworks are also investigated. To this end, different control frameworks for set-based task-priority control of AIAUVs are developed, these frameworks exploit the redundancy of the AIAUV in order to satisfy several equality control tasks arranged in priority, while satisfying high priority safety related set-based objectives at all times.

1.2 Problem Description

AIAUVs are redundant with respect to standard tasks like end-effector configuration control since they possess more DOFs than those strictly required to execute the task. This presents the possibility of achieving additional tasks simultaneously by utilizing the redundant DOFs of the system. To this end, task priority frameworks enable multiple tasks to be defined and prioritized with respect to their relative importance. Safety related tasks such as collision avoidance and joint limit avoidance are inherently described by inequalities representing the sets in which they are satisfied. These types of tasks are referred to as set-based tasks, and a large research effort has been put into extending task priority frameworks to handle these tasks in recent years.

The motivation for this thesis has been to investigate redundancy resolution methods for AIAUVs, with a focus on task priority frameworks and the inclusion of set-based tasks within these frameworks.

- Literature review: Compare the methods proposed in [4] and [5] for extension of the task priority inverse kinematic control approach to set-based tasks.
- Extend the set-based SRMTP framework from [4] to handle multidimensional set-based tasks.
- Investigate set-based task priority control at the acceleration and force/torque level.
- Implement and validate the proposed set-based task priority schemes for AIAUV control in simulations.

1.3 Literature review

1.3.1 Task priority inverse kinematic control of redundant robotic systems

The task priority strategy for kinematic control was first introduced in [6], and later developed in [7]. Within this framework, tasks are described in operational space and ordered according to their priority. The relationship between the task variables in operational space and those describing the robotic system in joint space are resolved at the velocity level, generating desired velocity references for some dynamic controller to follow. The idea is that lower priority tasks are realized by utilizing the redundancy not employed through satisfying the higher priority tasks, which is to say that the lower priority tasks are only satisfied in the null space of the higher priority tasks. This ensures that tasks of lower priority have no effect on the satisfaction of higher priority tasks.

This approach was further generalized to an arbitrary number of tasks through a recursive implementation in [8], which also simplifies the problem from a high-level programming point of view. This algorithm is known as the multiple-task priority (MTP) inverse kinematics framework. This method ensures the optimal execution of lower priority tasks as long as they are compatible with the higher priority tasks. However, whenever tasks are incompatible, algorithmic singularities arise.

In [9] a new task-priority redundancy resolution technique for two tasks is introduced which overcomes the effects of algorithmic singularities. This work was extended to an arbitrary number of tasks in [10], [11] and [12], which employ closed-loop inverse kinematics (CLIK) [13] versions of the algorithm to prevent drift of the desired joint angles. The framework presented in [12] is known as the singularity-robust multiple task-priority inverse kinematics framework.

1.3.2 Inclusion of set-based tasks in a task priority inverse kinematic framework

Tasks such as joint limit avoidance, collision avoidance or avoidance of kinematic singularities are inherently set-based tasks, also known as unilateral or inequality tasks in the literature. These types of tasks do not fit within the original task priority frameworks since they only account for equality tasks, i.e. tasks assigning an exact value to a controlled task. The collision avoidance problem for mobile robots was addressed in [14], by defining the concept of artificial potential fields which attempt to push the robot away from an obstacle. However, four distinct drawbacks are identified in [15], such as the possibility of undesirable oscillating system behavior in the presence of certain obstacles, and the fact that it is not possible to set a minimum distance between the obstacle and robot.

Within task priority frameworks, set-based tasks have often been included by transforming them into equality tasks, by assigning an arbitrary target value for the controlled variable within the valid set of the task [16]. This presents significant challenges since safety-related set-based tasks should be

classified as high-priority tasks, but doing so is infeasible in classical task-priority frameworks such as SRMTP, since it consumes too many degrees of freedom (DOF). Consequently, set-based tasks have historically ended up with a lower priority than mission related tasks such as end-effector position control, which meant that the satisfaction of safety-related set-based tasks could not be guaranteed, which is highly undesirable from a safety point of view.

In [4] the SRMTP framework was extended to scalar set-based tasks by introducing the concept of tangent cones. In this scheme set-based tasks are ignored and considered inactive when they are inside their valid set, and accounted for in the inverse kinematics whenever they reach the boundary of their valid set and the task derivative simultaneously points out of the valid set. This is accomplished by switching between different joint velocity reference solutions corresponding to which set-based tasks are active or inactive. This method respects the strict priority of all tasks and ensures that high-priority set-based tasks remain in their valid set at all times. However, a disadvantage of this method is that the system velocity references are inherently discontinuous due to the switching of modes without any kind of smoothing. The discontinuities arising from mode switching is discussed in [17], where a solution for smoothing the system reference velocities while switching between modes is proposed. An immediate drawback of the smoothing approach is that the strict priority of tasks is lost while switching between modes, which suggests that there is an inherent trade-off between strict priority and smoothness of system velocity references when employing this set-based inverse kinematics scheme.

A scheme which integrates activation and deactivation of tasks as well as set-based tasks within task-priority inverse kinematic control is presented in [5]. This method has the benefit of activating and deactivating set-based tasks without incurring discontinuities in the joint velocity references. As opposed to the set-based SRMTP framework, this method relies on defining activation thresholds around the boundaries of the valid set of set-based tasks. Another difference is that set-based tasks are controlled toward some arbitrary value within the valid set instead of freezing them at the boundary. Furthermore,

strict priority between tasks is lost whenever tasks are in transition, which occurs whenever tasks reach the activation threshold and ends when the task either reaches the boundary and is fully active, or is pushed inside the subset of the valid set defined by the activation threshold. The authors argue that the loss of strict priority is in fact a positive characteristic since DOFs are not fully consumed by the active set-based task while it is still contained inside its valid set and not on the boundary and fully active. However, in [18] the author argues that the loss of strict priority may lead to undesirable behaviors. Another approach to handle set-based and equality based tasks while respecting their priorities consists of transforming the inverse kinematics problem into a quadratic programming (QP) problem [19], [20]. This is an iterative procedure which is often computationally expensive.

1.3.3 Operational space task priority control

Prior to the operational space framework, a lot of research was based on joint space dynamic models for dynamic control of robotic manipulators. The operational space formulation was introduced in the seminal work of Khatib [21] as a unified approach to motion and force control. It was intended as a tool for analysis and control of robotic manipulators with respect to the dynamic behavior of their end-effectors. For redundant robotic systems, an operational space task can be defined and controlled, while a dynamically consistent null-space describes the additional motion of the redundant DOFs. Several tasks can be stacked into a single task vector and simultaneously controlled, however, there is no prioritization among tasks and conflicts between tasks lead to a tracking error affecting all of the conflicting tasks.

In [22], [23] and [24] the operational space formulation was extended to an arbitrary number of tasks by generalizing the dynamically consistent null-space operator defined in [21] to any number of priority levels, ensuring a prioritized hierarchy among tasks. In this work, redundancy resolution is performed in a way that dynamically decouples all tasks whenever they are compatible. In

[25], static and dynamic consistency of null-space operators is discussed, and a selection of null-space operators are compared. Since dynamic model parameters are involved in the computation of the null-space operators in the operational space framework, null-space operators may lose consistency properties whenever the model parameters are uncertain. This is investigated in [26], where the conclusion is that the dynamically consistent null-space operator from [21] is statically consistent even if the inertia matrix contains modeling errors. Finally, the operational space framework was also extended to include set-based tasks in [27], however, this approach does not scale well for systems with a high amount of DOFs.

1.4 Background and Contributions

The Matlab simulator used in the simulations was provided by PhD candidate Henrik Schmidt-Didlaukies, the simulator has been ported to Simulink in order to support fixed-step ODE solvers. Matlab functions for computation of a symbolic actuator configuration matrix as well as the partial and cross partial derivatives of the actuator configuration matrix has been provided by MSc student Arnt-Erik Stene.

The main contributions of this thesis are

- Extending the set-based SRMTP framework to support multidimensional set-based tasks.
- Formalizing the set-based SRMTP framework for acceleration level redundancy resolution.
- Introducing a scheme for inclusion of set-based tasks in an operational space framework.
- Evaluating different set-based task priority control schemes for an AIAUV application.

1.5 Outline

This thesis can be divided into two parts, Chapter 2-4 is concerned with task priority control frameworks for general robotic systems, while Chapter 5-9 considers these task priority control frameworks for an AIAUV application.

The rest of thesis is organized as follows. Chapter 2 presents background theory on kinematic task priority control of robotic systems. In Chapter 3, kinematic control frameworks for set-based kinematic control are presented. Moreover, a method for extending the set-based SRMTP framework to handle multidimensional tasks is formalized. Chapter 4 discusses operational space control in a task priority control setting, where a method for set-based operational space control is formalized.

Chapter 5 presents dynamic and kinematic models of an AIAUV for control and simulation purposes. In Chapter 6 the general control structure for AIAUV control and the set-based and equality tasks to be controlled in simulations are also presented. Moreover, the proposed task priority frameworks from Chapter 2-4 are described for AIAUV control. Simulation results of all schemes can be found in Chapter 7, while the results are discussed in Chapter 8. Finally, the conclusion and suggestions for future work can be found in Chapter 9.

Appendix A presents a few novel proofs of the generalized null space operator employed in the operational space controllers.

Chapter 2

Kinematic Control

2.1 Introduction

Robotic systems are often required to perform tasks specified in a suitably defined operational space, also known as task space, while the robotic system is naturally described and actuated in the joint space. Hence, in order to accomplish control tasks, it is necessary to exploit mathematical relations enabling the computation of reference joint-space variables corresponding to the assigned task variables. This is the objective of the inverse kinematics problem, which can be solved at the position, velocity or acceleration level. The reference joint-space variables are the input to a dynamic controller, which computes generalized forces and torques to be applied to the robotic system.

2.2 The Inverse Kinematics Problem

The system configuration of an n degree of freedom robotic system can be expressed by the joint variables $\boldsymbol{q} = [q_1, q_2, \ldots, q_n]^{\mathrm{T}}$. A task is defined as a generic m-dimensional control objective, specified as a function of the system configuration, $\boldsymbol{\sigma}(\boldsymbol{q}) \in \mathbb{R}^m$. The classical definition of the inverse kinematics

problem consists of finding the joint variables $\boldsymbol{q}_d$ corresponding to a given end-effector position and orientation. However, since this thesis focuses on redundant manipulators, tasks are not restricted to the end-effector configuration task. Hence, the inverse kinematics problem will herein consist of finding the system configuration vector $\boldsymbol{q}_d$ that brings a generic task vector $\boldsymbol{\sigma}(\boldsymbol{q})$ to its desired trajectory $\boldsymbol{\sigma}_d$. The relation between the task variable and the joint variables are generally given by

$$\boldsymbol{\sigma}(t) = \boldsymbol{f}(\boldsymbol{q}(t)), \tag{2.1}$$

which must be inverted and solved for $\boldsymbol{q}$. However, since (2.1) is nonlinear in general, it is difficult to obtain a trajectory $\boldsymbol{q}_d$, resulting in the desired task trajectory $\boldsymbol{\sigma}_d$. By differentiating (2.1) with respect to time, the first-order differential kinematics are obtained, viz.

$$\dot{\boldsymbol{\sigma}}(t) = \frac{\partial \boldsymbol{f}(\boldsymbol{q}(t))}{\partial \boldsymbol{q}} \dot{\boldsymbol{q}}(t) = \boldsymbol{J}(\boldsymbol{q}(t))\dot{\boldsymbol{q}}(t), \tag{2.2}$$

where $\boldsymbol{J}(\boldsymbol{q}(t)) \in \mathbb{R}^{m \times n}$ is the configuration-dependent task Jacobian matrix and $\dot{\boldsymbol{q}}(t) \in \mathbb{R}^n$ is the system velocity. Differentiation with respect to time yields the second-order differential kinematics

$$\ddot{\boldsymbol{\sigma}}(t) = \boldsymbol{J}(\boldsymbol{q}(t))\ddot{\boldsymbol{q}}(t) + \dot{\boldsymbol{J}}(\boldsymbol{q}(t))\dot{\boldsymbol{q}}(t). \tag{2.3}$$

Note that the relation between the joint velocity $\dot{\boldsymbol{q}}$ and task velocity $\dot{\boldsymbol{\sigma}}$ is linear, which is also true for the relation between the joint acceleration $\ddot{\boldsymbol{q}}$ and task acceleration $\ddot{\boldsymbol{\sigma}}$. This is why most inverse kinematics algorithms operate on the velocity or acceleration level.

For instance, an end-effector task for a 6-DOF robotic system results in a square task Jacobian. Omitting dependencies for readability, the solution to the first- and second-order differential kinematics equations can therefore be written

as

$$\dot{\boldsymbol{q}} = \boldsymbol{J}^{-1}\dot{\boldsymbol{\sigma}}, \tag{2.4}$$

and

$$\ddot{\boldsymbol{q}} = \boldsymbol{J}^{-1}\left(\ddot{\boldsymbol{\sigma}} - \dot{\boldsymbol{J}}\dot{\boldsymbol{q}}\right). \tag{2.5}$$

2.3 Redundant Robotic Systems

A robotic system is kinematically redundant when it has more DOFs than those strictly required to execute a given task [28]. A direct implication is that no robotic system is inherently redundant, but that there exists task which the system is redundant with respect to. Since a general end-effector configuration task requires six degrees of freedom, robotic systems with seven or more degrees of freedom are usually considered redundant systems.

Consider the case where the number of DOFs are greater than the task dimension. The task Jacobian matrix is no longer square, and the relations in (2.4) and (2.5) cannot be directly inverted to obtain $\dot{\boldsymbol{q}}(t)$. However, the right Moore-Penrose pseudoinverse can be employed to find a solution minimizing the norm of the joint velocities whenever the task Jacobian has full rank, viz.

$$\dot{\boldsymbol{q}} = \boldsymbol{J}^{\dagger}\dot{\boldsymbol{\sigma}}, \tag{2.6}$$

$$\ddot{\boldsymbol{q}} = \boldsymbol{J}^{\dagger}\left(\ddot{\boldsymbol{\sigma}} - \dot{\boldsymbol{J}}\dot{\boldsymbol{q}}\right), \tag{2.7}$$

where the matrix $\boldsymbol{J}^{\dagger} = \boldsymbol{J}^{\mathrm{T}}\left(\boldsymbol{J}\boldsymbol{J}^{\mathrm{T}}\right)^{-1}$ is the right pseudoinverse of $\boldsymbol{J}$. The common denominator of all task-priority schemes relies on the observation that the solution $\dot{\boldsymbol{q}}(t)$ of (2.6) lies in the row space of the task Jacobian. Hence, it is possible to exploit the orthogonal complement of the row space, i.e. the null space of the task Jacobians to potentially satisfy lower priority tasks if the system exhibits

enough redundancy. A more general solution to (2.6) is then given by [29]

$$\dot{\boldsymbol{q}} = \boldsymbol{J}^{\dagger}\dot{\boldsymbol{\sigma}} + \boldsymbol{N}\dot{\boldsymbol{q}}_0, \tag{2.8}$$

where $\dot{\boldsymbol{q}}_0 \in \mathbb{R}^n$ is an arbitrary system velocity vector, projected through the null-space operator given by

$$\boldsymbol{N} = \boldsymbol{I} - \boldsymbol{J}^{\dagger}\boldsymbol{J}. \tag{2.9}$$

Since $\boldsymbol{J}\boldsymbol{N} = \boldsymbol{0}$, the system velocity $\dot{\boldsymbol{q}}_0$ only generates motions in the null-space of the primary task Jacobian matrix, which does not affect the primary task.

2.4 Task-Priority Schemes

The most prominent task priority schemes for kinematic control are the multiple task priority (MTP) inverse kinematics framework [8] and the singularity-robust multiple task priority (SRMTP) inverse kinematics framework [9]. The MTP framework generally results in more accurate tracking behavior of lower priority tasks, but it is subject to algorithmic singularities. The SRMTP framework is robust against algorithmic singularities[1], but suffers from worse tracking accuracy of lower priority tasks. Algorithmic singularities denote all singularities that do not come from kinematic singularities, i.e. when a task Jacobian suffers a loss of rank. These types of singularities may occur in the MTP framework because of the pseudoinversion of $\boldsymbol{J}_i\boldsymbol{N}_i^A$, while the only pseudoinversions in the SRMTP framework are task Jacobians and augmented task Jacobians. However, for the general case of k tasks, the augmented task Jacobians may have linearly dependent rows even though none of the Jacobians are close to a kinematic singularity. To better understand how this may occur, consider the general case of k tasks, if two or more tasks are in conflict on priority level $1, \ldots, k-1$, the augmented Jacobian of the first $k-1$ tasks will have linearly dependent

[1] In its original definition for two tasks.

rows. Hence, for particular tasks, the SRMTP framework may also exhibit algorithmic singularities. This can be prevented by observing that the row space of the augmented Jacobian remains the same even if linearly dependent rows are removed, which entails that the null space, or orthogonal complement of the row space, remains the same when linearly dependent rows are removed.

2.5 Singularity-Robust Multiple Task-Priority Inverse Kinematics Framework

The singularity-robust multiple task priority inverse kinematics framework is a popular method for kinematic control of general robotic systems. The framework enables an arbitrary number of equality tasks, i.e. tasks with a specified desired value, to be defined, prioritized and achieved simultaneously. The SRMTP framework is a Jacobian-based method, originally proposed in [9] for velocity level redundancy resolution with two tasks. The framework was extended to an arbitrary number of tasks in [30, 10] as null space-based behavioral control. The SRMTP name was reintroduced when the approach was further extended to handle set-based tasks in [4]. In this thesis, the SRMTP name will be used.

The SRMTP framework can also be proposed for acceleration level redundancy resolution as shown in Section 2.5.2. The inverse kinematics problem is solved on the velocity or acceleration level by generating reference joint velocities or accelerations to be tracked by a dynamic controller, and if tracked successfully, fulfills multiple tasks in a prioritized order.

2.5.1 Redundancy resolution at the velocity level

For redundant robotic systems with $n > m$, the task Jacobian matrix is not square and (2.2) cannot be inverted directly to obtain the system velocity references $\dot{\boldsymbol{q}}_d(t)$. However, the Moore-Penrose pseudoinverse can be employed to

find a solution that minimizes the norm of the joint velocities [31], viz.

$$\dot{\boldsymbol{q}}_d = \boldsymbol{J}^{\dagger}\dot{\boldsymbol{\sigma}} = \boldsymbol{J}^{\mathrm{T}}\left(\boldsymbol{J}\boldsymbol{J}^{\mathrm{T}}\right)^{-1}\dot{\boldsymbol{\sigma}}, \tag{2.10}$$

where the matrix $\boldsymbol{J}^{\dagger}$ is the right pseudoinverse of $\boldsymbol{J}$. Since (2.10) is prone to drifting when integrated to obtain $\boldsymbol{q}_d$, a closed-loop inverse kinematics (CLIK) scheme is usually employed [13]. To derive such a scheme, let the task error be defined by

$$\tilde{\boldsymbol{\sigma}} = \boldsymbol{\sigma}_d - \boldsymbol{\sigma} = \boldsymbol{\sigma}_d - \boldsymbol{f}(\boldsymbol{q}). \tag{2.11}$$

The goal is to construct a suitable feedback control law which drives $\tilde{\boldsymbol{\sigma}}$ to zero. The derivative with respect to time is given by

$$\dot{\tilde{\boldsymbol{\sigma}}} = \dot{\boldsymbol{\sigma}}_d - \dot{\boldsymbol{\sigma}} \tag{2.12}$$

$$= \dot{\boldsymbol{\sigma}}_d - \boldsymbol{J}\dot{\boldsymbol{q}}. \tag{2.13}$$

In order for this equation to lead to a closed-loop inverse kinematics algorithm, the joint velocity vector $\dot{\boldsymbol{q}}$ should be related to the task error $\tilde{\boldsymbol{\sigma}}$ such that (2.13) represents a differential equation describing the error evolution over time [31]. By computing the desired system velocities from

$$\dot{\boldsymbol{q}}_d = \boldsymbol{J}^{\dagger}(\dot{\boldsymbol{\sigma}}_d + \boldsymbol{\Lambda}\tilde{\boldsymbol{\sigma}}), \tag{2.14}$$

and inserting this relationship into (2.13) under the assumption that $\dot{\boldsymbol{q}} = \dot{\boldsymbol{q}}_d$ leads to

$$\dot{\tilde{\boldsymbol{\sigma}}} = \dot{\boldsymbol{\sigma}}_d - \boldsymbol{J}\dot{\boldsymbol{q}} \tag{2.15}$$

$$= \dot{\boldsymbol{\sigma}}_d - \boldsymbol{J}\boldsymbol{J}^{\dagger}(\dot{\boldsymbol{\sigma}}_d + \boldsymbol{\Lambda}\tilde{\boldsymbol{\sigma}}) \tag{2.16}$$

$$= -\boldsymbol{\Lambda}\tilde{\boldsymbol{\sigma}}, \tag{2.17}$$

where it has been assumed that $\boldsymbol{J}$ has linearly independent rows such that $\boldsymbol{J}\boldsymbol{J}^\dagger = \boldsymbol{I}$. This is a linear system with a globally exponentially stable equilibrium point at $\tilde{\boldsymbol{\sigma}} = \boldsymbol{0}$ when $\boldsymbol{\Lambda}$ is positive definite [32].

A common approach in closed-loop inverse kinematic algorithms is to feed back the desired configurations, $\boldsymbol{q}_d$, instead of the measured configuration $\boldsymbol{q}$. This keeps the kinematic control loop separate from the dynamic control loop and is equivalent to assuming $\dot{\boldsymbol{q}} = \dot{\boldsymbol{q}}_d$ [12].

In the case of system redundancy, where $n > m$, (2.14) can be written

$$\dot{\boldsymbol{q}}_d = \boldsymbol{J}^\dagger(\dot{\boldsymbol{\sigma}}_d + \boldsymbol{\Lambda}\tilde{\boldsymbol{\sigma}}) + (\boldsymbol{I} - \boldsymbol{J}^\dagger\boldsymbol{J})\dot{\boldsymbol{q}}_0, \tag{2.18}$$

where $\dot{\boldsymbol{q}}_0 \in \mathbb{R}^n$ is an arbitrary system velocity vector. These velocities are projected through the null-space operator $(\boldsymbol{I} - \boldsymbol{J}\boldsymbol{J}^\dagger)$, which means that they can only generate velocities in the null space of the task Jacobian matrix. Therefore, these velocities do not affect that of the primary task [4].

From here on let $\dot{\boldsymbol{\sigma}}_r = \dot{\boldsymbol{\sigma}}_d + \boldsymbol{\Lambda}\tilde{\boldsymbol{\sigma}}$. A generalization of this framework to k equality tasks arranged by priority leads to [12]

$$\dot{\boldsymbol{q}}_d = \underbrace{\boldsymbol{J}_1^\dagger \dot{\boldsymbol{\sigma}}_{1,r}}_{\dot{\boldsymbol{q}}_{1,d}} + \boldsymbol{N}_1 \underbrace{\boldsymbol{J}_2^\dagger \dot{\boldsymbol{\sigma}}_{2,r}}_{\dot{\boldsymbol{q}}_{2,d}} + \ldots + \boldsymbol{N}^A_{12..(k-1)} \underbrace{\boldsymbol{J}_k^\dagger \dot{\boldsymbol{\sigma}}_{k,r}}_{\dot{\boldsymbol{q}}_{k,d}}, \tag{2.19}$$

where $\boldsymbol{N}^A_{12..(k-1)}$ is the null space operator of the augmented Jacobian obtained by stacking the $k-1$ higher priority tasks, viz.

$$\boldsymbol{J}^A_{12..(k-1)} = \begin{bmatrix} \boldsymbol{J}_1 \\ \boldsymbol{J}_2 \\ \vdots \\ \boldsymbol{J}_{k-1} \end{bmatrix}, \tag{2.20}$$

$$\boldsymbol{N}^A_{12..(k-1)} = \left(\boldsymbol{I} - \left(\boldsymbol{J}^A_{k-1}\right)^\dagger \boldsymbol{J}^A_{k-1}\right). \tag{2.21}$$

Since $\boldsymbol{J}_{12..(k-1)}^{A^\dagger}$ is a Moore-Penrose pseudoinverse the following holds

$$\boldsymbol{J}_i \boldsymbol{N}_{12..(k-1)}^{A} = \boldsymbol{0}, \tag{2.22}$$

for $i = 1, \ldots, k-1$, which ensures that lower priority tasks do not generate velocities affecting higher priority tasks. As mentioned in Section 2.3, it is evident from (2.20) that if two or more of the task Jacobians in the augmented Jacobian are conflicting, the augmented Jacobian will have linearly dependent rows. Hence, it suffers a loss of rank and $\boldsymbol{J}_{k-1}^{A} \left(\boldsymbol{J}_{k-1}^{A}\right)^{\mathrm{T}}$ is not invertible, which is to say that the pseudoinverse cannot be computed from the formula

$$\boldsymbol{J}^\dagger = \boldsymbol{J}^{\mathrm{T}} \left(\boldsymbol{J}\boldsymbol{J}^{\mathrm{T}}\right)^{-1}. \tag{2.23}$$

However, linearly dependent rows may be removed from the augmented Jacobian without affecting the row space. Therefore, the null-space defined by $\boldsymbol{I} - \boldsymbol{J}^\dagger \boldsymbol{J}$ remains the same.

2.5.2 Redundancy resolution at the acceleration level

A drawback of velocity level redundancy resolution is that joint space acceleration references have to computed by numeric differentiation of the desired joint velocity. Moreover, task space acceleration references cannot be included, which limits tracking performance. For second order systems such as a robotic manipulator, acceleration-based control is a more natural approach since it computes desired acceleration references directly, which enables the use of more advanced control schemes [33]. By utilizing the Moore-Penrose pseudoinverse, (2.3) can be solved for $\ddot{\boldsymbol{q}}$, viz.

$$\ddot{\boldsymbol{q}} = \boldsymbol{J}^\dagger \left(\ddot{\boldsymbol{\sigma}} - \dot{\boldsymbol{J}}\dot{\boldsymbol{q}}\right). \tag{2.24}$$

Employing a closed-loop inverse kinematics scheme yields

$$\ddot{\boldsymbol{q}}_d = \boldsymbol{J}^{\dagger}\left(\ddot{\boldsymbol{\sigma}}_d + \boldsymbol{K}_d\dot{\tilde{\boldsymbol{\sigma}}} + \boldsymbol{K}_p\tilde{\boldsymbol{\sigma}} - \dot{\boldsymbol{J}}\dot{\boldsymbol{q}}\right), \tag{2.25}$$

where $\tilde{\boldsymbol{\sigma}} = \boldsymbol{\sigma}_d - \boldsymbol{\sigma}$ denotes the task error. For redundant systems the SRMTP framework can be proposed for the acceleration level as follows [33]

$$\begin{aligned}\ddot{\boldsymbol{q}}_d = \boldsymbol{J}_1^{\dagger}\left(\ddot{\boldsymbol{\sigma}}_{1,r} - \dot{\boldsymbol{J}}_1\dot{\boldsymbol{q}}\right) + \boldsymbol{N}_1\boldsymbol{J}_2^{\dagger}\left(\ddot{\boldsymbol{\sigma}}_{2,r} - \dot{\boldsymbol{J}}_2\dot{\boldsymbol{q}}\right) + \dots \\ + \boldsymbol{N}_{12..(k-1)}^{A}\boldsymbol{J}_k^{\dagger}\left(\ddot{\boldsymbol{\sigma}}_{k,r} - \dot{\boldsymbol{J}}_k\dot{\boldsymbol{q}}\right),\end{aligned} \tag{2.26}$$

where $\boldsymbol{N}_{12..(k-1)}^{A}$ is given by (2.21) and the task space acceleration reference is given by

$$\ddot{\boldsymbol{\sigma}}_r = \ddot{\boldsymbol{\sigma}}_d + \boldsymbol{K}_d\dot{\tilde{\boldsymbol{\sigma}}} + \boldsymbol{K}_p\tilde{\boldsymbol{\sigma}}. \tag{2.27}$$

The obtained acceleration references may for instance be used in combination with a feedback linearizing control law ensuring that $\ddot{\boldsymbol{q}} = \ddot{\boldsymbol{q}}_d$. Note that for acceleration-based control the problem of internal instability of the joint angles may occur if not all the DOFs of the robotic system are accounted for in (2.26). Therefore, care must be taken when specifying the operational space tasks. To ensure internal stability, a task at the lowest priority level with a trivial Jacobian can be defined which regulates $\dot{\boldsymbol{q}}$ to zero.

Chapter 3

Set-Based Kinematic Control

3.1 Introduction

Only equality tasks have been considered so far, where the goal is to ensure that the task variable converges to a desired value. Set-based tasks, however, are satisfied whenever the task variable $\sigma \in \mathbb{R}$ is contained inside of its valid set, i.e. $\sigma \in D = [\sigma_{\min}, \sigma_{\max}]$. Typical examples of set-based tasks for a robotic system are mechanical joint limit avoidance, obstacle avoidance and manipulability index tasks. These types of tasks can only be introduced as low priority tasks when transformed into equality tasks in the MTP and SRMTP frameworks. Transforming set-based tasks into high-priority equality tasks is not feasible as it would consume too many DOFs, effectively overconstraining the system. This implies that safety related set-based tasks have to be introduced as low-priority tasks, which is highly undesirable since the satisfaction of low-priority tasks cannot be guaranteed.

In this section two different frameworks for set-based kinematic control is

presented. The first approach is an extension of the SRMTP framework which results in inherently discontinuous velocity references, while the second approach exploits regularization methods to obtain continuous velocity references at all times. Moreover, the set-based SRMTP framework is extended to multidimensional tasks as well as to acceleration level redundancy resolution.

3.2 Set-Based SRMTP Framework

In [4] the SRMTP framework is extended to handle scalar set-based tasks within a velocity level redundancy resolution scheme as follows. Whenever the set-based task variable σ is contained within its valid set, the set-based task is considered inactive and ignored by the inverse kinematics algorithm. If σ reaches the boundary of its valid set and the time derivative $\dot{\sigma}$ points outside of the valid set, the task is activated and introduced in the task priority framework as an additional equality task to ensure that the task stays on the boundary of its valid set. Whenever other tasks maintain σ in its valid set, the set-based task is deactivated. The activation and deactivation of set-based tasks is determined by the extended tangent cone function defined by

$$T_{\mathbb{R},D}(\sigma) = \begin{cases} [0, \infty), & \sigma \leq \sigma_{\min} \\ \mathbb{R}, & \sigma \in (\sigma_{\min}, \sigma_{\max}) \\ (-\infty, 0], & \sigma \geq \sigma_{\max} \end{cases} \tag{3.1}$$

Since $\sigma \in D$ at $t = t_0$ and $\dot{\sigma} \in T_{\mathbb{R},D}$ for all $t \geq t_0$ implies that $\sigma \in D$ for all $t \geq t_0$, the desired behavior of the set-based task is obtained by maintaining $\dot{\sigma} \in T_{\mathbb{R},D}$. If this set-based task is introduced into the SRMTP framework as a high priority task it will be satisfied for all time, while all compatible lower priority tasks will converge to their desired values. However, the satisfaction of lower priority set-based tasks cannot be guaranteed due to the presence of potentially conflicting higher priority equality tasks.

For a system with j set-based tasks there are 2^j different combinations of set-based tasks being active or inactive. These combinations are referred to as modes of the system, and the result is an algorithm which dynamically switches between joint velocity reference solutions, resulting in inherently discontinuous velocity references. However, the system is stable through the switching process, and the equality task errors are proven to converge to zero for compatible regulation tasks where the gains are chosen appropriately [4]. The algorithm that checks whether a given set-based task is in the extended tangent cone is presented in Algorithm 1.

Algorithm 1 A Boolean function that checks if $\dot{\sigma} \in T_{\mathbb{R},D}(\sigma)$

Input: $\dot{\sigma}$, σ, $\sigma_{\min}$, $\sigma_{\max}$
1: **if** $\sigma_{\min} < \sigma < \sigma_{\max}$ **then**
2: **return** True
3: **else if** $\sigma \leq \sigma_{\min}$ and $\dot{\sigma} \geq 0$ **then**
4: **return** True
5: **else if** $\sigma \geq \sigma_{\max}$ and $\dot{\sigma} \leq 0$ **then**
6: **return** True
7: **else**
8: **return** False
9: **end if**

Note that the input $\dot{\sigma}$ in Algorithm 1 is not a fed back quantity from the last time step, one instead checks if a less restrictive velocity reference solution will cause the derivative of the set-based task to leave the extended tangent cone. To illustrate this consider a case with one high-priority set-based task and an arbitrary number of equality based tasks. The idea is to compute the velocity references corresponding to mode 1, $\boldsymbol{f}_1$, where the set-based task is inactive, and then using $\dot{\sigma} = \boldsymbol{J}\boldsymbol{f}_1$ as an input to the algorithm. This observes the effect in σ by keeping the set-based task inactive, if the calculated velocity references corresponding to mode 1 implies that $\dot{\sigma} \notin T_{\mathbb{R},D}(\sigma)$, then mode 2 is activated.

3.2.1 Extension to acceleration level redundancy resolution

While not explicitly mentioned in [4], redundancy resolution may also be performed at the acceleration level in the set-based SRMTP framework. Algorithm 1 remains unchanged, but the predicted task velocity is now obtained from

$$\dot{\boldsymbol{\sigma}} = \boldsymbol{J}\dot{\boldsymbol{q}}_d, \tag{3.2}$$

where $\dot{\boldsymbol{q}}_d$ represents the numerical integration of a less restrictive acceleration reference $\ddot{\boldsymbol{q}}_d$ where the set-based tasks in question is inactive, viz.

$$\dot{\boldsymbol{q}}_d(t_k) = \dot{\boldsymbol{q}}(t_{k-1}) + \Delta t\ddot{\boldsymbol{q}}_d(t_{k-1}). \tag{3.3}$$

Furthermore, since set-based tasks are frozen at the boundary, the task space acceleration reference given by (2.27) reduces to

$$\ddot{\boldsymbol{\sigma}}_r = -\boldsymbol{K}_d\dot{\boldsymbol{\sigma}}, \tag{3.4}$$

for set-based tasks.

3.3 Extension to Multidimensional Set-Based Tasks

3.3.1 Introducing the activation matrix

The approach in Section 3.2 is extended to vector tasks $\boldsymbol{\sigma}(t) \in \mathbb{R}^m$ by constructing an activation matrix $\boldsymbol{A} \in \mathbb{R}^{p\times m}$, where $p \leq m$ is the number of active set-based scalar tasks in $\boldsymbol{\sigma}$. Consider a single multidimensional set-based task, $\boldsymbol{\sigma} \in \mathbb{R}^m$, described by the differential relationship

$$\dot{\boldsymbol{\sigma}} = \boldsymbol{J}\dot{\boldsymbol{q}}, \tag{3.5}$$

where $\boldsymbol{J} \in \mathbb{R}^{m \times n}$ and $\dot{\boldsymbol{q}} \in \mathbb{R}^n$. The Jacobian matrix with respect to the multidimensional set-based task can be expressed as follows

$$\boldsymbol{J} = \begin{bmatrix} \boldsymbol{J}_1 \\ \boldsymbol{J}_2 \\ \vdots \\ \boldsymbol{J}_m \end{bmatrix}, \tag{3.6}$$

where $\boldsymbol{J}_i \in \mathbb{R}^{1 \times n}$ for $i = 1, \ldots, m$. Define the activation matrix

$$\boldsymbol{A} = \begin{bmatrix} \boldsymbol{n}_1^{\mathrm{T}} \\ \boldsymbol{n}_2^{\mathrm{T}} \\ \vdots \\ \boldsymbol{n}_p^{\mathrm{T}} \end{bmatrix} \in \mathbb{R}^{p \times m}, \tag{3.7}$$

where $\boldsymbol{n}_j$ is a unit vector extracting a specific row of the Jacobian. The activation matrix extracts the p rows of the Jacobian corresponding to scalar components in $\boldsymbol{\sigma}$ with $\dot{\sigma}_i \notin T_{\mathbb{R},D}(\sigma_i)$. Hence, the Jacobian of the active set-based task becomes $\bar{\boldsymbol{J}} = \boldsymbol{A}\boldsymbol{J}$. The activation matrix is constructed from an $m \times m$ identity matrix by removing row i if $\dot{\sigma}_i \in T_{\mathbb{R},D}(\sigma_i)$ according to Algorithm 1. Removing all rows corresponds to zero active set-based tasks at this priority level. In other words, the activation matrix extracts the rows of the Jacobian matrix corresponding to the scalar components in $\boldsymbol{\sigma}$ that would leave the extended tangent cone in the following time step if not actively accounted for. This scheme enables the stacking of several scalar set-based tasks on the same priority level, which was not possible in the framework proposed in [4].

3.3.2 Example: High-priority set-based tasks

Consider a system with one high priority set-based task, $\boldsymbol{\sigma}_a \in \mathbb{R}^{m_1}$, and k lower priority equality task $\boldsymbol{\sigma}_1 \in \mathbb{R}^{m_2}, \boldsymbol{\sigma}_2 \in \mathbb{R}^{m_3}, \ldots, \boldsymbol{\sigma}_k \in \mathbb{R}^{m_{k+1}}$,. When none of the set-based tasks are active, the corresponding joint velocity references are computed from

$$\dot{\boldsymbol{q}}_d = \boldsymbol{J}_1^{\dagger}(\dot{\boldsymbol{\sigma}}_{1,d} + \boldsymbol{\Lambda}_1\tilde{\boldsymbol{\sigma}}_1) + \boldsymbol{N}_1\boldsymbol{J}_2^{\dagger}(\dot{\boldsymbol{\sigma}}_{2,d} + \boldsymbol{\Lambda}_2\tilde{\boldsymbol{\sigma}}_2) \tag{3.8}$$

$$+ \ldots + \boldsymbol{N}^A_{1..(k-1)}\boldsymbol{J}_k^{\dagger}(\dot{\boldsymbol{\sigma}}_{k,d} + \boldsymbol{\Lambda}_k\tilde{\boldsymbol{\sigma}}_k). \tag{3.9}$$

If one or more scalar set-based tasks in $\boldsymbol{\sigma}_a$ are active, the joint velocity references are computed from

$$\dot{\boldsymbol{q}}_d = \boldsymbol{N}_a\boldsymbol{J}_1^{\dagger}(\dot{\boldsymbol{\sigma}}_{1,d} + \boldsymbol{\Lambda}_1\tilde{\boldsymbol{\sigma}}_1) + \boldsymbol{N}^A_{a1}\boldsymbol{J}_2^{\dagger}(\dot{\boldsymbol{\sigma}}_{2,d} + \boldsymbol{\Lambda}_2\tilde{\boldsymbol{\sigma}}_2) \tag{3.10}$$

$$+ \ldots + \boldsymbol{N}^A_{a1..(k-1)}\boldsymbol{J}_k^{\dagger}(\dot{\boldsymbol{\sigma}}_{k,d} + \boldsymbol{\Lambda}_k\tilde{\boldsymbol{\sigma}}_k), \tag{3.11}$$

where the null-space projector of the active set-based tasks is given by

$$\boldsymbol{N}_a = \left(\boldsymbol{I} - \bar{\boldsymbol{J}}_a^{\dagger}\bar{\boldsymbol{J}}_a\right) \tag{3.12}$$

$$= \left(\boldsymbol{I} - \left(\boldsymbol{A}_a\boldsymbol{J}_a\right)^{\dagger}\boldsymbol{A}_a\boldsymbol{J}_a\right), \tag{3.13}$$

while the augmented null-space operators for the lower priority tasks are obtained by stacking the Jacobians of all the higher priority tasks as shown in (2.20) and (2.21).

3.3.3 Implementation aspects

Algorithm 1 is modified to handle multidimensional inputs as shown in Algorithm 2. Note that Algorithm 2 returns the activation matrix defined in Section 3.3.1 instead of a Boolean variable representing if the scalar set-based task has its derivative in the extended tangent cone.

Algorithm 2 Computing the activation matrix $\boldsymbol{A}$

Input: $\dot{\boldsymbol{\sigma}}$, $\boldsymbol{\sigma}$, $\boldsymbol{\sigma}_{\min}$, $\boldsymbol{\sigma}_{\max}$
Output: $\boldsymbol{A}$
$m = \text{size}(\sigma, 1)$
$\boldsymbol{A} = \boldsymbol{I}_{m \times m}$
for $i = 1$ to m **do**
  if $\sigma_{\min,i} < \sigma_i < \sigma_{\max,i}$ **then**
    Remove row i from $\boldsymbol{A}$
  else if $\sigma_i \leq \sigma_{\min,i}$ and $\dot{\sigma}_i \geq 0$ **then**
    Remove row i from $\boldsymbol{A}$
  else if $\sigma_i \geq \sigma_{\max,i}$ and $\dot{\sigma}_i \leq 0$ **then**
    Remove row i from $\boldsymbol{A}$
  end if
end for
return $\boldsymbol{A} \in \mathbb{R}^{p \times m}$

3.4 Inequality Control Objectives, Activations and Transitions (iCAT) Framework

The iCAT framework was formalized in [5] as a way of of integrating set-based tasks within a task-priority framework, without incurring practical discontinuities through the activation and deactivation of tasks. The framework is a kinematic controller which resolves redundancy at the velocity level. Hence, the output of the controller is the system reference velocities to be tracked by the dynamic controller. In order to obtain continuous reference velocities at all times, a novel regularization technique is combined with a singular value oriented regularization and a final minimization of the reference velocities which ensures that the reference velocities are continuous while set-based tasks are being activated and deactivated.

3.4.1 Activation functions

As opposed to the set-based SRMTP framework, the iCAT framework employs continuous sigmoidal activation functions to activate and deactivate set-based

tasks. An activation function $a(\sigma) \in [0, 1]$, constructed for a set-based task $\sigma \in \mathbb{R}$ defined by $\sigma \leq \sigma_{\max}$ is given by

$$a(\sigma) = \begin{cases} 1, & \sigma > \sigma_{\max} \\ s(\sigma), & \sigma_{\max} - \beta \leq \sigma \leq \sigma_{\max} \\ 0, & \sigma_i < \sigma_{\max} - \beta \end{cases} \tag{3.14}$$

where $s(\sigma)$ is any sigmoid function with a continuous behavior from 0 to 1 when $\sigma_{\max} - \beta \leq \sigma \leq \sigma_{\max}$. The value of β creates a transition zone in which the set-based task is satisfied, but where the activation value is greater than zero. Hence, set-based tasks are actively accounted for in the kinematic controller before it reaches the boundary of its valid set. The transition zone is necessary to prevent chattering around the boundary of the valid set. Note that a similar activation function can be defined for set-based tasks defined by $\sigma \geq \sigma_{\min}$. Moreover, for a set-based task defined by $\sigma_{\min} \leq \sigma \leq \sigma_{\max}$, the sum of the activation functions for the tasks $\sigma_{\min} \leq \sigma$ and $\sigma \leq \sigma_{\max}$ can be used as an activation function if the minimum and maximum value are sufficiently spaced, i.e. $\sigma_{\min} + \beta < \sigma_{\max} - \beta$. In case of a multidimensional task, the activation matrix $\boldsymbol{A}$ is defined as a diagonal matrix of activation functions corresponding to each scalar component in the multidimensional task. As a final remark, the activation functions for equality tasks are clearly $a \equiv 1$.

3.4.2 Set-based and equality tasks

A feedback reference rate is assigned to each scalar control objective, viz.

$$\dot{\bar{\sigma}} = \gamma \left(\sigma^* - \sigma \right), \tag{3.15}$$

representing a control law driving the task variable $\sigma(\boldsymbol{q})$ toward a point σ^*. In case of a set-based task, σ^* represents an arbitrary point within the subset of the valid set not contained in the transition zone, otherwise σ^* represents the desired value of the equality task. Furthermore, $\gamma > 0$ is a positive gain proportional

to the desired convergence rate for the task variable. Whenever the activation value of the corresponding task is non-zero, the task is actively accounted for in the kinematic controller. For set-based tasks, the reference point σ^* cannot be inside the transition zone as claimed by the authors in [5], as this would lead to overconstraining the system since the task would never become inactive, i.e. the activation function would never obtain a value of zero.

3.4.3 Kinematic control law

The kinematic control law combines singular value oriented regularization with a novel regularization called task oriented regularization and a final minimization of the control vector in order to remove any discontinuities. The drawback is that the projection matrix is no longer orthogonal whenever some activation values are different from zero or one. For a combination of k set-based and equality tasks the control law is given by [34]

$$\boldsymbol{\rho}_0 = \mathbf{0}, \tag{3.16}$$

$$\boldsymbol{Q}_0 = \boldsymbol{I}, \tag{3.17}$$

then for $i = 1, \ldots, k$

$$\boldsymbol{W}_i = \boldsymbol{J}_i \boldsymbol{Q}_{i-1} \left(\boldsymbol{J}_i \boldsymbol{Q}_{i-1}\right)^{\#, \boldsymbol{A}_i, \boldsymbol{Q}_{i-1}}, \tag{3.18}$$

$$\boldsymbol{Q}_i = \boldsymbol{Q}_{i-1} \left(\boldsymbol{I} - \left(\boldsymbol{J}_i \boldsymbol{Q}_{i-1}\right)^{\#, \boldsymbol{A}_i, \boldsymbol{I}} \boldsymbol{J}_i \boldsymbol{Q}_{i-1}\right), \tag{3.19}$$

$$\boldsymbol{\rho}_i = \boldsymbol{\rho}_{i-1} + \boldsymbol{Q}_{i-1} \left(\boldsymbol{J}_i \boldsymbol{Q}_{i-1}\right)^{\#, \boldsymbol{A}_i, \boldsymbol{I}} \boldsymbol{W}_i \left(\dot{\tilde{\boldsymbol{\sigma}}}_i - \boldsymbol{J}_i \boldsymbol{\rho}_{i-1}\right), \tag{3.20}$$

where the operator $\boldsymbol{X}^{\#, \boldsymbol{A}, \boldsymbol{Q}}$ is defined by

$$\boldsymbol{X}^{\#, \boldsymbol{A}, \boldsymbol{Q}} = \left(\boldsymbol{X}^{\mathrm{T}} \boldsymbol{A} \boldsymbol{X} + \eta \left(\boldsymbol{I} - \boldsymbol{Q}\right)^{\mathrm{T}} \left(\boldsymbol{I} - \boldsymbol{Q}\right) + \boldsymbol{V}^{\mathrm{T}} \boldsymbol{P} \boldsymbol{V}\right)^{\dagger} \boldsymbol{X}^{\mathrm{T}} \boldsymbol{A} \boldsymbol{A}, \tag{3.21}$$

with $\eta > 0$ and where $\boldsymbol{V}$ is the right orthonormal matrix of the singular value decomposition of

$$\boldsymbol{X}^{\mathrm{T}}\boldsymbol{A}\boldsymbol{X} + \eta\left(\boldsymbol{I} - \boldsymbol{Q}\right)^{\mathrm{T}}\left(\boldsymbol{I} - \boldsymbol{Q}\right). \tag{3.22}$$

Furthermore, $\boldsymbol{P}$ is a diagonal singular value oriented regularization matrix, where each diagonal element is a bell-shaped, finite support function of the corresponding singular value, or zero if the corresponding singular value does not exist. In this thesis, the diagonal elements are normally distributed with zero mean and a tunable standard deviation parameter. After the final iteration, the system velocity reference is given by

$$\dot{\boldsymbol{q}}_d = \boldsymbol{\rho}_k. \tag{3.23}$$

Chapter 4

Operational Space Control

4.1 Introduction

Operational space control [21] presents an alternative to controlling a robotic system in joint space by transforming the dynamic equations of motion into operational space, where forces and torques are computed directly instead of relying on a kinematic controller in combination with a dynamic controller. Redundancy is resolved at the torque level by defining null-space operators which ensure that torques generated by lower priority tasks do not generate accelerations or forces affecting the task dynamics of higher priority tasks. This approach enables the use of any kind of force control scheme and is thus highly suitable for robotic systems interacting with the environment.

4.2 Operational Space Dynamics

For a generic task $\boldsymbol{\sigma}_1(\boldsymbol{q}) \in \mathbb{R}^{m_1}$ the following relationships hold

$$\boldsymbol{\sigma}_1 = \boldsymbol{f}_1(\boldsymbol{q}), \tag{4.1}$$

$$\dot{\boldsymbol{\sigma}}_1 = \boldsymbol{J}_1(\boldsymbol{q})\dot{\boldsymbol{q}}, \tag{4.2}$$

$$\ddot{\boldsymbol{\sigma}}_1 = \boldsymbol{J}_1(\boldsymbol{q})\ddot{\boldsymbol{q}} + \dot{\boldsymbol{J}}_1(\boldsymbol{q}, \dot{\boldsymbol{q}})\dot{\boldsymbol{q}}. \tag{4.3}$$

The dynamic equations of motion for a robotic manipulator are given by [35]

$$\boldsymbol{M}(\boldsymbol{q})\ddot{\boldsymbol{q}} + \boldsymbol{C}(\boldsymbol{q}, \dot{\boldsymbol{q}})\dot{\boldsymbol{q}} + \boldsymbol{g}(\boldsymbol{q}) = \boldsymbol{\tau} \tag{4.4}$$

$$\implies \ddot{\boldsymbol{q}} = \boldsymbol{M}^{-1}\left(\boldsymbol{\tau} - \boldsymbol{C}(\boldsymbol{q}, \dot{\boldsymbol{q}})\dot{\boldsymbol{q}} - \boldsymbol{g}(\boldsymbol{q})\right). \tag{4.5}$$

Mapping the generalized torque vector into a generalized force vector through the relation

$$\boldsymbol{\tau} = \boldsymbol{J}^{\mathrm{T}}\boldsymbol{F}, \tag{4.6}$$

inserting (4.5) into (4.3) and omitting dependencies for readability yields

$$\ddot{\boldsymbol{\sigma}}_1 = \boldsymbol{J}_1\boldsymbol{M}^{-1}\left(\boldsymbol{J}_1^{\mathrm{T}}\boldsymbol{F}_1 - \boldsymbol{C}\dot{\boldsymbol{q}} - \boldsymbol{g}\right) + \dot{\boldsymbol{J}}_1\dot{\boldsymbol{q}}. \tag{4.7}$$

Furthermore, by defining the inertia matrix associated with task 1 by

$$\boldsymbol{M}_1 = \left(\boldsymbol{J}_1\boldsymbol{M}^{-1}\boldsymbol{J}_1^{\mathrm{T}}\right)^{-1} \in \mathbb{R}^{m_1 \times m_1}, \tag{4.8}$$

and multiplying both sides of (4.7) by $\boldsymbol{M}_1$, the operational space dynamics are obtained as

$$\boldsymbol{M}_1\ddot{\boldsymbol{\sigma}}_1 + \boldsymbol{M}_1\left(\boldsymbol{J}_1\boldsymbol{M}^{-1}\boldsymbol{C}\dot{\boldsymbol{q}} - \dot{\boldsymbol{J}}_1\dot{\boldsymbol{q}}\right) + \boldsymbol{M}_1\boldsymbol{J}_1\boldsymbol{M}^{-1}\boldsymbol{g} = \boldsymbol{F}_1 \tag{4.9}$$

$$\boldsymbol{M}_1\ddot{\boldsymbol{\sigma}}_1 + \boldsymbol{c}_1 + \boldsymbol{g}_1 = \boldsymbol{F}_1. \tag{4.10}$$

In case of system redundancy, the generalized torque vector may be decomposed into a torque corresponding to the primary task and another torque acting in the null space of the primary task as follows

$$\boldsymbol{\tau} = \boldsymbol{J}_1^{\mathrm{T}} \boldsymbol{F}_1 + \boldsymbol{N}_2 \boldsymbol{\tau}_0, \tag{4.11}$$

where $\boldsymbol{\tau}_0$ is an arbitrary torque and the null space operator is given by

$$\boldsymbol{N}_2 = \boldsymbol{I} - \boldsymbol{J}_1^{\mathrm{T}} \bar{\boldsymbol{J}}_1^{\ \mathrm{T}}, \tag{4.12}$$

with

$$\bar{\boldsymbol{J}}_1 = \boldsymbol{M}^{-1} \boldsymbol{J}_1^{\mathrm{T}} \boldsymbol{M}_1 \tag{4.13}$$

$$= \boldsymbol{M}^{-1} \boldsymbol{J}_1^{\mathrm{T}} \left(\boldsymbol{J}_1 \boldsymbol{M}^{-1} \boldsymbol{J}_1^{\mathrm{T}} \right)^{-1} \in \mathbb{R}^{n \times m_1}. \tag{4.14}$$

$\bar{\boldsymbol{J}}_1$ is known as the dynamically consistent pseudoinverse of $\boldsymbol{J}_1$, which is a weighted pseudoinverse of $\boldsymbol{J}$ where the weight is the inverse of the inertia matrix [21].

4.3 Generalized Null Space Operator

In order to control an arbitrary number of tasks arranged in priority, the null space operator in (4.12) needs to be extended to an arbitrary number of priority levels. In [22] a null space operator for the $n-1$ previous tasks is defined by

$$\boldsymbol{N}_k^{\mathrm{T}} = \boldsymbol{I} - \sum_{i=1}^{k-1} \boldsymbol{M}^{-1} \boldsymbol{N}_i \boldsymbol{J}_i^{\mathrm{T}} \left(\boldsymbol{J}_i \boldsymbol{N}_i^{\mathrm{T}} \boldsymbol{M}^{-1} \boldsymbol{N}_i \boldsymbol{J}_i^{\mathrm{T}} \right)^{-1} \boldsymbol{J}_i \boldsymbol{N}_i^{\mathrm{T}} \tag{4.15}$$

$$\boldsymbol{N}_k = \boldsymbol{I} - \sum_{i=1}^{k-1} \boldsymbol{N}_i \boldsymbol{J}_i^{\mathrm{T}} \left(\boldsymbol{J}_i \boldsymbol{N}_i^{\mathrm{T}} \boldsymbol{M}^{-1} \boldsymbol{N}_i \boldsymbol{J}_i^{\mathrm{T}} \right)^{-1} \boldsymbol{J}_i \boldsymbol{N}_i^{\mathrm{T}} \boldsymbol{M}^{-1}. \tag{4.16}$$

Since the null space operator satisfies the idempotent property $\boldsymbol{N}^2 = \boldsymbol{N}$ as well as $\boldsymbol{N}^{\mathrm{T}}\boldsymbol{M}^{-1} = \boldsymbol{M}^{-1}\boldsymbol{N}$, (4.16) can be reformulated recursively as

$$\boldsymbol{N}_1 = \boldsymbol{I}, \tag{4.17}$$

$$\boldsymbol{N}_{k+1} = \boldsymbol{N}_k \left(\boldsymbol{I} - \boldsymbol{J}_k^{\mathrm{T}} \left(\boldsymbol{J}_k \boldsymbol{M}^{-1} \boldsymbol{N}_k \boldsymbol{J}_k^{\mathrm{T}} \right)^{-1} \boldsymbol{J}_k \boldsymbol{M}^{-1} \boldsymbol{N}_k \right), \tag{4.18}$$

$$\boldsymbol{N}_{k+1} = \left(\boldsymbol{I} - \boldsymbol{N}_k \boldsymbol{J}_k^{\mathrm{T}} \left(\boldsymbol{J}_k \boldsymbol{M}^{-1} \boldsymbol{N}_k \boldsymbol{J}_k^{\mathrm{T}} \right)^{-1} \boldsymbol{J}_k \boldsymbol{M}^{-1} \right) \boldsymbol{N}_k. \tag{4.19}$$

The proof is provided in Appendix A. Moreover, by defining

$$\boldsymbol{M}_i = \left(\boldsymbol{J}_i \boldsymbol{M}^{-1} \boldsymbol{N}_{P(i)} \boldsymbol{J}_i^{\mathrm{T}} \right)^{-1}, \tag{4.20}$$

$$\bar{\boldsymbol{J}}_i = \boldsymbol{M}^{-1} \boldsymbol{J}_i^{\mathrm{T}} \boldsymbol{M}_i, \tag{4.21}$$

the null-space projection equations may be rewritten

$$\boldsymbol{N}_1 = \boldsymbol{I}, \tag{4.22}$$

$$\boldsymbol{N}_{k+1} = \boldsymbol{N}_k \left(\boldsymbol{I} - \boldsymbol{J}_k^{\mathrm{T}} \bar{\boldsymbol{J}}_k^{\mathrm{T}} \boldsymbol{N}_k \right), \tag{4.23}$$

$$\boldsymbol{N}_{k+1} = \left(\boldsymbol{I} - \boldsymbol{N}_k \boldsymbol{J}_k^{\mathrm{T}} \bar{\boldsymbol{J}}_k^{\mathrm{T}} \right) \boldsymbol{N}_k. \tag{4.24}$$

This null space operator is dynamically consistent, satisfying $\boldsymbol{J}_i \boldsymbol{M}^{-1} \boldsymbol{N}_j = \boldsymbol{0}$, for all $i < j$. Moreover, it also satisfies $\boldsymbol{N}_j \boldsymbol{J}_i^{\mathrm{T}} = \boldsymbol{0}$ for all $i < j$. The proofs of these properties are shown in Appendix A. These are the same null space operator properties that were used in [26] in order to show that the priority among tasks holds in steady state, even if there are modeling errors in the inertia matrix used to compute the null space operators.

4.4 Consistency of Null Space Operators

The operational space dynamics for a specific task can be obtained by pre-multiplying (4.4) by the dynamically consistent pseudoinverse of the Jacobian

corresponding to the task in question. Therefore, a sufficient condition to ensure that forces generated by lower priority tasks have no effect on higher priority tasks in any static equilibrium [25]

$$\bar{\boldsymbol{J}}_i^{\mathrm{T}} \boldsymbol{N}_j = \boldsymbol{0}, \tag{4.25}$$

for $i < j$ in any steady state with $\dot{\boldsymbol{q}} = \ddot{\boldsymbol{q}} = \boldsymbol{0}$. A null-space operator satisfying this property is known as a statically consistent null-space operator.

A stronger condition is that of dynamic consistency, which entails that lower priority tasks cannot generate acceleration effects in the operational spaces of higher priority tasks. A null space operator is said to be dynamically consistent if it satisfies

$$\boldsymbol{J}_i \boldsymbol{M}^{-1} \boldsymbol{N}_j = \boldsymbol{0}, \tag{4.26}$$

for all $i < j$. For the null space operator defined in (4.12) this property is easily verified

$$\begin{aligned}
\boldsymbol{J}_1 \boldsymbol{M}^{-1} \boldsymbol{N}_2 &= \boldsymbol{J}_1 \boldsymbol{M}^{-1} \left(\boldsymbol{I} - \boldsymbol{J}_1^{\mathrm{T}} \bar{\boldsymbol{J}}_1^{\mathrm{T}} \right) && (4.27)\\
&= \boldsymbol{J}_1 \boldsymbol{M}^{-1} - \boldsymbol{J}_1 \boldsymbol{M}^{-1} \boldsymbol{J}_1^{\mathrm{T}} \boldsymbol{M}_1 \boldsymbol{J}_1 \boldsymbol{M}^{-1} && (4.28)\\
&= \boldsymbol{J}_1 \boldsymbol{M}^{-1} - \boldsymbol{J}_1 \boldsymbol{M}^{-1} \boldsymbol{J}_1^{\mathrm{T}} \left(\boldsymbol{J}_1 \boldsymbol{M}^{-1} \boldsymbol{J}_1^{\mathrm{T}} \right)^{-1} \boldsymbol{J}_1 \boldsymbol{M}^{-1} && (4.29)\\
&= \boldsymbol{0}. && (4.30)
\end{aligned}$$

Another desirable property of this particular null space operator is that

$$\begin{aligned}
\boldsymbol{N}_2 \boldsymbol{J}_1^{\mathrm{T}} &= (\boldsymbol{I} - \boldsymbol{J}_1^{\mathrm{T}} \bar{\boldsymbol{J}}_1^{\mathrm{T}}) \boldsymbol{J}_1^{\mathrm{T}} && (4.31)\\
&= \boldsymbol{J}_1^{\mathrm{T}} - \boldsymbol{J}_1^{\mathrm{T}} \left(\boldsymbol{J}_1 \boldsymbol{M}^{-1} \boldsymbol{J}_1^{\mathrm{T}} \right)^{-1} \boldsymbol{J}_1 \boldsymbol{M}^{-1} \boldsymbol{J}_1^{\mathrm{T}} && (4.32)\\
&= \boldsymbol{0}, && (4.33)
\end{aligned}$$

holds for any values used to compute inertia matrix, as long as it is positive

definite. This property can be utilized in order to prove static consistency of the null space operator even if the inertia matrix used to compute it contains modeling errors [26]. This entails that tasks at different priority levels are not coupled in steady-state, since the lower priority task does not generate interfering forces in the operational space of the higher priority task. Moreover, all steady state task errors of the higher priority task are therefore to be attributed to modeling errors in the gravitational term.

As remarked in Section 4.3, the generalized null space operator satisfies the same properties as (4.12). Therefore, it is conjectured that the results in [26] should also hold for tasks at any priority level in a control scheme with an arbitrary number of tasks employing the generalized null space operator from Section 4.3.

4.5 Extension to n Tasks

There are different approaches in the literature in order to extend the control law defined in (4.11) to the general case of n tasks arranged in priority. In this section different control approaches are considered, mainly differing in how the inertia matrix $\boldsymbol{M}_i$ associated with each task is defined. How the inertia matrix is defined has implications for the task dynamics of task k, and by a clever choice of $\boldsymbol{M}_i$ the resulting task dynamics is linear in task k. Hence, if tasks are compatible, interference from higher priority tasks can compensated for by lower priority tasks.

4.5.1 Null space operator within the task inertia matrix

By using the dynamically consistent null space operator defined above, a generalization of the operational space framework to n tasks was proposed in [22]. A similar approach is presented here, where the main difference is that Coriolis, centrifugal and gravitational terms are compensated in joint space and not in task space. This is a more intuitive approach which reduces the dependability

on the inertia matrix. Let the control input be given by

$$\boldsymbol{\tau} = \boldsymbol{\tau}_1 + \boldsymbol{N}_2\boldsymbol{\tau}_2 + \ldots + \boldsymbol{N}_n\boldsymbol{\tau}_n + \boldsymbol{C}\dot{\boldsymbol{q}} + \boldsymbol{g}, \tag{4.34}$$

where the task specific torques are given by

$$\begin{aligned} \boldsymbol{\tau}_i &= \boldsymbol{J}_i^{\mathrm{T}}\boldsymbol{f}_i & (4.35)\\ &= \boldsymbol{J}_i^{\mathrm{T}}\boldsymbol{M}_i\boldsymbol{a}_i, & (4.36) \end{aligned}$$

and the inertia matrix associated with task i is given by

$$\boldsymbol{M}_i = \left(\boldsymbol{J}_i\boldsymbol{M}^{-1}\boldsymbol{N}_i\boldsymbol{J}_i^{\mathrm{T}}\right)^{-1}. \tag{4.37}$$

The equations of motion then reduces to

$$\boldsymbol{M}\ddot{\boldsymbol{q}} = \sum_{i=1}^{n}\boldsymbol{N}_i\boldsymbol{\tau}_i. \tag{4.38}$$

In order to obtain the task dynamics of task k the equations of motion in joint space are pre-multiplied by $\boldsymbol{J}_k\boldsymbol{M}^{-1}$, viz.

$$\begin{aligned} \boldsymbol{J}_k\boldsymbol{M}^{-1}\boldsymbol{M}\ddot{\boldsymbol{q}} &= \boldsymbol{J}_k\boldsymbol{M}^{-1}\sum_{i=1}^{n}\boldsymbol{N}_i\boldsymbol{\tau}_i & (4.39)\\ \ddot{\boldsymbol{\sigma}}_k - \dot{\boldsymbol{J}}_k\dot{\boldsymbol{q}} &= \boldsymbol{J}_k\boldsymbol{M}^{-1}\boldsymbol{\tau}_1 + \ldots + \boldsymbol{J}_k\boldsymbol{M}^{-1}\boldsymbol{N}_k\boldsymbol{\tau}_k & (4.40)\\ \ddot{\boldsymbol{\sigma}}_k - \dot{\boldsymbol{J}}_k\dot{\boldsymbol{q}} &= \boldsymbol{J}_k\boldsymbol{M}^{-1}\boldsymbol{J}_1^{\mathrm{T}}\boldsymbol{M}_1\boldsymbol{a}_1 + \ldots + \underbrace{\boldsymbol{J}_k\boldsymbol{M}^{-1}\boldsymbol{N}_k\boldsymbol{J}_k^{\mathrm{T}}\boldsymbol{M}_k}_{\boldsymbol{I}}\boldsymbol{a}_k & (4.41)\\ \ddot{\boldsymbol{\sigma}}_k - \dot{\boldsymbol{J}}_k\dot{\boldsymbol{q}} &= \boldsymbol{J}_k\boldsymbol{M}^{-1}\left(\boldsymbol{J}_1^{\mathrm{T}}\boldsymbol{M}_1\boldsymbol{a}_1 + \ldots + \boldsymbol{N}_{k-1}\boldsymbol{J}_{k-1}^{\mathrm{T}}\boldsymbol{M}_{k-1}\boldsymbol{a}_{k-1}\right) + \boldsymbol{a}_k, & (4.42) \end{aligned}$$

where the property $\boldsymbol{J}_k\boldsymbol{M}^{-1}\boldsymbol{N}_j = \boldsymbol{0}$ for $k < j$ has been employed in the first step.

4.5.1.1 Compensating for coupling effects from higher priority tasks

In [22] the task acceleration reference is embedded with compensation terms for higher priority tasks in order to fully linearize the task dynamics of every task, viz.

$$\boldsymbol{a}_i = \ddot{\boldsymbol{\sigma}}_{d,i} + \boldsymbol{K}_{d,i}\dot{\tilde{\boldsymbol{\sigma}}}_n + \boldsymbol{K}_{p,i}\tilde{\boldsymbol{\sigma}}_i - \dot{\boldsymbol{J}}_i\dot{\boldsymbol{q}} - \boldsymbol{J}_i\boldsymbol{M}^{-1}\sum_{j=1}^{i-1}\boldsymbol{N}_j\boldsymbol{J}_j^{\mathrm{T}}\boldsymbol{M}_j\boldsymbol{a}_j, \tag{4.43}$$

where $\tilde{\boldsymbol{\sigma}} = \boldsymbol{\sigma}_d - \boldsymbol{\sigma}$ represents the task error and $\boldsymbol{K}_{d,i}, \boldsymbol{K}_{p,i}$ are derivative and proportional task gains, respectively. Inserting (4.43) into (4.42) yields the linear closed loop dynamics

$$\ddot{\tilde{\boldsymbol{\sigma}}}_k + \boldsymbol{K}_{d,k}\dot{\tilde{\boldsymbol{\sigma}}}_k + \boldsymbol{K}_{p,k}\tilde{\boldsymbol{\sigma}}_k = \boldsymbol{0}, \tag{4.44}$$

which holds assuming $\boldsymbol{M}_i$ has full rank for all $i = 1, \ldots, k$. Hence, if all tasks are compatible and kinematic singularities are avoided, the task dynamics of every task is GES. However, independent sets of task parameters do not necessarily represent a set of generalized coordinates for the robotic system. Therefore, the dynamic behavior of the entire robotic system may not be fully represented by the dynamic models in task coordinates, which occurs whenever the combination of all tasks does not consume every DOF in the system. This may lead to the problem of internal instability as discussed in [36], however, a final task at the lowest priority level regulating the joint velocities to zero can be defined. The addition of this task to the hierarchy ensures that the entire system is stable since all DOFs of the robotic system are accounted for in the control law.

4.5.1.2 Omitting compensation terms

The task acceleration reference is now given by

$$\boldsymbol{a}_i = \ddot{\boldsymbol{\sigma}}_{d,i} + \boldsymbol{K}_{d,i}\dot{\tilde{\boldsymbol{\sigma}}}_n + \boldsymbol{K}_{p,i}\tilde{\boldsymbol{\sigma}}_i - \dot{\boldsymbol{J}}_i\dot{\boldsymbol{q}}, \tag{4.45}$$

inserted into (4.42) yields the task dynamics

$$\ddot{\tilde{\boldsymbol{\sigma}}}_k + \boldsymbol{K}_{d,k}\dot{\tilde{\boldsymbol{\sigma}}}_k + \boldsymbol{K}_{p,k}\tilde{\boldsymbol{\sigma}}_k = \boldsymbol{J}_k\boldsymbol{M}^{-1}\left(\boldsymbol{J}_1^{\mathrm{T}}\boldsymbol{M}_1\boldsymbol{a}_1 + \ldots + \boldsymbol{N}_{k-1}\boldsymbol{J}_{k-1}^{\mathrm{T}}\boldsymbol{M}_{k-1}\boldsymbol{a}_{k-1}\right), \tag{4.46}$$

which contains the interference terms from higher priority tasks on the right side of the equation.

4.5.1.3 Controllability of lower priority objectives

If $\boldsymbol{M}_i^{-1}$ defined by (4.37) suffers a loss of rank, the task at priority level i is only partially controllable. The inertia matrix associated with objective i has the following singular value decomposition [37]

$$\boldsymbol{M}_i^{-1} = \boldsymbol{J}_i\boldsymbol{M}^{-1}\boldsymbol{N}_{P(i)}\boldsymbol{J}^{\mathrm{T}} = \begin{bmatrix}\boldsymbol{U}_1 & \boldsymbol{U}_2\end{bmatrix}\begin{bmatrix}\boldsymbol{\Sigma} & \\ & \boldsymbol{0}\end{bmatrix}\begin{bmatrix}\boldsymbol{V}_1^{\mathrm{T}} \\ \boldsymbol{V}_2^{\mathrm{T}}\end{bmatrix}, \tag{4.47}$$

where $\boldsymbol{\Sigma}$ corresponds to the non-zero singular values of $\boldsymbol{M}_i^{-1}$ and the columns of $\boldsymbol{U}_1$ and $\boldsymbol{V}_1$ contain the left and right singular vectors of $\boldsymbol{M}_i^{-1}$ corresponding to non-zero singular values. Therefore, the inverse, i.e. $\boldsymbol{M}_i$ may be computed from

$$\boldsymbol{M}_i = \boldsymbol{V}_1\boldsymbol{\Sigma}^{-1}\boldsymbol{U}_1^{\mathrm{T}}. \tag{4.48}$$

4.5.2 Omitting the null space operator within the task specific inertia

Define the inertia matrix associated with task i by

$$\boldsymbol{M}_i = \left(\boldsymbol{J}_i\boldsymbol{M}^{-1}\boldsymbol{J}_i^{\mathrm{T}}\right)^{-1}. \tag{4.49}$$

Furthermore, let the control input be given by (4.34) where

$$\boldsymbol{\tau}_i = \boldsymbol{J}_i^{\mathrm{T}}\boldsymbol{M}_i\boldsymbol{a}_i, \tag{4.50}$$

the task space dynamics of task k is then given by

$$\ddot{\boldsymbol{\sigma}}_k - \dot{\boldsymbol{J}}_k = \boldsymbol{J}_k \boldsymbol{M}^{-1} \boldsymbol{J}_1^{\mathrm{T}} \boldsymbol{M}_1 \boldsymbol{a}_1 + \ldots + \boldsymbol{J}_k \boldsymbol{M}^{-1} \boldsymbol{N}_k \boldsymbol{J}_k^{\mathrm{T}} \boldsymbol{M}_k \boldsymbol{a}_k. \tag{4.51}$$

Since $\boldsymbol{J}_k \boldsymbol{M}^{-1} \boldsymbol{N}_k \boldsymbol{J}_k^{\mathrm{T}} \boldsymbol{M}_k \neq \boldsymbol{I}$ for $k \neq 1$ when the null space operator of the $k-1$ previous tasks is not included within the task specific inertia matrix of task k, it is not possible to obtain sufficiently linearized dynamics allowing for compensation of interfering accelerations from higher priority tasks. Hence, of the two task acceleration references considered in Section 4.5.1 only (4.45) is feasible. The task dynamics of the highest priority task is then given by

$$\ddot{\tilde{\boldsymbol{\sigma}}}_1 + \boldsymbol{K}_{d,1} \dot{\tilde{\boldsymbol{\sigma}}}_1 + \boldsymbol{K}_{p,1} \tilde{\boldsymbol{\sigma}}_1 = \boldsymbol{0}, \tag{4.52}$$

since $\boldsymbol{N}_1 = \boldsymbol{I}$ and $\boldsymbol{J}_1 \boldsymbol{M}^{-1} \boldsymbol{J}_1^{\mathrm{T}} \boldsymbol{M}_1 = \boldsymbol{I}$.

4.6 Set-Based Operational Space Control

The control laws presented in Section 4.5 are essentially operational space inverse dynamics control laws, where torques are projected through the null space of the Jacobians of higher priority tasks. Task velocities can be obtained by computing the joint accelerations according to

$$\ddot{\boldsymbol{q}} = \boldsymbol{M}^{-1} \left(\boldsymbol{\tau} - \boldsymbol{C} \dot{\boldsymbol{q}} - \boldsymbol{g} \right), \tag{4.53}$$

followed by integration of $\ddot{\boldsymbol{q}}$ and pre-multiplication by the task Jacobian in question as shown in Section 3.2.1. Algorithm 2 can then be employed in order to determine when to activate and deactivate set-based tasks.

It is apparent that the set-based operational space approach relies on accurate knowledge of the dynamic parameters in order to predict joint accelerations that are subsequently integrated in order to predict task velocities. This is different to set-based acceleration level kinematic control, where acceleration references are generated without any consideration of the dynamics and assumed to be

tracked perfectly. If there are uncertainties in the dynamic controller whose goal is to satisfy $\ddot{\boldsymbol{q}}_d = \ddot{\boldsymbol{q}}$, there are no ways of incorporating this information into the kinematic controller. On the other hand, the dynamic model is used to predict task velocities for the operational space approach, which permits estimates of dynamic parameters to be used. Hence, set-based operational space control appears more robust to modeling inaccuracies resulting in imperfect control.

Chapter 5

Modeling of AIAUVs

This chapter describes the dynamic and kinematic modeling of AIAUVs presented in [38]. The notation has so far been consistent with that of a general robotic system since no particular robotic application has been considered. However, the rest of this thesis concerns AIAUV control and some notation will therefore be slightly different.

5.1 Reference Frames for Navigation

Since AIAUVs in work mode operate in local areas, approximately constant longitude and latitude can be assumed. Moreover, the Earth's rotation is neglected and a local Earth-fixed North-East-Down tangent frame denoted $\{n\}$ is used for navigation. This frame is assumed to be inertial such that Newton's laws still apply. The origin of the NED frame is fixed, the x-axis points North, the y-axis points East while the z-axis points downwards normal to the Earth's surface. Note that in the navigation literature this frame is often referred to as NED assumed to be inertial or a non-rotating tangent frame [39].

The body-fixed reference frame $\{b\}$ is rigidly attached to the AIAUV and located at the base, the x-axis points forward, z-axis points upwards and the

y-axis points sideways to complete the right-handed coordinate system. The reference frame of the end-effector $\{e\}$ is rigidly attached to the front of the AIAUV, where the x-axis points forward, z-axis points upwards and the y-axis points sideways to complete the right-handed coordinate system.

5.2 Kinematic Modeling of AIAUVs

For an AIAUV the system configuration is defined as $\boldsymbol{\xi} = [\boldsymbol{\eta}^{\mathrm{T}}, \boldsymbol{\theta}^{\mathrm{T}}]^{\mathrm{T}} \in \mathbb{R}^{6+n}$, where $\boldsymbol{\theta} \in \mathbb{R}^n$ represents the joint angles and

$$\boldsymbol{\eta} = \begin{bmatrix} \boldsymbol{p}_{nb}^{n} \\ \boldsymbol{\Theta}_{nb} \end{bmatrix} \in \mathbb{R}^6, \tag{5.1}$$

with $\boldsymbol{p}_{nb}^{n} = [N, E, D]^{\mathrm{T}} \in \mathbb{R}^3$ representing the position of the base frame in the aforementioned NED frame, while the orientation between $\{n\}$ and $\{b\}$ is represented by the Euler angles $\boldsymbol{\Theta}_{nb}^{b} = [\phi_b, \theta_b, \psi_b]^{\mathrm{T}} \in \mathbb{R}^3$. The components of $\boldsymbol{\Theta}_{nb}$ are denoted roll, pitch and yaw, respectively. Alternatively, the orientation can be represented by unit quaternions $\boldsymbol{q} = [\eta, \boldsymbol{\epsilon}^{\mathrm{T}}]^{\mathrm{T}} \in \mathbb{R}^4$, $\|\boldsymbol{q}\| = 1$, where $\eta \in \mathbb{R}$ is the real part of the quaternion, while $\boldsymbol{\epsilon} \in \mathbb{R}^3$ corresponds to the vector part. By using a unit quaternion representation, the system configuration is defined as $\boldsymbol{\xi} = \left[\boldsymbol{\eta}_q^{\mathrm{T}}, \boldsymbol{\theta}^{\mathrm{T}}\right]^{\mathrm{T}} \in \mathbb{R}^{7+n}$ where

$$\boldsymbol{\eta}_q = \begin{bmatrix} \boldsymbol{p}_{nb}^{n} \\ \eta \\ \boldsymbol{\epsilon} \end{bmatrix} \in \mathbb{R}^7. \tag{5.2}$$

5.2.1 Differential kinematics

The body-fixed velocity of the AIAUV is given by

$$\boldsymbol{\zeta} = \begin{bmatrix} \boldsymbol{V}_{nb}^{b} \\ \dot{\boldsymbol{\theta}} \end{bmatrix} \in \mathbb{R}^{6+n}, \quad \boldsymbol{V}_{nb}^{b} = \begin{bmatrix} \boldsymbol{v}_{nb}^{b} \\ \boldsymbol{\omega}_{nb}^{b} \end{bmatrix} \in \mathbb{R}^{6}, \tag{5.3}$$

where $\boldsymbol{v}_{nb}^{b} = [u, v, w]^{\mathrm{T}} \in \mathbb{R}^3$ and $\boldsymbol{\omega}_{nb}^{b} = [p, q, r]^{\mathrm{T}} \in \mathbb{R}^3$ are the linear and angular velocities of the body-fixed frame, respectively. $\dot{\boldsymbol{\theta}} \in \mathbb{R}^n$ represents the joint velocities.

The relation between the body-fixed velocities and the NED-frame velocities can be expressed by

$$\dot{\boldsymbol{\xi}} = \boldsymbol{J}_{\xi}(\boldsymbol{\Theta}_{nb})\boldsymbol{\zeta}, \tag{5.4}$$

where

$$\boldsymbol{J}_{\boldsymbol{\xi}}(\boldsymbol{\Theta}_{nb}) = \begin{bmatrix} \boldsymbol{R}_{nb}(\boldsymbol{\Theta}_{nb}) & \boldsymbol{0}_{3\times 3} & \boldsymbol{0}_{3\times n} \\ \boldsymbol{0}_{3\times 3} & \boldsymbol{T}_{nb}(\boldsymbol{\Theta}_{nb}) & \boldsymbol{0}_{3\times n} \\ \boldsymbol{0}_{n\times 3} & \boldsymbol{0}_{n\times 3} & \boldsymbol{I}_{n} \end{bmatrix}. \tag{5.5}$$

The rotation matrix $\boldsymbol{R}_{nb}(\boldsymbol{\Theta}_{nb}) = \boldsymbol{R}_z(\psi_b)\boldsymbol{R}_y(\theta_b)\boldsymbol{R}_x(\phi_b)$ transforms the linear velocity from the body-frame to the NED-frame and is computed according to the *zyx* convention, which is common practice in guidance, navigation and control applications [40]. The angular velocity transformation matrix mapping the body-fixed angular velocities to the Euler angle derivatives is given by

$$\boldsymbol{T}(\boldsymbol{\Theta}) = \begin{bmatrix} 1 & \sin\phi\tan\theta & \cos\phi\tan\theta \\ 0 & \cos\phi & -\sin\phi \\ 0 & \frac{\sin\phi}{\cos\theta} & \frac{\cos\phi}{\cos\theta} \end{bmatrix}. \tag{5.6}$$

5.2.2 Quaternions and Euler angles

The Euler angle attitude parametrization is intuitive and only requires three parameters. However, it suffers from a singularity corresponding to $\theta = \pm 90^\circ$ as observed from (5.6). This singularity is not a problem for surface vehicles, but for an AIAUV it can become problematic if the AIAUV is working close to the singularities. This is why a four-parameter unit quaternion representation is used in all of the simulations conducted as part of this thesis.

By employing unit quaternions, the relation between the body-fixed velocities and the NED-frame velocities can be expressed by

$$\dot{\boldsymbol{\xi}} = \boldsymbol{J}_{\boldsymbol{\xi}}(\boldsymbol{q})\boldsymbol{\zeta}, \tag{5.7}$$

where

$$\boldsymbol{J}_{\boldsymbol{\xi}}(\boldsymbol{q}) = \begin{bmatrix} \boldsymbol{R}_{nb}(\boldsymbol{q}) & \boldsymbol{0}_{3\times 3} & \boldsymbol{0}_{3\times n} \\ \boldsymbol{0}_{3\times 3} & \boldsymbol{T}_{\boldsymbol{q}}(\boldsymbol{q}) & \boldsymbol{0}_{3\times n} \\ \boldsymbol{0}_{n\times 3} & \boldsymbol{0}_{n\times 3} & \boldsymbol{I}_n \end{bmatrix}. \tag{5.8}$$

The rotation matrix is given by

$$\boldsymbol{R}_{nb}(\boldsymbol{q}) = \boldsymbol{I}_{3\times 3} + 2\eta\boldsymbol{\epsilon}_\times + 2\left(\boldsymbol{\epsilon}_\times\right)^2, \tag{5.9}$$

where $\boldsymbol{\epsilon}_\times \in \mathrm{so}(3)$ represents the skew-symmetric form of $\boldsymbol{\epsilon}$. The transformation matrix mapping the angular velocity decomposed in the base-frame to the quaternion time derivative is given by

$$\boldsymbol{T}_{\boldsymbol{q}}(\boldsymbol{q}) = \frac{1}{2}\begin{bmatrix} -\epsilon_1 & -\epsilon_2 & -\epsilon_3 \\ \eta & -\epsilon_3 & \epsilon_2 \\ \epsilon_3 & \eta & -\epsilon_1 \\ -\epsilon_2 & \epsilon_1 & \eta \end{bmatrix}. \tag{5.10}$$

Furthermore, define the transformation matrix from angular velocity decomposed in the base-frame to the vector part of the quaternion time derivative as

$$\boldsymbol{T}_{\boldsymbol{\epsilon}}(\boldsymbol{q}) = \frac{1}{2}\begin{bmatrix} \eta & -\epsilon_3 & \epsilon_2 \\ \epsilon_3 & \eta & -\epsilon_1 \\ -\epsilon_2 & \epsilon_1 & \eta \end{bmatrix}. \tag{5.11}$$

5.2.3 Forward kinematics

An AIAUV consists of $n+1$ links interconnected by n joints, the links are labeled $1, \ldots, (n+1)$, where link 1 is the tail, or base link and link $n+1$ is the head. All joints are single DOF joints, and the AIAUV is assumed to only consist of revolute joints. Joints with multiple DOFs are therefore modeled as two consecutive joints with an additional link between them.

For each link a homogenous transformation matrix is defined

$$\boldsymbol{H}_i = \begin{bmatrix} \boldsymbol{R}_{ni} & \boldsymbol{p}_{ni} \\ \boldsymbol{0}_{1\times 3} & 1 \end{bmatrix} \in \mathrm{SE}(3), \tag{5.12}$$

for $i = 1, \ldots, n+1$ which uniquely specifies the pose of link i in the NED frame defined in Section 5.1 in this thesis. Furthermore, let $\boldsymbol{A}_i(\theta_i) \in \mathrm{SE}(3), i = 1, \ldots n$ represent the mapping from the coordinate frame defined by $\boldsymbol{H}_i$ to the coordinate frame defined by $\boldsymbol{H}_{i+1}$, where θ_i is the joint variable of joint i. Given the transformation matrix $\boldsymbol{H}$, describing the position $\boldsymbol{p}_{nb}^n$ and orientation $\boldsymbol{R}_{nb}$ of the base frame in the NED frame, the position and orientation of link $i+1$ is then given by the recursive equations

$$\boldsymbol{H}_1 = \boldsymbol{H} \tag{5.13}$$

$$\boldsymbol{H}_{i+1} = \boldsymbol{H}_i \boldsymbol{A}_i(\theta_i) \tag{5.14}$$

$$= \boldsymbol{H}\boldsymbol{A}_1(\theta_1)\boldsymbol{A}_2(\theta_2)\cdots\boldsymbol{A}_i(\theta_i). \tag{5.15}$$

The position and orientation of link $n+1$, or the head frame can then be written

$$\boldsymbol{H}_{n+1} = \boldsymbol{H}\boldsymbol{A}_{1,n}(\boldsymbol{\theta}), \tag{5.16}$$

where $\boldsymbol{\theta} = [\theta_1, \theta_2, \ldots, \theta_n]^{\mathrm{T}} \in \mathbb{R}^n$ is the vector of joint parameters and

$$\boldsymbol{A}_{i,j}(\boldsymbol{\theta}) = \begin{cases} \boldsymbol{A}_i(\theta_i)\boldsymbol{A}_{i+1}(\theta_{i+1})\cdots\boldsymbol{A}_j(\theta_j), & \text{if } i \leq j \\ \boldsymbol{0}, & \text{if } i > j \end{cases}. \tag{5.17}$$

The instantaneous velocity of a rigid body in terms of its linear and angular components will from now on be described by twists, which are infinitesimal versions of a screw motion. Background material can be found in [41]. Let $\boldsymbol{a}_i = \left[\boldsymbol{\beta}_i^{\mathrm{T}}, \boldsymbol{\lambda}_i^{\mathrm{T}}\right]^{\mathrm{T}} \in \mathbb{R}^6$ represent the twist coordinates of joint i, the corresponding twist is given by

$$\boldsymbol{a}^{\wedge} = \begin{bmatrix} \boldsymbol{\beta} \\ \boldsymbol{\lambda} \end{bmatrix}^{\wedge} = \begin{bmatrix} \boldsymbol{\lambda}_{\times} & \boldsymbol{\beta} \\ 0 & 0 \end{bmatrix} \in \mathbb{R}^{4\times 4}, \tag{5.18}$$

where the wedge operator represents the mapping $\wedge : \mathbb{R}^6 \to \mathrm{se}(3)$ with $\mathrm{se}(3)$ defined as

$$\mathrm{se}(3) = \left\{ (\boldsymbol{\beta}, \boldsymbol{\lambda}_{\times}) : \boldsymbol{\beta} \in \mathbb{R}^3, \boldsymbol{\lambda}_{\times} \in \mathrm{so}(3) \right\}. \tag{5.19}$$

The 4×4 matrix $\boldsymbol{a}^{\wedge}$ in (5.18) can be interpreted as a generalization of the skew-symmetric matrix $\boldsymbol{\omega}_{\times} \in \mathrm{so}(3)$. The rigid motion associated with rotating and translating along the axis of the twist can be represented by [41]

$$\boldsymbol{A}_i(\theta_i) = \boldsymbol{A}_i(0)e^{\boldsymbol{a}_i^{\wedge}\theta_i}, \tag{5.20}$$

where the exponential map $e^{\boldsymbol{a}^{\wedge}\theta} : \mathrm{se}(3) \to \mathrm{SE}(3)$ is given by

$$e^{\boldsymbol{a}^{\wedge}\theta_i} = \begin{bmatrix} e^{\boldsymbol{\lambda}_{\times}\theta} & \left(\boldsymbol{I} - e^{\boldsymbol{\lambda}_{\times}\theta}\right)(\boldsymbol{\lambda}_{\times}\beta) + \boldsymbol{\lambda}\boldsymbol{\lambda}^{\mathrm{T}}\beta\theta \\ \boldsymbol{0}_{1\times 3} & 1 \end{bmatrix}, \quad \boldsymbol{\omega} \neq \boldsymbol{0}, \tag{5.21}$$

$$e^{\boldsymbol{\lambda}_{\times}\theta} = \boldsymbol{I} + \boldsymbol{\lambda}_{\times}\sin\theta + (\boldsymbol{\lambda}_{\times})^2(1-\cos\theta), \quad \|\boldsymbol{\lambda}\| = 1, \tag{5.22}$$

where $e^{\boldsymbol{\lambda}_{\times}\theta} : so(3) \to SO(3)$. Assuming that the AIAUV only consists of revolute joints, the twist of each joint is given by

$$\boldsymbol{a}_i = \begin{bmatrix} 0 \\ 0 \\ 0 \\ \boldsymbol{\lambda}_i \end{bmatrix}, \tag{5.23}$$

where $\boldsymbol{\lambda_i} \in \mathbb{R}^3$ is a unit vector defining the axis of rotation of joint i. Moreover, under the additional assumption that the coordinate frame of link $i+1$ is attached to joint i with its x-axis parallel to the link direction, (5.20) can be written

$$\boldsymbol{A}_i(\theta_i) = \boldsymbol{A}_i(0)\begin{bmatrix} e^{(\boldsymbol{\lambda}_i)_{\times}\theta_i} & \boldsymbol{0} \\ \boldsymbol{0}_{1\times 3} & 1 \end{bmatrix}, \tag{5.24}$$

with

$$\boldsymbol{A}_i(0) = \begin{bmatrix} \boldsymbol{I}_3 & l_i\boldsymbol{e}_1 \\ \boldsymbol{0}_{1\times 3} & 1 \end{bmatrix}, \tag{5.25}$$

where l_i is the length of link i and $\boldsymbol{e}_1 = [1,0,0]^{\mathrm{T}}$.

The head frame defined by $\boldsymbol{H}_{n+1}$ has its origin at head joint, which is at the back of the head link. The transformation $\boldsymbol{A}_e$ from the head frame to the

end-effector frame is given by a pure translation in the x-direction

$$\boldsymbol{A}_e = \begin{bmatrix} \boldsymbol{I}_3 & l_{n+1}\boldsymbol{e}_1 \\ \boldsymbol{0}_{1\times 3} & 1 \end{bmatrix}, \tag{5.26}$$

such that the position $\boldsymbol{p}_e^n$ and orientation $\boldsymbol{R}_{ne}$ of the end-effector frame relative to the NED frame is found by the transformation

$$\boldsymbol{H}_e = \begin{bmatrix} \boldsymbol{R}_{ne} & \boldsymbol{p}_e^n \\ \boldsymbol{0}_{1\times 3} & 1 \end{bmatrix} = \boldsymbol{H}_{n+1}\boldsymbol{A}_e. \tag{5.27}$$

5.2.4 Jacobians

The body twist of the base is defined as

$$\left(\boldsymbol{V}_{nb}^b\right)^\wedge = \boldsymbol{H}^{-1}\dot{\boldsymbol{H}}, \tag{5.28}$$

where $\boldsymbol{H}$ is the homogenous transformation matrix from the base frame to the NED frame. Hence, the body twist of link i is given by

$$\left(\boldsymbol{V}_{ni}^i\right)^\wedge = \boldsymbol{H}_i^{-1}\dot{\boldsymbol{H}}_i, \tag{5.29}$$

where $\boldsymbol{H}_i$ is given by (5.15), (5.17) and (5.20). Since the joint twist given by (5.20) is constant, taking the time derivative of (5.20) yields

$$\dot{\boldsymbol{A}}_i(\theta_i) = \boldsymbol{A}_i(0)e^{\boldsymbol{a}_i^\wedge\theta_i}\boldsymbol{a}_i^\wedge\dot{\theta}_i, \tag{5.30}$$

which yields

$$\begin{aligned} \boldsymbol{A}_i^{-1}\dot{\boldsymbol{A}}_i &= e^{-\boldsymbol{a}_i^\wedge\theta_i}\boldsymbol{A}_i^{-1}(0)\boldsymbol{A}_i(0)e^{\boldsymbol{a}_i^\wedge\theta_i}\boldsymbol{a}_i^\wedge\theta_i & (5.31)\\ &= \boldsymbol{a}_i^\wedge\theta_i. & (5.32) \end{aligned}$$

The body twist of link i is then given by

$$\begin{aligned}\left(\boldsymbol{V}_{ni}^{i}\right)^{\wedge} &= \boldsymbol{H}_{i}^{-1}\dot{\boldsymbol{H}}_{i} \\ &= \boldsymbol{A}_{1,i-1}^{-1}\boldsymbol{H}^{-1}\dot{\boldsymbol{H}}\boldsymbol{A}_{1,i-1} \\ &+ \boldsymbol{A}_{2,i-1}^{-1}\boldsymbol{a}_{1}^{\wedge}\boldsymbol{A}_{2,i-1}\dot{\theta}_{i} + \boldsymbol{A}_{3,i-1}^{-1}\boldsymbol{a}_{2}^{\wedge}\boldsymbol{A}_{3,i-1}\dot{\theta}_{2} + \dots \\ &+ \boldsymbol{A}_{i-1,i-1}^{-1}\boldsymbol{a}_{i-2}^{\wedge}\boldsymbol{A}_{i-1,i-1}\dot{\theta}_{i-2} + \boldsymbol{a}_{i-1}^{\wedge}\dot{\theta}_{i-1}.\end{aligned} \tag{5.33}$$

In order to proceed the adjoint operator and its inverse has to be defined. The adjoint operator, $\mathrm{Ad}(\boldsymbol{A}_i) : \mathbb{R}^6 \to \mathbb{R}^6$ maps a velocity twist in frame $i+1$ to frame i, representing the adjoint transformation associated with the homogenous transformation matrix $\boldsymbol{A}_i$ and is defined by

$$\left(\mathrm{Ad}(\boldsymbol{H})\boldsymbol{V}\right)^{\wedge} = \boldsymbol{H}\boldsymbol{V}^{\wedge}\boldsymbol{H}^{-1}, \tag{5.34}$$

and its inverse, $\mathrm{Ad}^{-1}(\boldsymbol{H}) : \mathbb{R}^6 \to \mathbb{R}^6$ is defined by

$$\left(\mathrm{Ad}^{-1}(\boldsymbol{H})\boldsymbol{V}\right)^{\wedge} = \boldsymbol{H}^{-1}\boldsymbol{V}^{\wedge}\boldsymbol{H}. \tag{5.35}$$

Both operators have matrix representations given by

$$\mathrm{Ad}(\boldsymbol{H}) = \begin{bmatrix} \boldsymbol{R} & \boldsymbol{p}_{\times}\boldsymbol{R} \\ \boldsymbol{0}_{1\times 3} & \boldsymbol{R} \end{bmatrix} \in \mathbb{R}^{6\times 6}, \tag{5.36}$$

$$\mathrm{Ad}^{-1}(\boldsymbol{H}) = \begin{bmatrix} \boldsymbol{R}^{\mathrm{T}} & -\boldsymbol{R}^{\mathrm{T}}\boldsymbol{p}_{\times} \\ \boldsymbol{0}_{1\times 3} & \boldsymbol{R}^{\mathrm{T}} \end{bmatrix} \in \mathbb{R}^{6\times 6}. \tag{5.37}$$

Therefore, (5.33) can be rewritten in body velocity twist coordinates as

$$\boldsymbol{V}_{ni}^{i} = \mathrm{Ad}^{-1}(\boldsymbol{A}_{1,i-1})\boldsymbol{V}_{nb}^{b} + \boldsymbol{a}_{i-1}\dot{\theta}_{i-1} + \sum_{j=1}^{i-2}\mathrm{Ad}^{-1}(\boldsymbol{A}_{j+1,i})\boldsymbol{a}_{j}\dot{\theta}_{j}. \tag{5.38}$$

The body velocity twist coordinates of the base frame and the joint angle velocities are collected in the vector $\boldsymbol{\zeta} = \left[\boldsymbol{V}_{nb}^{b}, \dot{\boldsymbol{\theta}}\right]^{\mathrm{T}} \in \mathbb{R}^{6+n}$. Define the Jacobian matrices $\boldsymbol{J}_i \in \mathbb{R}^{6\times(6+n)}$ by

$$\boldsymbol{V}_{ni}^{i} = \boldsymbol{J}_i \boldsymbol{\zeta}, \tag{5.39}$$

which maps the velocity twist coordinates $\boldsymbol{V}_{nb}^{b}$ and joint velocities $\dot{\boldsymbol{\theta}}$ to link velocity twist coordinates $\boldsymbol{V}_{ni}^{i}$ decomposed in their own frame. From inspection of (5.38) the Jacobians are given by

$$\boldsymbol{J}_1 = \begin{bmatrix} \boldsymbol{I}_6 & \boldsymbol{0}_{6\times n} \end{bmatrix}, \tag{5.40}$$

$$\boldsymbol{J}_{i+1} = \begin{bmatrix} \mathrm{Ad}^{-1}(\boldsymbol{A}_{1,i}) & \mathrm{Ad}^{-1}(\boldsymbol{A}_{2,i})\boldsymbol{a}_1 & \dots & \boldsymbol{a}_i & \boldsymbol{0}_{6\times(n-i)} \end{bmatrix} \tag{5.41}$$

$$= \mathrm{Ad}^{-1}(\boldsymbol{A}_i)\boldsymbol{J}_i + \begin{bmatrix} \boldsymbol{0}_{6\times(5+i)} & \boldsymbol{a}_i & \boldsymbol{0}_{6\times(n-i)} \end{bmatrix}. \tag{5.42}$$

Moreover, the time derivatives of the Jacobians are needed for acceleration and force/torque level control and is found recursively by differentiation of (5.40) and (5.42) as [38]

$$\dot{\boldsymbol{J}}_1 = \boldsymbol{0}_{6\times(6+n)}, \tag{5.43}$$

$$\dot{\boldsymbol{J}}_{i+1} = -\operatorname{ad}(\boldsymbol{a}_i)\boldsymbol{J}_{i+1}\dot{\theta}_i + \mathrm{Ad}^{-1}(\boldsymbol{A}_i)\dot{\boldsymbol{J}}_i. \tag{5.44}$$

Since the transformation matrix $\boldsymbol{A}_e$ from the head frame to the end-effector frame is constant, the body manipulator Jacobian of the end-effector is therefore given by

$$\boldsymbol{J}_e = \mathrm{Ad}^{-1}\left(\boldsymbol{A}_e\right)\boldsymbol{J}_{n+1}, \tag{5.45}$$

where $\boldsymbol{J}_{n+1}$ is the Jacobian of the head frame. The time derivative of the end-effector Jacobian is

$$\dot{\boldsymbol{J}}_e = \mathrm{Ad}^{-1}\left(\boldsymbol{A}_e\right)\dot{\boldsymbol{J}}_{n+1}. \tag{5.46}$$

5.3 Equations of Motion

The equations of motion in the base frame are given by [38]

$$\dot{\boldsymbol{\xi}} = \boldsymbol{J}_{\boldsymbol{\xi}}(\boldsymbol{q})\boldsymbol{\zeta} \tag{5.47}$$

$$\boldsymbol{M}(\boldsymbol{\theta})\dot{\boldsymbol{\zeta}} + \boldsymbol{C}(\boldsymbol{\theta},\boldsymbol{\zeta})\boldsymbol{\zeta} + \boldsymbol{D}(\boldsymbol{\theta},\boldsymbol{\zeta})\boldsymbol{\zeta} + \boldsymbol{g}(\boldsymbol{\xi}) = \boldsymbol{\tau}, \tag{5.48}$$

where the control inputs $\boldsymbol{u}$ are mapped to commanded forces and moments $\boldsymbol{\tau}$ through the actuator configuration matrix $\boldsymbol{B}(\boldsymbol{\theta})$ viz.

$$\boldsymbol{\tau} = \boldsymbol{B}(\boldsymbol{\theta})\boldsymbol{u}, \tag{5.49}$$

where $\boldsymbol{u} = \begin{bmatrix}\boldsymbol{u}_t, \boldsymbol{u}_j\end{bmatrix}^{\mathrm{T}}$ consists of the thruster inputs $\boldsymbol{u}_t \in \mathbb{R}^m$ and joint torque inputs $\boldsymbol{u}_j \in \mathbb{R}^n$. The actuator configuration matrix is given by

$$\boldsymbol{B}(\boldsymbol{\theta}) = \begin{bmatrix}\boldsymbol{J}_1(\boldsymbol{\theta})^{\mathrm{T}}\boldsymbol{B}_1 & \boldsymbol{J}_2(\boldsymbol{\theta})^{\mathrm{T}}\boldsymbol{B}_2 & \cdots & \boldsymbol{J}_n(\boldsymbol{\theta})^{\mathrm{T}}\boldsymbol{B}_n & \boldsymbol{B}_{\text{joint}}\end{bmatrix}, \tag{5.50}$$

where the link thrust configuration matrices $\boldsymbol{B}_i$ are constant and expressed as

$$\boldsymbol{B}_i = \begin{bmatrix}\boldsymbol{\beta}_{t,i,1} & \boldsymbol{\beta}_{t,i,2} & \cdots & \boldsymbol{\beta}_{t,i,m} \\ \boldsymbol{r}_{t,i,1} \times \boldsymbol{\beta}_{t,i,1} & \boldsymbol{r}_{t,i,2} \times \boldsymbol{\beta}_{t,i,2} & \cdots & \boldsymbol{r}_{t,i,m} \times \boldsymbol{\beta}_{t,i,m}\end{bmatrix}, \tag{5.51}$$

where $\boldsymbol{\beta}_{t,i,j}$ and $\boldsymbol{r}_{t,i,j}$ are the thrust direction and point of attack of the jth thruster of link i expressed in the frame of link i. The matrix $\boldsymbol{B}_{\text{joint}}$ is given by

$$\boldsymbol{B}_{\text{joint}} = \begin{bmatrix}\boldsymbol{0}_{6\times m} \\ \boldsymbol{I}_n\end{bmatrix}. \tag{5.52}$$

Note that the thruster inputs also generate torques affecting the joints, while joint torque inputs only affect the joints directly.

The inertia matrix $\boldsymbol{M}(\boldsymbol{\theta})$ is given by

$$\boldsymbol{M}(\boldsymbol{\theta}) = \sum_{i=1}^{n} \boldsymbol{J}_i^{\mathrm{T}}(\boldsymbol{\theta}) \boldsymbol{M}_i \boldsymbol{J}_i(\boldsymbol{\theta}), \tag{5.53}$$

where $\boldsymbol{M}_i$ is the inertia matrix of link i containing the rigid body mass and inertia matrix and the added mass matrix

$$\boldsymbol{M}_i = \boldsymbol{M}_{R,i} + \boldsymbol{M}_{A,i}, \tag{5.54}$$

where

$$\boldsymbol{M}_{R,i} = \begin{bmatrix} m_i \boldsymbol{I}_3 & m_i \left(\boldsymbol{r}_{g,i}\right)_{\times}^{\mathrm{T}} \\ m_i \left(\boldsymbol{r}_{g,i}\right)_{\times} & \boldsymbol{I}_{R,i} \end{bmatrix}, \tag{5.55}$$

and $\boldsymbol{I}_{R,i}$ is the rigid body inertia matrix of link i. By assuming cylindrical links with a common radius r and link lengths l_i the added mass matrix for link i is

$$\boldsymbol{M}_{A,i} = \rho \pi r^2 l_i C_a \begin{bmatrix} \alpha_i & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & \frac{1}{2} l_i \\ 0 & 0 & 1 & 0 & -\frac{1}{2} l_i & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & -\frac{1}{2} l_i & 0 & \frac{1}{3} l_i^2 & 0 \\ 0 & \frac{1}{2} l_i & 0 & 0 & 0 & \frac{1}{3} l_i^2 \end{bmatrix}, \tag{5.56}$$

where α_i is a parameter that permits added mass in surge, ρ is the density of water and C_a is the added mass coefficient.

The Coriolis and centripetal matrix $\boldsymbol{C}(\boldsymbol{\theta}, \boldsymbol{\zeta})$ is given by

$$\boldsymbol{C}(\boldsymbol{\theta}, \boldsymbol{\zeta}) = \sum_{i=1}^{n} \left(\boldsymbol{J}_i(\boldsymbol{\theta})^{\mathrm{T}} \boldsymbol{M}_i \dot{\boldsymbol{J}}_i(\boldsymbol{\theta}, \dot{\boldsymbol{\theta}}) - \boldsymbol{J}_i(\boldsymbol{\theta})^{\mathrm{T}} \boldsymbol{W}_i(\boldsymbol{\theta}, \boldsymbol{\zeta}) \boldsymbol{J}_i(\boldsymbol{\theta}) \right), \tag{5.57}$$

$$\boldsymbol{W}_i(\boldsymbol{\theta}, \boldsymbol{\zeta}) = \begin{bmatrix} \boldsymbol{0}_{3\times 3} & \left(\left\{ \boldsymbol{M}_i \boldsymbol{V}_{ni}^i \right\}_{\boldsymbol{v}} \right)_{\times} \\ \left(\left\{ \boldsymbol{M}_i \boldsymbol{V}_{ni}^i \right\}_{\boldsymbol{v}} \right)_{\times} & \left(\left\{ \boldsymbol{M}_i \boldsymbol{V}_{ni}^i \right\}_{\boldsymbol{\omega}} \right)_{\times} \end{bmatrix}, \tag{5.58}$$

where $\left\{ \boldsymbol{M}_i \boldsymbol{V}_{ni}^i \right\}_{\boldsymbol{v}} \in \mathbb{R}^3$ and $\left\{ \boldsymbol{M}_i \boldsymbol{V}_{ni}^i \right\}_{\boldsymbol{\omega}} \in \mathbb{R}^3$ are the first three and final three entries of $\boldsymbol{M}_i \boldsymbol{V}_{ni}^i$, respectively.

Hydrodynamic damping is modeled by $\boldsymbol{D}(\boldsymbol{\theta}, \boldsymbol{\zeta})$, which is given by

$$\boldsymbol{D}(\boldsymbol{\theta}, \boldsymbol{\zeta}) = \sum_{i=1}^{n} \boldsymbol{J}_i(\boldsymbol{\theta})^{\mathrm{T}} \boldsymbol{D}_i(\boldsymbol{\theta}, \boldsymbol{\zeta}) \boldsymbol{J}_i(\boldsymbol{\theta}), \tag{5.59}$$

where $\boldsymbol{D}_i(\boldsymbol{\theta}, \boldsymbol{\zeta})$ is the hydrodynamic damping matrix of link i, such that $\boldsymbol{D}_i \boldsymbol{V}_{ni}^i$ yields the hydrodynamic forces and moments on link i.

The generalized hydrostatic force $\boldsymbol{g}(\boldsymbol{\xi})$ is given as

$$\boldsymbol{g}(\boldsymbol{\xi}) = \sum_{i=1}^{n} \boldsymbol{J}_i(\boldsymbol{\theta})^{\mathrm{T}} \boldsymbol{g}_i(\boldsymbol{\xi}), \tag{5.60}$$

where $\boldsymbol{g}_i(\boldsymbol{\xi})$ are the hydrostatic forces and moments on link i

$$\boldsymbol{g}_i(\boldsymbol{\xi}) = \boldsymbol{G}_i \boldsymbol{R}_{ni}^{\mathrm{T}} \boldsymbol{\gamma}_0, \tag{5.61}$$

where $\boldsymbol{\gamma}_0$ is the constant direction of gravity in the NED frame, and $\boldsymbol{R}_{ni}$ is the rotation matrix from the NED frame to the frame of link i. The matrices $\boldsymbol{G}_i \in \mathbb{R}^{6\times 6}$ are constant and given as

$$\boldsymbol{G}_i = \begin{bmatrix} \left(\rho v_i g - m_i g \right) \boldsymbol{I}_3 \\ \rho v_i g \left(\boldsymbol{r}_{b,i} \right)_{\times} - m_i g \left(\boldsymbol{r}_{g,i} \right)_{\times} \end{bmatrix}, \tag{5.62}$$

where ρ is the density of water, v_i is the effective volume of link i, g is the gravitational acceleration constant, m_i is the mass of link i and $\boldsymbol{r}_{b,i}$ and $\boldsymbol{r}_{g,i}$ is the location of the center of buoyancy and center of gravity of link i expressed in the coordinate frame of link i, respectively.

Chapter 6

Set-Based Control of AIAUVs

This chapter begins by presenting the general control structure for an AIAUV, followed by the AIAUV model used for simulations. Then, control allocation is discussed, where the problem of rank deficiency in the actuator configuration matrix is highlighted. In Section 6.4, the set-based and equality tasks to be controlled by the task priority frameworks are defined, along with their corresponding Jacobians as well as analytic expressions for the time derivatives of the Jacobians. Finally, the task priority control frameworks described in Chapter 2-4 are presented for an AIAUV application executing the set-based and equality tasks defined in Section 6.4.

6.1 Introduction

A general framework for control of k set-based and equality tasks is presented in Figure 6.1. The controller takes as input k desired task values, velocities and accelerations as well as boundary values in case of set-based tasks. The

controller generates commanded control forces and torques $\boldsymbol{\tau}$, which serve as the input to the control allocation algorithm. The control allocation block solves a model-based optimization problem in order to distribute the generalized control forces among the actuators in terms of the control inputs $\boldsymbol{u}$. It should be noted that for velocity-level redundancy resolution in kinematic control schemes, task accelerations cannot be utilized in the controller. This is an inherent drawback and leads to lower tracking performance or the need for high task space gains [42]. The controller block implements the different control schemes discussed in the previous chapters, and therefore differs for each approach, while the control allocation block remains the same in every control scheme.

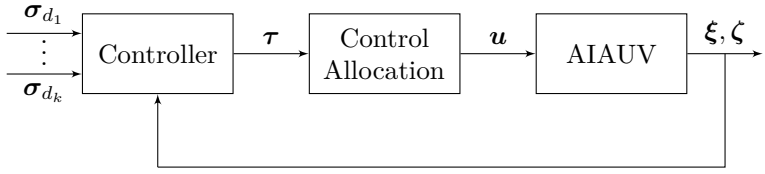


Figure 6.1: General task control framework for an AIAUV application, the desired operational space goal is fed to a controller which computes the required joint torques.

6.2 AIAUV Model

The AIAUV simulation model is identical to the one in [43]. The AIAUV consists of 5 links with 4 cardan joints connecting them, which are joints that can rotate about the y and z-axis they are attached to. The cardan joints are modeled as consecutive 1-DOF joints, thereby introducing short links separating two joints actuated about different axes, creating four new links. The total number of joints and links are therefore 8 and 9, respectively. Odd-numbered joints rotate about the z-axis, and even-numbered joints rotate about the y-axis.

The AIAUV has 7 thrusters in total, two of which are positioned on the third link, pointing in the z and y directions of the coordinate frame of link 3. Three

thrusters are positioned on the fifth link, pointing in the x, y and z directions in the coordinate frame of link 5. The last two thrusters are located at the seventh link, pointing in the y and z directions in the coordinate frame of link 7.

6.3 Control Allocation

Control allocation is the problem of distributing the commanded control forces and torques $\boldsymbol{\tau} \in \mathbb{R}^n$ computed by the controller block in Figure 6.1 to the actuators in terms of control inputs $\boldsymbol{u} \in \mathbb{R}^p$ [40]. The design of a control allocation algorithm is based on the relationship between $\boldsymbol{\tau}$ and $\boldsymbol{u}$ defined in (5.49) as

$$\boldsymbol{\tau} = \boldsymbol{B}(\boldsymbol{\theta})\boldsymbol{u}, \tag{6.1}$$

where $\boldsymbol{B}(\boldsymbol{\theta})$ is known as the actuator configuration matrix which depends on the joint configuration of the AIAUV. AIAUVs are overactuated since they are endowed with more actuators than DOFs. Specifically, for the AIAUV model used in the simulations in this thesis $\boldsymbol{\tau} \in \mathbb{R}^{14}$ and $\boldsymbol{u} \in \mathbb{R}^{15}$. Therefore, the relationship in (6.1) cannot be directly inverted in order to obtain the control inputs $\boldsymbol{u}$. Moreover, the problem now admits infinite solutions, which entails that some kind of priority should be involved such that the computation of the control inputs $\boldsymbol{u}$ can be represented as an optimization problem [44]. By neglecting any saturation or rate constraints on $\boldsymbol{u}$, the control allocation problem can be represented by the minimization problem

$$\min_{u \in \mathbb{R}^p} \frac{1}{2} \left(\boldsymbol{u} - \boldsymbol{u}_p\right)^{\mathrm{T}} \boldsymbol{W} \left(\boldsymbol{u} - \boldsymbol{u}_p\right) \tag{6.2}$$

$$\text{subject to:} \qquad \boldsymbol{\tau} = \boldsymbol{B}(\boldsymbol{\theta})\boldsymbol{u}, \tag{6.3}$$

where $\boldsymbol{W} \in \mathbb{R}^{p \times p}$ is a positive definite weight matrix, while $\boldsymbol{u}_p$ is the preferred value of $\boldsymbol{u}$. When $\boldsymbol{B}$ has full row rank, the problem has an explicit solution

given by

$$\boldsymbol{u} = \left(\boldsymbol{I} - \boldsymbol{B}_{\boldsymbol{W}}^{\dagger}\boldsymbol{B}\right)\boldsymbol{u}_p + \boldsymbol{B}_{\boldsymbol{W}}^{\dagger}\boldsymbol{\tau}_c, \tag{6.4}$$

with

$$\boldsymbol{B}_{\boldsymbol{W}}^{\dagger} = \boldsymbol{W}^{-1}\boldsymbol{B}^{\mathrm{T}}\left(\boldsymbol{B}\boldsymbol{W}^{-1}\boldsymbol{B}^{\mathrm{T}}\right)^{-1}, \tag{6.5}$$

which is a generalized weighted least squares inverse of $\boldsymbol{B}$. In the case of $\boldsymbol{u}_p = \boldsymbol{0}$ and $\boldsymbol{W} = \boldsymbol{I}$ the solution reduces to the right Moore-Penrose pseudoinverse viz.

$$\boldsymbol{u} = \boldsymbol{B}^{\dagger}\boldsymbol{\tau} = \boldsymbol{B}^{\mathrm{T}}\left(\boldsymbol{B}\boldsymbol{B}^{\mathrm{T}}\right)^{-1}\boldsymbol{\tau}. \tag{6.6}$$

The problem of rank deficiency of $\boldsymbol{B}(\boldsymbol{\theta})$ was pointed out in [1], and implies that no force or moment can be generated in certain directions in the vector space $\mathbb{R}^m$ belonging to $\boldsymbol{\tau}$. Regularization methods can be employed to overcome this problem, and a method known as the damped least-squares pseudoinverse was suggested in [1]

$$\boldsymbol{B}_{\boldsymbol{W},\lambda}^{\dagger} = \boldsymbol{W}^{-1}\boldsymbol{B}^{\mathrm{T}}\left(\boldsymbol{B}\boldsymbol{W}^{-1}\boldsymbol{B}^{\mathrm{T}} + \lambda\boldsymbol{I}\right)^{-1}, \tag{6.7}$$

where $\lambda > 0$ is a small regularization parameter ensuring that the matrix to be inverted always has full rank. However, the introduction of the regularization parameter inevitably leads to a loss of performance and an increased tracking error [45]. The tracking error can be improved by varying λ dynamically, for instance by only adding the damping parameter to the smallest singular values [9]. This approach introduces the problem of tuning the damping coefficient, while still not eliminating the problem of tracking errors entirely. Alternatively, a singular value decomposition of the matrix

$$\boldsymbol{B}\boldsymbol{W}^{-\mathbf{1}}\boldsymbol{B}^{\mathbf{T}} = \boldsymbol{U}\boldsymbol{\Sigma}\boldsymbol{V}^{\mathrm{T}}, \tag{6.8}$$

can be employed, where $\boldsymbol{\Sigma}$ is a rectangular diagonal matrix containing the singular values σ_i. The SVD characterizes the directions where no generalized force can be produced [37]. This leads to the following approximate inverse

$$\boldsymbol{B}^{\dagger}_{\boldsymbol{W},\mathrm{SVD}} = \boldsymbol{W}^{-1}\boldsymbol{B}^{\mathrm{T}}\boldsymbol{V}\boldsymbol{\Sigma}^{*}_{\delta}\boldsymbol{U}^{\mathrm{T}}, \tag{6.9}$$

where $\boldsymbol{\Sigma}^{*}_{\delta}$ is formed by replacing every singular value on the diagonal satisfying $\sigma_i \geq \delta$ by its reciprocal, setting the remaining singular values to zero and transposing the resulting matrix.

Within the aforementioned approach to unconstrained control allocation no effort has been made to satisfy any constraints on the inputs $\boldsymbol{u}$. A naïve and straightforward solution is to saturate the obtained control inputs $\boldsymbol{u}$. However, this implies that the allocated control force is different from the commanded force whenever any of the inputs in $\boldsymbol{u}$ are in saturation. Hence, the allocated force cannot be guaranteed to equal the required force even if exact allocation is feasible under the constraints. Moreover, no attempt is made to minimize the error between allocated and commanded force whenever exact allocation is not possible. Instead, more advanced constrained control allocation methods such as solving linear programs [44] could be employed in order to guarantee that the constraints on the inputs $\boldsymbol{u}$ are satisfied, and that the error between allocated and commanded force is minimized in some sense whenever exact allocation is not possible.

The drawback of singularity avoidance within a control allocation algorithm is that no effort is made to prevent the robotic system from obtaining singular configurations in the first place. As an alternative approach, singularity avoidance may be introduced as a high priority set-based task in a kinematic or operational space controller, thereby preventing singular configurations while other control objectives may be simultaneously accomplished by exploiting the redundancy of the system.

6.4 Set-Based and Equality Tasks for AIAUV Control

Several different tasks are of interest for an AIAUV operating in inspection mode. The proposed control schemes are tested with three set-based tasks and three equality tasks.

6.4.1 End-effector collision avoidance

To avoid a collision between the end-effector and some obstacle, the scalar distance measure between them is used as a set-based task

$$\sigma_{a_1} = \sqrt{\left(\boldsymbol{p}^n_{\text{obs}} - \boldsymbol{p}^n_{ne}\right)^{\mathrm{T}} \left(\boldsymbol{p}^n_{\text{obs}} - \boldsymbol{p}^n_{ne}\right)} \in \mathbb{R}, \tag{6.10}$$

$$\dot{\sigma}_{a_1} = \boldsymbol{J}_{a_1}\boldsymbol{\zeta}, \tag{6.11}$$

$$\ddot{\sigma}_{a_1} = \boldsymbol{J}_{a_1}\dot{\boldsymbol{\zeta}} + \dot{\boldsymbol{J}}_{a_1}\boldsymbol{\zeta}, \tag{6.12}$$

where the task Jacobian $\boldsymbol{J}_{a_1} \in \mathbb{R}^{1\times(6+n)}$ is given by

$$\boldsymbol{J}_{a_1} = \frac{-\left(\boldsymbol{p}^n_{\text{obs}} - \boldsymbol{p}^n_{ne}\right)^{\mathrm{T}}}{\sigma_{a_1}} \begin{bmatrix} \boldsymbol{R}_{ne}(\boldsymbol{q}) & \boldsymbol{0}_{3\times 3} \end{bmatrix} \boldsymbol{J}_e\boldsymbol{\zeta}, \tag{6.13}$$

where the end-effector Jacobian $\boldsymbol{J}_e \in \mathbb{R}^{6\times(6+n)}$ is given by the recursive equations (5.40), (5.42) and (5.45). To derive the time derivative of the task Jacobian consider the term

$$\boldsymbol{x} = \frac{-(\boldsymbol{p}_o - \boldsymbol{p})}{\sigma_{a_1}}, \tag{6.14}$$

the time derivative is given by

$$\frac{\mathrm{d}\boldsymbol{x}}{\mathrm{d}t} = \frac{\partial \boldsymbol{x}}{\partial \boldsymbol{p}}\dot{\boldsymbol{p}} \tag{6.15}$$

$$= \frac{\boldsymbol{I}\sigma_{a_1} + (\boldsymbol{p}_o - \boldsymbol{p})\frac{\partial \sigma_{a_1}}{\partial \boldsymbol{p}}}{\sigma_{a_1}^2}\boldsymbol{v} \tag{6.16}$$

$$= \left(\frac{\boldsymbol{I}}{\sigma_{a_1}} - \frac{(\boldsymbol{p}_o - \boldsymbol{p})(\boldsymbol{p}_o - \boldsymbol{p})^{\mathrm{T}}}{\sigma_{a_1}^3}\right)\boldsymbol{v}, \tag{6.17}$$

transposing both sides yields

$$\frac{\mathrm{d}\boldsymbol{x}^{\mathrm{T}}}{\mathrm{d}t} = \boldsymbol{v}^{\mathrm{T}}\left(\frac{\boldsymbol{I}}{\sigma_{a_1}} - \frac{(\boldsymbol{p}_o - \boldsymbol{p})(\boldsymbol{p}_o - \boldsymbol{p})^{\mathrm{T}}}{\sigma_{a_1}^3}\right)^{\mathrm{T}} \tag{6.18}$$

$$= \boldsymbol{v}^{\mathrm{T}}\left(\frac{\boldsymbol{I}}{\sigma_{a_1}} - \frac{(\boldsymbol{p}_o - \boldsymbol{p})(\boldsymbol{p}_o - \boldsymbol{p})^{\mathrm{T}}}{\sigma_{a_1}^3}\right). \tag{6.19}$$

Hence, the time derivative of the task Jacobian is given by

$$\dot{\boldsymbol{J}}_{a_1} = (\boldsymbol{v}_{ne}^n)^{\mathrm{T}}\left(\frac{\boldsymbol{I}}{\sigma_{a_1}} - \frac{(\boldsymbol{p}_{\mathrm{obs}}^n - \boldsymbol{p}_{ne}^n)(\boldsymbol{p}_{\mathrm{obs}}^n - \boldsymbol{p}_{ne}^n)^{\mathrm{T}}}{\sigma_{a_1}^3}\right)\begin{bmatrix}\boldsymbol{R}_{ne} & \boldsymbol{0}_{3\times 3}\end{bmatrix}\boldsymbol{J}_e \tag{6.20}$$

$$+ \frac{-(\boldsymbol{p}_{\mathrm{obs}}^n - \boldsymbol{p}_{ne}^n)^{\mathrm{T}}}{\sigma_{a_1}}\left(\begin{bmatrix}\boldsymbol{R}_{ne}(\boldsymbol{\omega}_{ne}^e)_\times & \boldsymbol{0}_{3\times 3}\end{bmatrix}\boldsymbol{J}_e + \begin{bmatrix}\boldsymbol{R}_{ne} & \boldsymbol{0}_{3\times 3}\end{bmatrix}\dot{\boldsymbol{J}}_e\right), \tag{6.21}$$

where the time derivative of the end-effector Jacobian $\dot{\boldsymbol{J}}_e(\boldsymbol{\theta}) \in \mathbb{R}^{6\times 14}$ is given by the recursive equations (5.43), (5.44) and (5.46).

6.4.2 Joint Limit Avoidance

To avoid exceeding mechanical joint limits a set-based joint limit avoidance task is defined by

$$\boldsymbol{\sigma}_{a_2} = \boldsymbol{\theta}, \tag{6.22}$$

$$\dot{\boldsymbol{\sigma}}_{a_2} = \boldsymbol{J}_{a_2}\boldsymbol{\zeta}, \tag{6.23}$$

$$\ddot{\boldsymbol{\sigma}}_{a_2} = \boldsymbol{J}_{a_2}\dot{\boldsymbol{\zeta}}, \tag{6.24}$$

where the task Jacobian is constant and given by

$$\boldsymbol{J}_{a_2} = \begin{bmatrix} \mathbf{0}_{8\times n} & \boldsymbol{I}_{n\times n} \end{bmatrix}. \tag{6.25}$$

6.4.3 Actuator singularity avoidance

As discussed in Section 6.3, it is desirable to avoid configurations where the actuator configuration matrix is singular. Inspired by the manipulability index [35], the actuation index task is defined by

$$\sigma_{a_3} = \det\left(\boldsymbol{B}(\boldsymbol{\theta})\boldsymbol{B}(\boldsymbol{\theta})^{\mathrm{T}}\right), \tag{6.26}$$

$$\dot{\sigma}_{a_3} = \boldsymbol{J}_{a_3}\boldsymbol{\zeta}, \tag{6.27}$$

$$\ddot{\sigma}_{a_3} = \boldsymbol{J}_{a_3}\dot{\boldsymbol{\zeta}} + \dot{\boldsymbol{J}}_{a_3}\boldsymbol{\zeta}, \tag{6.28}$$

where $\boldsymbol{B}(\boldsymbol{\theta})$ is the actuator configuration matrix and the task Jacobian is given by

$$\boldsymbol{J}_{a_3} = \begin{bmatrix} \mathbf{0}_{1\times 6} & \frac{\partial \sigma_a}{\partial \theta_1} & \frac{\partial \sigma_a}{\partial \theta_2} & \frac{\partial \sigma_a}{\partial \theta_3} & \cdots & \frac{\partial \sigma_a}{\partial \theta_n} \end{bmatrix}, \tag{6.29}$$

and the actuation index derivative is given by [46]

$$\frac{\partial \sigma}{\partial \theta_i} = 2\sigma \,\mathrm{Tr}\left(\frac{\partial \boldsymbol{B}}{\partial \theta_i}\boldsymbol{B}^{\dagger}\right), \tag{6.30}$$

where $\boldsymbol{B}^\dagger$ is the right Moore-Penrose pseudoinverse of $\boldsymbol{B}$. Furthermore, the task Jacobian derivative is given by [46]

$$\dot{\boldsymbol{J}}_{a_3} = \begin{bmatrix} \frac{\partial}{\partial \boldsymbol{\theta}} \left(\mathbf{0}_{1\times 6}\right) \dot{\boldsymbol{\theta}} & \frac{\partial}{\partial \boldsymbol{\theta}} \left(\frac{\partial \sigma}{\partial \theta_1}\right) \dot{\boldsymbol{\theta}} & \frac{\partial}{\partial \boldsymbol{\theta}} \left(\frac{\partial \sigma}{\partial \theta_2}\right) \dot{\boldsymbol{\theta}} & \cdots & \frac{\partial}{\partial \boldsymbol{\theta}} \left(\frac{\partial \sigma}{\partial \theta_n}\right) \dot{\boldsymbol{\theta}} \end{bmatrix} \tag{6.31a}$$

$$= \begin{bmatrix} \mathbf{0}_{6\times 1} \\ \frac{\partial^2 \sigma}{\partial \theta_1^2}\dot{\theta}_1 + \frac{\partial^2 \sigma}{\partial \theta_1 \partial \theta_2}\dot{\theta}_2 + \frac{\partial^2 \sigma}{\partial \theta_1 \partial \theta_3}\dot{\theta}_3 + \cdots + \frac{\partial^2 \sigma}{\partial \theta_1 \partial \theta_n}\dot{\theta}_n \\ \frac{\partial^2 \sigma}{\partial \theta_2 \partial \theta_1}\dot{\theta}_1 + \frac{\partial^2 \sigma}{\partial \theta_2^2}\dot{\theta}_2 + \frac{\partial^2 \sigma}{\partial \theta_2 \partial \theta_3}\dot{\theta}_3 + \cdots + \frac{\partial^2 \sigma}{\partial \theta_2 \partial \theta_n}\dot{\theta}_n \\ \frac{\partial^2 \sigma}{\partial \theta_3 \partial \theta_1}\dot{\theta}_1 + \frac{\partial^2 \sigma}{\partial \theta_3 \partial \theta_2}\dot{\theta}_2 + \frac{\partial^2 \sigma}{\partial \theta_3^2}\dot{\theta}_3 + \cdots + \frac{\partial^2 \sigma}{\partial \theta_3 \partial \theta_n}\dot{\theta}_n \\ \vdots \\ \frac{\partial^2 \sigma}{\partial \theta_n \partial \theta_1}\dot{\theta}_1 + \frac{\partial^2 \sigma}{\partial \theta_n \partial \theta_2}\dot{\theta}_2 + \frac{\partial^2 \sigma}{\partial \theta_n \partial \theta_3}\dot{\theta}_3 + \cdots + \frac{\partial^2 \sigma}{\partial \theta_n^2}\dot{\theta}_n \end{bmatrix}^T \tag{6.31b}$$

$$= \begin{bmatrix} \mathbf{0}_{6\times 1} \\ \dot{\theta}_1 \\ \dot{\theta}_2 \\ \dot{\theta}_3 \\ \vdots \\ \dot{\theta}_n \end{bmatrix}^T \begin{bmatrix} \mathbf{0}_{6\times n} \\ \\ \frac{\partial^2 \sigma}{\partial \theta_1^2} & \frac{\partial^2 \sigma}{\partial \theta_2 \partial \theta_1} & \frac{\partial^2 \sigma}{\partial \theta_3 \partial \theta_1} & \cdots & \frac{\partial^2 \sigma}{\partial \theta_n \partial \theta_1} \\ \frac{\partial^2 \sigma}{\partial \theta_1 \partial \theta_2} & \frac{\partial^2 \sigma}{\partial \theta_2^2} & \frac{\partial^2 \sigma}{\partial \theta_3 \partial \theta_2} & \cdots & \frac{\partial^2 \sigma}{\partial \theta_n \partial \theta_2} \\ \frac{\partial^2 \sigma}{\partial \theta_1 \partial \theta_3} & \frac{\partial^2 \sigma}{\partial \theta_2 \partial \theta_3} & \frac{\partial^2 \sigma}{\partial \theta_3^2} & \cdots & \frac{\partial^2 \sigma}{\partial \theta_n \partial \theta_3} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ \frac{\partial^2 \sigma}{\partial \theta_1 \partial \theta_n} & \frac{\partial^2 \sigma}{\partial \theta_2 \partial \theta_n} & \frac{\partial^2 \sigma}{\partial \theta_3 \partial \theta_n} & \cdots & \frac{\partial^2 \sigma}{\partial \theta_n^2} \end{bmatrix}, \tag{6.31c}$$

where the cross partials are given by

$$\frac{\partial^2 \sigma}{\partial \theta_i \partial \theta_j} = 2\frac{\partial \sigma}{\partial \theta_j} \operatorname{Tr}\left(\frac{\partial \boldsymbol{B}}{\partial \theta_i}\boldsymbol{B}^+\right) + 2\sigma \operatorname{Tr}\left(\frac{\partial^2 \boldsymbol{B}}{\partial \theta_j \partial \theta_i}\boldsymbol{B}^+ + \frac{\partial \boldsymbol{B}}{\partial \theta_i}\left\{\left(\frac{\partial \boldsymbol{B}}{\partial \theta_j}\right)^T - \boldsymbol{B}^+\left[\frac{\partial \boldsymbol{B}}{\partial \theta_j}\boldsymbol{B}^T + \boldsymbol{B}\left(\frac{\partial \boldsymbol{B}}{\partial \theta_j}\right)^T\right]\right\}\left(\boldsymbol{B}\boldsymbol{B}^T\right)^{-1}\right). \tag{6.32}$$

When implemented, the actuator configuration matrix $\boldsymbol{B}(\boldsymbol{\theta})$ is computed symbolically in order to find the partial and cross partial derivatives $\frac{\partial \boldsymbol{B}}{\partial \theta_i}$ and $\frac{\partial}{\partial \theta_j}\frac{\partial \boldsymbol{B}}{\partial \theta_i}$. The computations are performed offline and with respect to a specific AIAUV configuration due to the computational complexity.

6.4.4 End-effector configuration control

The pose of the end-effector relative to the NED frame is given by the forward kinematics described in Section 5.2.3. A unit quaternion representation is employed, where the end-effector quaternion $\boldsymbol{q} = [\eta, \boldsymbol{\epsilon}^{\mathrm{T}}]^{\mathrm{T}}$ can be obtained from the rotation matrix $\boldsymbol{R}_{ne}$ given by the forward kinematics. From a quaternion $\boldsymbol{q}_d = \left[\eta_d, \boldsymbol{\epsilon}_d^{\mathrm{T}}\right]^{\mathrm{T}}$ describing the desired attitude, a corresponding desired rotation matrix $\boldsymbol{R}_{nd}$ can be calculated. The rotation matrix representing the attitude error between the desired end-effector attitude and the end-effector attitude given by the forward kinematics is [19]

$$\tilde{\boldsymbol{R}} = \boldsymbol{R}_{nd}\left(\boldsymbol{R}_{ne}\right)^{\mathrm{T}}. \tag{6.33}$$

The quaternion associated with $\tilde{\boldsymbol{R}}$ can be computed from the quaternion product $\tilde{\boldsymbol{q}} = \boldsymbol{q}_d \otimes \boldsymbol{q}^*$ given by

$$\tilde{\eta} = \eta_d \eta + \boldsymbol{\epsilon}_d^{\mathrm{T}} \boldsymbol{\epsilon}, \tag{6.34}$$

$$\tilde{\boldsymbol{\epsilon}} = \eta \boldsymbol{\epsilon}_d - \eta_d \boldsymbol{\epsilon} + \boldsymbol{\epsilon}_{\times} \boldsymbol{\epsilon}_d, \tag{6.35}$$

where $\boldsymbol{q}^* = [\eta, -\boldsymbol{\epsilon}^{\mathrm{T}}]^{\mathrm{T}}$ is the conjugate of the end-effector quaternion. Now, since the rotation matrix representing two aligned frames is given by $\tilde{\boldsymbol{R}} = \boldsymbol{I}$, which corresponds to the quaternion $\tilde{\boldsymbol{q}} = [1, \boldsymbol{0}^{\mathrm{T}}]^{\mathrm{T}}$, it is sufficient to represent the attitude error as the three-dimensional imaginary part $\tilde{\boldsymbol{\epsilon}}$ of the quaternion error

vector $\tilde{\boldsymbol{q}}$. The end-effector configuration task is then defined by

$$\boldsymbol{\sigma}_1 = \begin{bmatrix} \boldsymbol{p}_{ne}^n \\ \boldsymbol{\epsilon} \end{bmatrix} \in \mathbb{R}^6, \tag{6.36}$$

$$\dot{\boldsymbol{\sigma}}_1 = \boldsymbol{J}_1 \boldsymbol{\zeta}, \tag{6.37}$$

$$\ddot{\boldsymbol{\sigma}}_1 = \boldsymbol{J}_1 \dot{\boldsymbol{\zeta}} + \dot{\boldsymbol{J}}_1 \boldsymbol{\zeta}, \tag{6.38}$$

where the Jacobian and its time derivative is given by

$$\boldsymbol{J}_1 = \begin{bmatrix} \boldsymbol{R}_{ne}(\boldsymbol{q}) & \boldsymbol{0}_{3\times 3} \\ \boldsymbol{0}_{3\times 3} & \boldsymbol{T}_{\boldsymbol{\epsilon}_{ne}}(\boldsymbol{q}) \end{bmatrix} \boldsymbol{J}_e(\boldsymbol{\theta}) \in \mathbb{R}^{6\times(6+n)} \tag{6.39}$$

$$\dot{\boldsymbol{J}}_1 = \begin{bmatrix} \boldsymbol{R}_{ne}(\boldsymbol{q}) \left(\boldsymbol{\omega}_{ne}^e\right)_\times & \boldsymbol{0}_{3\times 3} \\ \boldsymbol{0}_{3\times 3} & \dot{\boldsymbol{T}}_{\boldsymbol{\epsilon}_{ne}}(\boldsymbol{q}) \end{bmatrix} \boldsymbol{J}_e(\boldsymbol{\theta}) + \begin{bmatrix} \boldsymbol{R}_{ne}(\boldsymbol{q}) & \boldsymbol{0}_{3\times 3} \\ \boldsymbol{0}_{3\times 3} & \boldsymbol{T}_{\boldsymbol{\epsilon}_{ne}}(\boldsymbol{q}) \end{bmatrix} \dot{\boldsymbol{J}}_e(\boldsymbol{\theta}), \tag{6.40}$$

where the end-effector Jacobian $\boldsymbol{J}_e \in \mathbb{R}^{6\times(6+n)}$ is given by the recursive equations (5.40), (5.42) and (5.45), and its time derivative $\dot{\boldsymbol{J}}_e(\boldsymbol{\theta}) \in \mathbb{R}^{6\times(6+n)}$ is given by (5.43), (5.44) and (5.46). Moreover, the time derivative of the transformation matrix mapping angular velocities into the time derivative of the vector part of the quaternion is given by [47]

$$\dot{\boldsymbol{T}}_{\boldsymbol{\epsilon}}(\boldsymbol{q}) = \frac{\partial \boldsymbol{T}}{\partial \boldsymbol{q}} \left(\boldsymbol{I}_3 \otimes \dot{\boldsymbol{q}}\right), \tag{6.41}$$

where the partial derivative of the transformation matrix with respect to the quaternion vector is

$$\frac{\partial \boldsymbol{T}_{\boldsymbol{\epsilon}}}{\partial \boldsymbol{q}} = \frac{1}{2} \begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 & 0 & -1 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 & 1 & 0 & 0 & 0 & 0 & -1 & 0 & 0 \\ 0 & 0 & -1 & 0 & 0 & 1 & 0 & 0 & 1 & 0 & 0 & 0 \end{bmatrix} \in \mathbb{R}^{3\times 12}, \tag{6.42}$$

$$\tag{6.43}$$

and $\otimes$ denotes the Kronecker product which is given by

$$(\boldsymbol{I}_3 \otimes \dot{\boldsymbol{q}}) = \begin{bmatrix} \dot{\boldsymbol{q}} & \boldsymbol{0}_{4\times1} & \boldsymbol{0}_{4\times1} \\ \boldsymbol{0}_{4\times1} & \dot{\boldsymbol{q}} & \boldsymbol{0}_{4\times1} \\ \boldsymbol{0}_{4\times1} & \boldsymbol{0}_{4\times1} & \dot{\boldsymbol{q}} \end{bmatrix} \in \mathbb{R}^{12\times3}. \tag{6.44}$$

6.4.5 Base position control

For intervention or inspection tasks within the workspace of the robotic manipulator, it is desirable to keep the base in a fixed position while attempting to utilize the joints instead of the thrusters for reconfiguration of the end-effector. The position of the base in the NED frame is given by $\boldsymbol{p}_{nb}^{n} \in \mathbb{R}^3$, the base position task is defined by

$$\boldsymbol{\sigma}_2 = \boldsymbol{p}_{nb}^{n} \in \mathbb{R}^3, \tag{6.45}$$

$$\dot{\boldsymbol{\sigma}}_2 = \boldsymbol{J}_2\boldsymbol{\zeta}, \tag{6.46}$$

$$\ddot{\boldsymbol{\sigma}}_2 = \boldsymbol{J}_2\dot{\boldsymbol{\zeta}} + \dot{\boldsymbol{J}}_2\boldsymbol{\zeta}, \tag{6.47}$$

where the Jacobian $\boldsymbol{J}_2 \in \mathbb{R}^{3\times(6+n)}$ and its time derivative is given by

$$\boldsymbol{J}_2 = \begin{bmatrix} \boldsymbol{R}_{nb}(\boldsymbol{q}) & \boldsymbol{0}_{3\times(3+n)} \end{bmatrix}, \tag{6.48}$$

$$\dot{\boldsymbol{J}}_2 = \begin{bmatrix} \boldsymbol{R}_{nb}(\boldsymbol{q}) \left(\boldsymbol{\omega}_{nb}^{b}\right)_{\times} & \boldsymbol{0}_{3\times(3+n)} \end{bmatrix}. \tag{6.49}$$

$$\tag{6.50}$$

6.4.6 Null space velocity

Since the end-effector configuration and base positioning tasks only consume 9 DOFs, the problem of internal instability may arise for acceleration-based and operational space controllers applied to highly redundant AIAUVs. In order to prevent this, a task at the lowest priority is defined which tries to regulate the angular velocity of the base and the joint velocities to zero, consuming any

residual DOFs in the system. The task is defined by

$$\dot{\boldsymbol{\sigma}}_3 = \boldsymbol{J}_3 \boldsymbol{\zeta}, \tag{6.51}$$

where the Jacobian $\boldsymbol{J}_3 \in \mathbb{R}^{(3+n)\times(6+n)}$ is given by

$$\boldsymbol{J}_3 = \begin{bmatrix} \boldsymbol{0}_{(3+n)\times 3} & \boldsymbol{I}_{3+n} \end{bmatrix}. \tag{6.52}$$

Note that the base linear velocity is omitted from this task, since it is always dealt with by the base positioning task. In the robotics literature, it is common to add a term penalizing deviations from some preferred pose to the final null space task in addition to the velocity term [42].

6.4.7 Priority levels

In every implementation the highest priority task is the stacked joint limit avoidance, collision avoidance and actuation index set-based tasks[1] $\boldsymbol{\sigma}_a = \begin{bmatrix} \sigma_{a_1}, \boldsymbol{\sigma}_{a_2}^{\mathrm{T}}, \sigma_{a_3} \end{bmatrix}^{\mathrm{T}}$. The second priority level contains the 6 DOF end-effector configuration task $\boldsymbol{\sigma}_1$, while the third priority level contains the base positioning task $\boldsymbol{\sigma}_2$. Moreover, the velocity task $\boldsymbol{\sigma}_3$ is included in the acceleration-based and operational space controllers to ensure internal stability. Letters are used in task subscripts to denote set-based tasks and numbers are used to denote equality-based tasks. By using this convention, tasks are labelled such that task a has strictly higher priority than task b, task 1 has strictly higher priority than task 2 and so on. Furthermore, only high-priority set-based tasks are considered, which implies that every set-based task always has higher priority than any equality-based task.

[1] Except for the iCAT framework, where there is strict priority among set-based tasks.

6.5 Set-Based Velocity Control

When solving redundancy at the velocity level, the controller block from Figure 6.1 is decomposed into a kinematic controller and a dynamic controller as depicted in Figure 6.2. A feedback linearizing dynamic controller is employed in

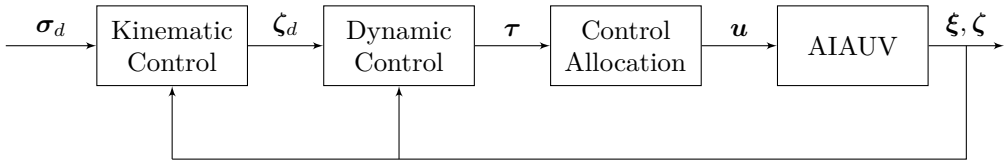


Figure 6.2: Overall control architecture for an AIAUV application when employing a kinematic control scheme. The kinematic controller transforms a goal specified through an operational space task into desired system velocities accomplishing the goal.

all velocity control schemes, the control law is given by

$$\boldsymbol{\tau} = \boldsymbol{M}(\boldsymbol{\theta})\boldsymbol{a}^b + \boldsymbol{C}(\boldsymbol{\theta}, \boldsymbol{\zeta})\boldsymbol{\zeta} + \boldsymbol{D}(\boldsymbol{\theta}, \boldsymbol{\zeta})\boldsymbol{\zeta} + \boldsymbol{g}(\boldsymbol{\xi}), \tag{6.53}$$

where the commanded acceleration vector is given by

$$\boldsymbol{a}^b = \dot{\boldsymbol{\zeta}}_d + \boldsymbol{K}_p\tilde{\boldsymbol{\zeta}} + \boldsymbol{K}_i \int_0^t \tilde{\boldsymbol{\zeta}}(\tau)\,\mathrm{d}\tau, \tag{6.54}$$

where $\tilde{\boldsymbol{\zeta}} = \boldsymbol{\zeta}_d - \boldsymbol{\zeta}$ denotes the velocity error and $\boldsymbol{\zeta}_d$ is the output from the kinematic controller. The desired system acceleration is computed by numeric differentiation viz.

$$\dot{\boldsymbol{\zeta}}_d \simeq \frac{\boldsymbol{\zeta}_d(t) - \boldsymbol{\zeta}_d(t - \Delta t)}{\Delta t}, \tag{6.55}$$

where Δt is the sampling period. By inserting this control law into (5.48) the equations of motion reduce to

$$\dot{\tilde{\boldsymbol{\zeta}}} + \boldsymbol{K}_p \tilde{\boldsymbol{\zeta}} + \boldsymbol{K}_i \int_0^t \boldsymbol{\zeta}(\tau)\, \mathrm{d}\tau = \mathbf{0}, \tag{6.56}$$

which has a globally exponentially stable equilibrium point corresponding to $\left[\tilde{\boldsymbol{\zeta}}^{\mathrm{T}}, \dot{\tilde{\boldsymbol{\zeta}}}^{\mathrm{T}}\right]^{\mathrm{T}} = \mathbf{0}$ when the gains are chosen according to

$$\boldsymbol{K}_i = \mathrm{diag}\left\{\omega_1^2, \ldots \omega_n^2\right\} \tag{6.57}$$

$$\boldsymbol{K}_p = \mathrm{diag}\left\{2\omega_1, \ldots, 2\omega_n\right\}. \tag{6.58}$$

Moreover, this choice of gains yield a globally decoupled closed loop system, where the response of each component in $\boldsymbol{\zeta}$ is equal to the response of a critically damped linear second order system with natural frequency ω_i, determining the rate of decay of the tracking error.

6.5.1 Set-based SRMTP framework

Within the set-based SRMTP framework presented in [4], the desired system velocities are discontinuous when activating and deactivating set-based tasks. Consequently, the desired accelerations computed by (6.55) can become excessively large during activation or deactivation of tasks. To prevent this, acceleration feedforward is removed from the commanded acceleration vector given by (6.54), resulting in worse tracking performance and the need for higher gains in the dynamic controller in order to satisfy $\boldsymbol{\zeta}_d = \boldsymbol{\zeta}$.

The highest priority task is a stacked set-based task $\boldsymbol{\sigma}_a$ consisting of the joint limit avoidance task, the collision avoidance task and the actuation index tasks all described in Section 6.4. The second priority level consists of the end-effector configuration task $\boldsymbol{\sigma}_1$, while the base positioning task $\boldsymbol{\sigma}_2$ is on the third and lowest priority level. When no set-based tasks are active, the desired system

velocities are given by

$$\boldsymbol{\zeta}_d = \boldsymbol{J}_1^\dagger \dot{\boldsymbol{\sigma}}_{1,r} + \boldsymbol{N}_1 \boldsymbol{J}_2^\dagger \dot{\boldsymbol{\sigma}}_{2,r}, \tag{6.59}$$

where $\dot{\boldsymbol{\sigma}}_{i,r} = \dot{\boldsymbol{\sigma}}_i + \boldsymbol{\Lambda}_i \tilde{\boldsymbol{\sigma}}_i$. When one or more scalar component in $\boldsymbol{\sigma}_a$ is active according to Algorithm 2, the desired system velocities are computed according to

$$\boldsymbol{\zeta}_d = \boldsymbol{N}_a \boldsymbol{J}_1^\dagger \dot{\boldsymbol{\sigma}}_{1,r} + \boldsymbol{N}_{a1}^A \boldsymbol{J}_2^\dagger \dot{\boldsymbol{\sigma}}_{2,r}, \tag{6.60}$$

where the null space operator $\boldsymbol{N}_a$ of the active set-based tasks is computed from (3.13) and the null space operator $\boldsymbol{N}_{a1}^A$ of the augmented Jacobian is computed according to (2.20) and (2.21).

The assumption $\boldsymbol{\zeta} = \boldsymbol{\zeta}_d$ can be satisfied by feeding back the desired states $\boldsymbol{\zeta}_d, \boldsymbol{\xi}_d$ instead of the actual states. This keeps the kinematic control loop separate from the dynamic control loop and the stability of the kinematic controller can be considered independently of the dynamic controller. As discussed in [12], this is a good assumption for industrial robotics, where fast and accurate dynamic control can be obtained with relative ease. For an AIAUV however, small inaccuracies in the dynamic control leads to error accumulation in the desired states. This can lead to set-based tasks such as collision avoidance not being activated because of errors in the estimated end-effector position. Therefore, the actual states are used for feedback as shown in Figure 6.2. In this case, the assumption requires the outer loop kinematic controller to be much slower than the inner loop dynamic controller.

6.5.1.1 Smoothing of the system velocity references

Activation and deactivation of tasks cause the system reference velocity $\boldsymbol{\zeta}_d$ to change abruptly, which often requires large accelerations. A possible solution is to smooth the reference between the previous and new reference by employing

the sigmoid smoothing function

$$\alpha(t, t_s) = \frac{1}{\pi} \arctan\big(a\,(t - t_s - b)\big) + \frac{1}{2}, \tag{6.61}$$

where t_s is the time of the last switch between velocity references. The parameters a and b determine the sharpness of the smoothing function and time delay before the transition takes place, respectively. Since activation and deactivation of tasks no longer occurs instantaneously, an activation threshold similar to that of the iCAT framework is defined. Note that strict priority among tasks is lost while transitioning between velocity reference solutions. Whenever the activation threshold is reached, the idea is to slowly control the set-based task variables toward their maximum values as long as every less restrictive velocity reference solution does not maintain the set-based variable within its extended tangent cone. Hence, (6.60) is modified with a gain for the high priority set-based task viz.

$$\boldsymbol{\zeta}_d = (\boldsymbol{A}\boldsymbol{J}_a)^\dagger \boldsymbol{A}\boldsymbol{\Lambda}_a \boldsymbol{A}^{\mathrm{T}} \boldsymbol{A}\tilde{\boldsymbol{\sigma}}_a + \boldsymbol{N}_a \boldsymbol{J}_1^\dagger \dot{\boldsymbol{\sigma}}_{1,r} + \boldsymbol{N}_{a1}^A \boldsymbol{J}_2^\dagger \dot{\boldsymbol{\sigma}}_{2,r}, \tag{6.62}$$

where $\boldsymbol{A}$ is the $p \times m_a$ activation matrix of the multidimensional set-based task $\boldsymbol{\sigma}_a \in \mathbb{R}^{m_a}$ and $\boldsymbol{\Lambda}_a$ is a diagonal $m_a \times m_a$ gain matrix. Pre-multiplication of the gain matrix by $\boldsymbol{A}$ and post multiplication by $\boldsymbol{A}^{\mathrm{T}}$ ensures that the resulting gain matrix has the appropriate dimension. By storing the time a switch between velocity references t_s occurs and the velocity reference before the switch $\boldsymbol{\zeta}_r(t_s)$, the desired system velocities at time t can be calculated from

$$\boldsymbol{\zeta}_d(t) = \big(1 - \alpha(t, t_s)\big)\, \boldsymbol{\zeta}_r(t_s) + \alpha(t, t_s)\boldsymbol{\zeta}_r(t), \tag{6.63}$$

where $\boldsymbol{\zeta}_r$ is calculated from (6.59) or (6.62) depending on whether set-based tasks are active or not.

6.5.2 iCAT framework

In the implementation of the iCAT framework, the collision avoidance task σ_a has the highest priority, followed by the actuation index task σ_b, joint limit avoidance task $\boldsymbol{\sigma}_c$, end-effector configuration task $\boldsymbol{\sigma}_1$, and finally the base positioning task $\boldsymbol{\sigma}_2$. The recursive implementation allows effortless addition and removal of tasks, which is why the set-based tasks have been placed on their own priority levels. In the set-based SRMTP framework this requires more effort and the complexity of the implementation increases significantly whenever additional set-based tasks are considered on their own priority levels. Therefore, since tasks are generally not in conflict, they are often augmented into a single set-based task in set-based SRMTP implementations. Note that the subscripts on the set-based tasks differ in this approach because of the strict priority among them.

The following activation function has been employed in the simulations

$$a_l(\sigma) = \begin{cases} 1, & \sigma \leq \sigma_{\min} \\ \frac{1}{2}\left(\cos\left(\frac{(\sigma-\sigma_{\min})\pi}{\beta}\right)+1\right), & \sigma_{\min} \leq \sigma \leq \sigma_{\min}+\beta \\ 0, & \sigma > \sigma_{\min}+\beta \end{cases}. \tag{6.64}$$

for lower-bounded set-based tasks. For tasks with an upper bound the activation function $a_u(\sigma) = 1 - a_l(\sigma)$ is used.

6.6 Set-Based Acceleration Control

When the inverse kinematics problem is resolved at the acceleration level, the control architecture is similar to Figure 6.2, except that the output of the kinematic control block is now desired system accelerations $\dot{\boldsymbol{\zeta}}_d$ instead of desired system velocities. By employing the feedback linearizing control law given by (6.53) with commanded acceleration $\boldsymbol{a}^b = \dot{\boldsymbol{\zeta}}_d$ the dynamics reduces to

$$\dot{\boldsymbol{\zeta}} = \dot{\boldsymbol{\zeta}}_d. \tag{6.65}$$

The desired system accelerations $\dot{\boldsymbol{\zeta}}_d$ are obtained from the set-based SRMTP framework at the acceleration level as described in Chapter 3. Note that end-effector configuration and base position control only consumes 9 DOFs, while an AIAUV in general has $6+n$ DOFs. Specifically, the AIAUV model used in the simulations has $n=8$ joints, resulting in a 14 DOF system. Hence, the null space velocity task from Section 6.4.6 is added at the lowest priority level for stability reasons.

When the set-based tasks are inactive, the desired acceleration reference is given by

$$\begin{aligned}\dot{\boldsymbol{\zeta}}_d = &\boldsymbol{J}_1^\dagger(\ddot{\boldsymbol{\sigma}}_{d_1} + \boldsymbol{K}_{d_1}\dot{\tilde{\boldsymbol{\sigma}}}_1 + \boldsymbol{K}_{p_1}\tilde{\boldsymbol{\sigma}}_1 - \dot{\boldsymbol{J}}_1\boldsymbol{\zeta}) \\ &+ \boldsymbol{N}_1\boldsymbol{J}_2^\dagger(\ddot{\boldsymbol{\sigma}}_{d_2} + \boldsymbol{K}_{d_2}\dot{\tilde{\boldsymbol{\sigma}}}_2 + \boldsymbol{K}_{p_2}\tilde{\boldsymbol{\sigma}}_1 - \dot{\boldsymbol{J}}_2\boldsymbol{\zeta}) \\ &- \boldsymbol{N}_{12}^A\boldsymbol{J}_3^\dagger\boldsymbol{K}_{d_3}\boldsymbol{\zeta}.\end{aligned} \tag{6.66}$$

When one or more scalar set-based tasks are active, the desired acceleration reference is given by

$$\begin{aligned}\dot{\boldsymbol{\zeta}}_d = &(\boldsymbol{A}\boldsymbol{J}_a)^\dagger\left(-\boldsymbol{A}\boldsymbol{K}_{d_a}\boldsymbol{A}^{\mathrm{T}}\boldsymbol{A}\dot{\boldsymbol{\sigma}}_a + \boldsymbol{A}\boldsymbol{K}_{p_a}\boldsymbol{A}^{\mathrm{T}}\boldsymbol{A}\tilde{\boldsymbol{\sigma}}_a - \boldsymbol{A}\dot{\boldsymbol{J}}_a\boldsymbol{\zeta}\right) \\ &+ \boldsymbol{N}_a\boldsymbol{J}_1^\dagger(\ddot{\boldsymbol{\sigma}}_{d_1} + \boldsymbol{K}_{d_1}\dot{\tilde{\boldsymbol{\sigma}}}_1 + \boldsymbol{K}_{p_1}\tilde{\boldsymbol{\sigma}}_1 - \dot{\boldsymbol{J}}_1\boldsymbol{\zeta}) \\ &+ \boldsymbol{N}_{a1}^A\boldsymbol{J}_2^\dagger(\ddot{\boldsymbol{\sigma}}_{d_2} + \boldsymbol{K}_{d_2}\dot{\tilde{\boldsymbol{\sigma}}}_2 - \boldsymbol{K}_{p_2}\tilde{\boldsymbol{\sigma}}_1 - \dot{\boldsymbol{J}}_2\boldsymbol{\zeta}) \\ &- \boldsymbol{N}_{a12}^A\boldsymbol{J}_3^\dagger\boldsymbol{K}_{d_3}\boldsymbol{\zeta},\end{aligned} \tag{6.67}$$

where $\boldsymbol{A}$ is the activation matrix returned by Algorithm 2.

6.6.1 Smoothing the acceleration references

The acceleration reference can smoothened by utilizing the sigmoid smoothing function in (6.61). The time t_s when the acceleration reference changes because of the activation/deactivation of a set-based task, as well as the acceleration reference $\dot{\boldsymbol{\zeta}}_r(t_s^-)$ just before the activation or deactivation is stored in memory at simulation run-time. Then, the smoothened acceleration reference can be

calculated from

$$\dot{\boldsymbol{\zeta}}_d = \left(1 - \alpha(t, t_s)\right) \dot{\boldsymbol{\zeta}}_r(t_s^-) + \alpha(t, t_s)\dot{\boldsymbol{\zeta}}_r(t), \tag{6.68}$$

where $\dot{\boldsymbol{\zeta}}_r$ is calculated from (6.66) or (6.67) according to which tasks where active at time t_s^- and time t.

6.7 Set-Based Operational Space Control

In operational space control, the overall control architecture is shown in Figure 6.1. For any of the control laws outlined in Section 4.5 the control torques only differ in how the acceleration term is defined. Therefore, the control torque for any of the operational space controllers whenever no set-based tasks are active is given by

$$\boldsymbol{\tau} = \boldsymbol{J}_1^{\mathrm{T}} \boldsymbol{M}_1 \boldsymbol{a}_1 + \boldsymbol{N}_2 \boldsymbol{J}_2^{\mathrm{T}} \boldsymbol{M}_2 \boldsymbol{a}_2 + \boldsymbol{N}_3 \boldsymbol{J}_3^{\mathrm{T}} \boldsymbol{M}_3 \boldsymbol{a}_3 + \boldsymbol{C}\boldsymbol{\zeta} + \boldsymbol{D}\boldsymbol{\zeta} + \boldsymbol{g}. \tag{6.69}$$

To determine whether one or more scalar set-based tasks within $\boldsymbol{\sigma}_a$ should be activated or deactivated, Algorithm 2 is employed. The predicted unconstrained task velocity $\dot{\boldsymbol{\sigma}}_a$ is found by integrating the body-fixed accelerations

$$\dot{\boldsymbol{\zeta}} = \boldsymbol{M}^{-1} \left(\boldsymbol{\tau} - \boldsymbol{C}\boldsymbol{\zeta} - \boldsymbol{D}\boldsymbol{\zeta} - \boldsymbol{g}\right), \tag{6.70}$$

where $\boldsymbol{\tau}$ is given by (6.69). The unconstrained task velocities are then computed according to

$$\dot{\boldsymbol{\sigma}}_a = \boldsymbol{J}_a \boldsymbol{\zeta}. \tag{6.71}$$

If one or more set-based tasks are active, the control torque is given by

$$\begin{aligned} \boldsymbol{\tau} = {} & \left(\boldsymbol{A}\boldsymbol{J}_a\right)^{\mathrm{T}} \boldsymbol{M}_a \boldsymbol{a}_a + \boldsymbol{N}_1 \boldsymbol{J}_1^{\mathrm{T}} \boldsymbol{M}_1 \boldsymbol{a}_1 + \boldsymbol{N}_2 \boldsymbol{J}_2^{\mathrm{T}} \boldsymbol{M}_2 \boldsymbol{a}_2 + \boldsymbol{N}_3 \boldsymbol{J}_3^{\mathrm{T}} \boldsymbol{M}_3 \boldsymbol{a}_3 \\ & + \boldsymbol{C}\boldsymbol{\zeta} + \boldsymbol{D}\boldsymbol{\zeta} + \boldsymbol{g}, \end{aligned} \tag{6.72}$$

where $\boldsymbol{A}$ is the activation matrix returned by Algorithm 2, the task specific inertia matrix $\boldsymbol{M}_i$ is given by (4.37) or (4.49) and the acceleration reference $\boldsymbol{a}_i$ is given by (4.43) or (4.45). If the acceleration reference is given by (4.43), then every acceleration reference changes whenever set-based tasks are activated or deactivated. The task specific inertia matrix for the active set-based tasks is computed according to

$$\boldsymbol{M}_a = \left(\boldsymbol{A}\boldsymbol{J}_a\boldsymbol{M}^{-1}\left(\boldsymbol{A}\boldsymbol{J}_a\right)^{\mathrm{T}}\right)^{-1} \tag{6.73}$$

$$= \left(\boldsymbol{A}\boldsymbol{J}_a\boldsymbol{M}^{-1}\boldsymbol{J}_a^{\mathrm{T}}\boldsymbol{A}^{\mathrm{T}}\right)^{-1}, \tag{6.74}$$

while the null-space operators are computed according to (4.17) and (4.18) with $\boldsymbol{N}_a = \boldsymbol{I}$ when set-based tasks are active, and $\boldsymbol{N}_1 = \boldsymbol{I}$ when no set-based tasks are active. Finally, note that there are only three combinations of acceleration reference and task inertia matrix, since (4.43) should not be used with (4.49).

6.7.1 Method 1: Dynamically decoupled tasks

In this approach, the task specific inertia matrix $\boldsymbol{M}_i$ is given by (4.37) while the acceleration references $\boldsymbol{a}_i$ are given by (4.43) resulting in the linear task dynamics (4.44), whenever all tasks are compatible. The control torque when no set-based tasks are active given by (6.66) becomes

$$\begin{aligned}\boldsymbol{\tau} = {} & \boldsymbol{J}_1^{\mathrm{T}}\boldsymbol{M}_1\left(\boldsymbol{K}_{p,1}\tilde{\boldsymbol{\sigma}}_1 - \boldsymbol{K}_{d,1}\dot{\boldsymbol{\sigma}}_1 - \dot{\boldsymbol{J}}_1\boldsymbol{\zeta}\right) \\ & + \boldsymbol{N}_2\boldsymbol{J}_2^{\mathrm{T}}\boldsymbol{M}_2\left(\boldsymbol{K}_{p,2}\tilde{\boldsymbol{\sigma}}_2 - \boldsymbol{K}_{d,2}\dot{\boldsymbol{\sigma}}_2 - \dot{\boldsymbol{J}}_2\boldsymbol{\zeta} - \boldsymbol{J}_{\boldsymbol{3}}\boldsymbol{M}^{-1}\boldsymbol{J}_1^{\mathrm{T}}\boldsymbol{M}_1\boldsymbol{a}_1\right) \\ & + \boldsymbol{N}_3\boldsymbol{J}_3^{\mathrm{T}}\boldsymbol{M}_3\left(-\boldsymbol{K}_{d,3}\boldsymbol{\zeta} - \boldsymbol{J}_3\boldsymbol{M}^{-1}\left(\boldsymbol{J}_1^{\mathrm{T}}\boldsymbol{M}_1\boldsymbol{a}_1 + \boldsymbol{N}_2\boldsymbol{J}_2^{\mathrm{T}}\boldsymbol{M}_2\boldsymbol{a}_2\right)\right) \\ & + \boldsymbol{D}\boldsymbol{\zeta} + \boldsymbol{C}\boldsymbol{\zeta} + \boldsymbol{g}.\end{aligned} \tag{6.75}$$

However, since the lowest priority task is only present for stability reasons, there is no point in complicating the control law by compensating for higher priority tasks at the lowest priority level. Moreover, when compensating for the task

dynamics of higher priority tasks, the acceleration reference changes whenever set-based tasks are activated/deactivated, which may incur larger discontinuities in the control torque. The control torque when no set-based tasks are active is therefore modified as

$$\begin{aligned}\boldsymbol{\tau} = &\boldsymbol{J}_1^{\mathrm{T}}\boldsymbol{M}_1\left(\boldsymbol{K}_{p,1}\tilde{\boldsymbol{\sigma}}_1 - \boldsymbol{K}_{d,1}\dot{\boldsymbol{\sigma}}_1 - \dot{\boldsymbol{J}}_1\boldsymbol{\zeta}\right)\\ &+ \boldsymbol{N}_2\boldsymbol{J}_2^{\mathrm{T}}\boldsymbol{M}_2\left(\boldsymbol{K}_{p,2}\tilde{\boldsymbol{\sigma}}_2 - \boldsymbol{K}_{d,2}\dot{\boldsymbol{\sigma}}_2 - \dot{\boldsymbol{J}}_2\boldsymbol{\zeta} - \boldsymbol{J}_2\boldsymbol{M}^{-1}\boldsymbol{J}_1^{\mathrm{T}}\boldsymbol{M}_1\boldsymbol{a}_1\right)\\ &- \boldsymbol{N}_3\boldsymbol{J}_3^{\mathrm{T}}\boldsymbol{M}_3\boldsymbol{K}_{d,3}\boldsymbol{\zeta}\\ &+ \boldsymbol{D}\boldsymbol{\zeta} + \boldsymbol{C}\boldsymbol{\zeta} + \boldsymbol{g},\end{aligned} \tag{6.76}$$

where $\boldsymbol{a}_1$ can be inferred by comparing (6.76) with (6.69). When one or more scalar set-based tasks within $\boldsymbol{\sigma}_a$ are active the control torque is given by

$$\begin{aligned}\boldsymbol{\tau} = &\left(\boldsymbol{A}\boldsymbol{J}_a\right)^{\mathrm{T}}\boldsymbol{M}_a\left(\boldsymbol{A}\boldsymbol{K}_{p,a}\boldsymbol{A}^{\mathrm{T}}\boldsymbol{A}\tilde{\boldsymbol{\sigma}}_a - \boldsymbol{A}\boldsymbol{K}_{d,a}\boldsymbol{A}^{\mathrm{T}}\boldsymbol{A}\dot{\boldsymbol{\sigma}}_a - \boldsymbol{A}\dot{\boldsymbol{J}}_a\boldsymbol{\zeta}\right)\\ &+ \boldsymbol{N}_1\boldsymbol{J}_1^{\mathrm{T}}\boldsymbol{M}_1\left(\boldsymbol{K}_{p,1}\tilde{\boldsymbol{\sigma}}_1 - \boldsymbol{K}_{d,1}\dot{\boldsymbol{\sigma}}_1 - \dot{\boldsymbol{J}}_1\boldsymbol{\zeta} - \boldsymbol{J}_1\boldsymbol{M}^{-1}(\boldsymbol{A}\boldsymbol{J}_a)^{\mathrm{T}}\boldsymbol{M}_a\boldsymbol{a}_a\right)\\ &+ \boldsymbol{N}_2\boldsymbol{J}_2^{\mathrm{T}}\boldsymbol{M}_2\Big(\boldsymbol{K}_{p,2}\tilde{\boldsymbol{\sigma}}_2 - \boldsymbol{K}_{d,2}\dot{\boldsymbol{\sigma}}_2 - \dot{\boldsymbol{J}}_2\boldsymbol{\zeta}\\ &-\boldsymbol{J}_2\boldsymbol{M}^{-1}\left(\left(\boldsymbol{A}\boldsymbol{J}_a\right)^{\mathrm{T}}\boldsymbol{M}_a\boldsymbol{a}_a + \boldsymbol{N}_1\boldsymbol{J}_1^{\mathrm{T}}\boldsymbol{M}_1\boldsymbol{a}_1\right)\Big)\\ &- \boldsymbol{N}_3\boldsymbol{J}_3^{\mathrm{T}}\boldsymbol{M}_3\boldsymbol{K}_{d,3}\boldsymbol{\zeta}\\ &+ \boldsymbol{D}\boldsymbol{\zeta} + \boldsymbol{C}\boldsymbol{\zeta} + \boldsymbol{g},\end{aligned} \tag{6.77}$$

where $\boldsymbol{a}_a$ and $\boldsymbol{a}_1$ can be inferred by comparing (6.77) and (6.72), and $\boldsymbol{A}$ is the activation matrix returned by Algorithm 2.

6.7.2 Method 2: Omitting compensation terms in the acceleration references

By computing the task specific inertia matrix $\boldsymbol{M}_i$ from (4.37) and the acceleration reference $\boldsymbol{a}_i$ from (4.45), the control torque from (6.69) with no active set-based

tasks becomes

$$\begin{aligned}\boldsymbol{\tau} = &\boldsymbol{J}_1^{\mathrm{T}}\boldsymbol{M}_1\left(\boldsymbol{K}_{p,1}\tilde{\boldsymbol{\sigma}}_1 - \boldsymbol{K}_{d,1}\dot{\boldsymbol{\sigma}}_1 - \dot{\boldsymbol{J}}_1\boldsymbol{\zeta}\right)\\ &+ \boldsymbol{N}_2\boldsymbol{J}_2^{\mathrm{T}}\boldsymbol{M}_2\left(\boldsymbol{K}_{p,2}\tilde{\boldsymbol{\sigma}}_2 - \boldsymbol{K}_{d,2}\dot{\boldsymbol{\sigma}}_2 - \dot{\boldsymbol{J}}_2\boldsymbol{\zeta}\right)\\ &- \boldsymbol{N}_3\boldsymbol{J}_3^{\mathrm{T}}\boldsymbol{M}_3\boldsymbol{K}_{d,3}\dot{\boldsymbol{\sigma}}_3 + \boldsymbol{C}\boldsymbol{\zeta} + \boldsymbol{D}\boldsymbol{\zeta} + \boldsymbol{g}.\end{aligned} \tag{6.78}$$

When one or more scalar set-based tasks within $\boldsymbol{\sigma}_a$ are active, the control torques are computed according to

$$\begin{aligned}\boldsymbol{\tau} = &\left(\boldsymbol{A}\boldsymbol{J}_a\right)^{\mathrm{T}}\boldsymbol{M}_a\left(\boldsymbol{A}\boldsymbol{K}_{p,a}\boldsymbol{A}^{\mathrm{T}}\boldsymbol{A}\tilde{\boldsymbol{\sigma}}_a - \boldsymbol{A}\boldsymbol{K}_{d,a}\boldsymbol{A}^{\mathrm{T}}\boldsymbol{A}\dot{\boldsymbol{\sigma}}_a - \boldsymbol{A}\dot{\boldsymbol{J}}_a\boldsymbol{\zeta}\right)\\ &+ \boldsymbol{N}_1\boldsymbol{J}_1^{\mathrm{T}}\boldsymbol{M}_1\left(\boldsymbol{K}_{p,1}\tilde{\boldsymbol{\sigma}}_1 - \boldsymbol{K}_{d,1}\dot{\boldsymbol{\sigma}}_1 - \dot{\boldsymbol{J}}_1\boldsymbol{\zeta}\right)\\ &+ \boldsymbol{N}_2\boldsymbol{J}_2^{\mathrm{T}}\boldsymbol{M}_2\left(\boldsymbol{K}_{p,2}\tilde{\boldsymbol{\sigma}}_2 - \boldsymbol{K}_{d,2}\dot{\boldsymbol{\sigma}}_2 - \dot{\boldsymbol{J}}_2\boldsymbol{\zeta}\right)\\ &- \boldsymbol{N}_3\boldsymbol{J}_3^{\mathrm{T}}\boldsymbol{M}_3\boldsymbol{K}_{d,3}\dot{\boldsymbol{\sigma}}_3 + \boldsymbol{C}\boldsymbol{\zeta} + \boldsymbol{D}\boldsymbol{\zeta} + \boldsymbol{g},\end{aligned} \tag{6.79}$$

where $\boldsymbol{A}$ is the activation matrix obtained from Algorithm 2. The task specific inertia matrix $\boldsymbol{M}_a$ for the set-based task is computed at every iteration according to (6.74).

6.7.3 Method 3: Omitting the null space operator from the task specific inertia matrix

When the task specific inertia matrix is computed from (4.49), the control torque is given by (6.78) when no set-based tasks are active, and from (6.79) when set-based tasks are active.

6.7.4 Smoothing the control torques

The control torques in Section 6.7 exhibit discontinuities whenever a set-based task is activated or deactivated. Continuity of the control torques can be obtained by employing the sigmoid smoothing function in (6.61) in order to obtain a continuous transition from the previous control torque to the current one. By

storing the switching time t_s when the control torque changes because of the activation/deactivation of one or more set-based tasks and the control torque $\boldsymbol{\tau}(t_s^-)$ just before the switch, the smoothened control torque can be computed according to

$$\boldsymbol{\tau}(t) = \big(1 - \alpha(t, t_s)\big)\, \boldsymbol{\tau}(t_s^-) + \alpha(t, t_s)\boldsymbol{\tau}(t), \tag{6.80}$$

where the control torques $\boldsymbol{\tau}$ are calculated from (6.69) or (6.72) depending on which tasks were active at time t_s^- and time t.

Chapter 7

Simulations

In this chapter simulation results of the control frameworks from Chapter 6 are presented. For the kinematic controllers, perfect knowledge of the dynamic parameters in the dynamic controller is always assumed. For the operational space controllers, simulations are presented when perfect parameter knowledge is assumed, and when the added mass terms are omitted from the inertia matrix. This is motivated by the observation that the inertia matrix is used in the null space operators for operational space control, which may lead to imperfect null space projections and interference from lower priority tasks in the dynamics of higher priority tasks.

7.1 Control Objectives

7.1.1 Equality tasks

Only set-point regulation is considered in the simulations, where the end-effector configuration task consists of a series of constant steps occurring at $t = [0, 50, 100, 150, 200, 250, 300, 350]^{\mathrm{T}}$. The base positioning task is constant and does not change, with the goal of minimizing movement of the base while

reconfiguring the end-effector. Note that after $t = 350\,\text{s}$, the end-effector position and base position are no longer compatible, this is intended to serve as a test of incompatible tasks, where it is expected that the end-effector task will be executed perfectly, while steady state errors will be observed in the base positioning task.

Unfortunately, since no reference model is employed to smooth out the steps in the desired end-effector configuration, discontinuities will be present in the control inputs whenever the set-points change suddenly. Hence, acceleration feedforward by numeric differentiation is also infeasible for the iCAT framework. Moreover, it becomes harder to spot the discontinuities incurred by the activation/deactivation of set-based tasks.

7.1.2 Set-based tasks

The obstacle to be avoided by the collision avoidance task is a sphere with radius $r_{\text{obs}} = 0.3\,\text{m}$ centered at $\boldsymbol{p}_{\text{obs}} = [2.5\,\text{m}, 0.5\,\text{m}, -10\,\text{m}]^{\text{T}}$, where the coordinates are given in a North-West-Up coordinate frame. In order to make sure that the AIAUV avoids the obstacle at all times, a safety threshold is defined such that the minimum distance to the obstacle becomes slightly larger. The valid domains for all set-based tasks are shown in Table 7.1.

Table 7.1: The valid domains for the set-based tasks.

	σ_{a_1}	$\sigma_{a_2,i}$	σ_{a_3}
$\sigma_{\min}$	$r_{obs} + 0.05\,\text{m}$	$-60°$	0.1
$\sigma_{\max}$	∞	$60°$	∞

7.2 Implementation Specifics

Note that the Matlab simulator employs a local North-West-Up coordinate frame as opposed to the local North-East-Down coordinate frame defined in Section

5.1. The coordinate frames of the base, links and end-effector remain the same as in Chapter 5.

7.2.1 The mode definition

Since every simulation except for the iCAT framework employ the extended set-based SRMTP framework from Section 3.3 to activate and deactivate tasks, the mode concept defined in [4] has to be redefined slightly. The modes in the simulations are defined according to:

- **Mode 1**: No set-based tasks are active.
- **Mode 2-9**: 1-8 active joint limit tasks, collision avoidance and actuation index tasks inactive.
- **Mode 10**: Collision avoidance task active, everything else inactive.
- **Mode 11-18**: 1-8 active joint limit tasks, collision avoidance task active, actuation index task inactive.
- **Mode 20**: Actuation index task active, everything else inactive.
- **Mode 21-28**: 1-8 active joint limit tasks, collision avoidance task inactive, actuation index task active.
- **Mode 30-37**: 1-8 active joint limit tasks, collision avoidance and actuation index tasks active.

7.2.2 Smoothing function

The smoothing function from (6.61) is used with the parameters $a = 110$ and $b = 0.05$ for all smoothing simulations.

7.2.3 Control allocation

Since actuator configuration matrix singularities are avoided by the introduction of the actuation index task, the simple unconstrained control allocation scheme from Section 6.3 is employed within every control framework. The control input $\boldsymbol{u} \in \mathbb{R}^{15}$ containing the allocated thrust and joint torques are obtained from the commanded forces and torques $\boldsymbol{\tau} \in \mathbb{R}^{14}$ through

$$\boldsymbol{u} = \boldsymbol{B}^{\dagger}\boldsymbol{\tau} = \boldsymbol{B}^{\mathrm{T}}\left(\boldsymbol{B}\boldsymbol{B}^{\mathrm{T}}\right)^{-1}\boldsymbol{\tau}. \tag{7.1}$$

The control parameters are tuned such that the resulting control inputs have sufficiently small values instead of saturating the control inputs $\boldsymbol{u}$ after allocation.

7.3 Kinematic Control

7.3.1 Set-based velocity control

For set-based velocity control, the dynamic control law is given by (6.53) and (6.54) where

$$\boldsymbol{K}_p = \begin{bmatrix} \boldsymbol{K}_{p,\boldsymbol{V}} & \boldsymbol{0}_{6\times 8} \\ \boldsymbol{0}_{8\times 6} & \boldsymbol{K}_{p,\boldsymbol{\theta}} \end{bmatrix}, \quad \boldsymbol{K}_i = \begin{bmatrix} \boldsymbol{K}_{i,\boldsymbol{V}} & \boldsymbol{0}_{6\times 8} \\ \boldsymbol{0}_{8\times 6} & \boldsymbol{K}_{i,\boldsymbol{\theta}} \end{bmatrix}. \tag{7.2}$$

The numerical values of the control parameters are shown in Table 7.2

Table 7.2: Dynamic control parameters for velocity-based control.

	$\boldsymbol{V}$	$\boldsymbol{\theta}$
λ	0.6	0.7
$\boldsymbol{K}_p$	$2\lambda_{\boldsymbol{V}}\boldsymbol{I}_{6\times 6}$	$2\lambda_{\boldsymbol{\theta}}\boldsymbol{I}_{8\times 8}$
$\boldsymbol{K}_i$	$\lambda_{\boldsymbol{V}}^2\boldsymbol{I}_{6\times 6}$	$\lambda_{\boldsymbol{\theta}}^2\boldsymbol{I}_{8\times 8}$

7.3.1.1 Set-based SRMTP framework

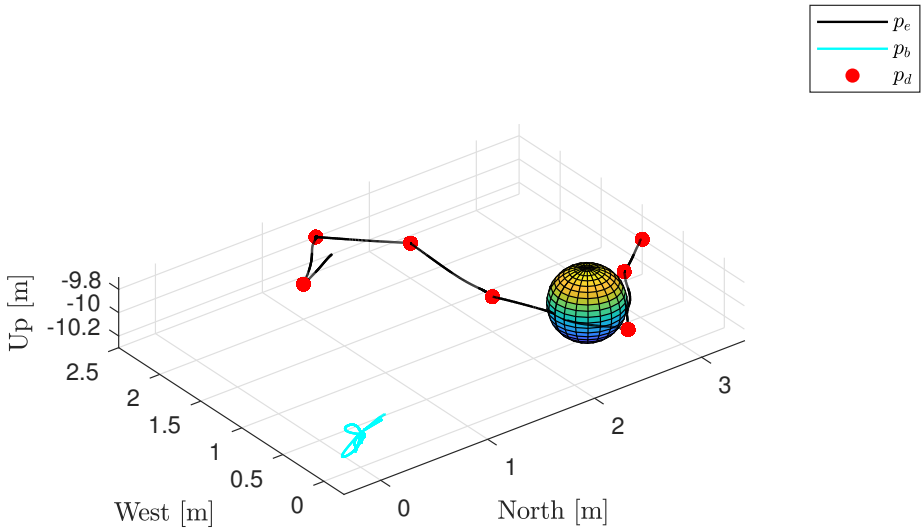
For set-point regulation, the task space references $\dot{\boldsymbol{\sigma}}_{i,r}$ in the desired system velocity references given by (6.59) and (6.60) reduce to

$$\dot{\boldsymbol{\sigma}}_{i,r} = \boldsymbol{\Lambda}_i \tilde{\boldsymbol{\sigma}}_i. \tag{7.3}$$

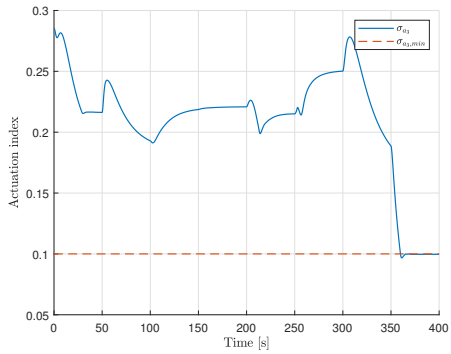
The task space gains in Table 7.3 resulted in acceptable behavior for a wide variety of set-point regulation tasks. Simulation results are shown in Figure 7.1.

Table 7.3: SRMTP task space gains.

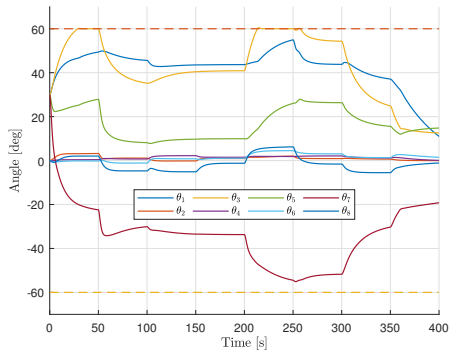
	$\boldsymbol{\sigma}_1$	$\boldsymbol{\sigma}_2$
$\boldsymbol{\Lambda}$	$0.2\boldsymbol{I}_6$	$0.2\boldsymbol{I}_3$



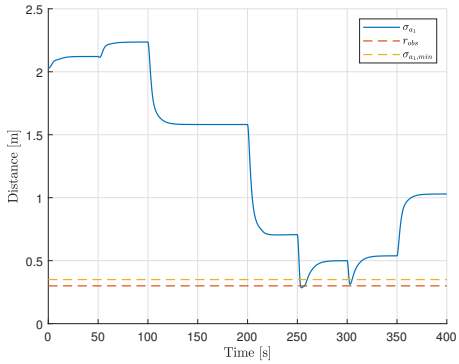
(a) North-West-Up plot. p_e, p_b and p_d represents the end-effector, base and the desired end-effector position, respectively.



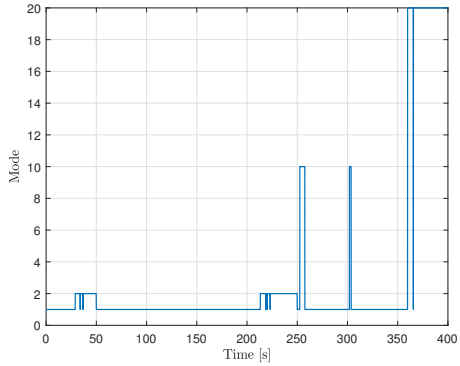
(b) The actuation index and its minimum value.



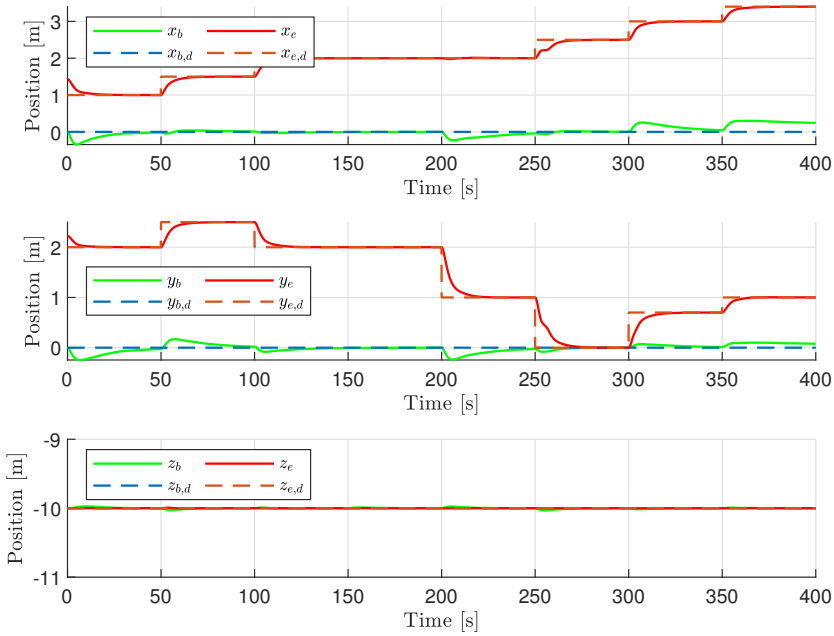
(c) Joint angles and their maximum and minimum values.



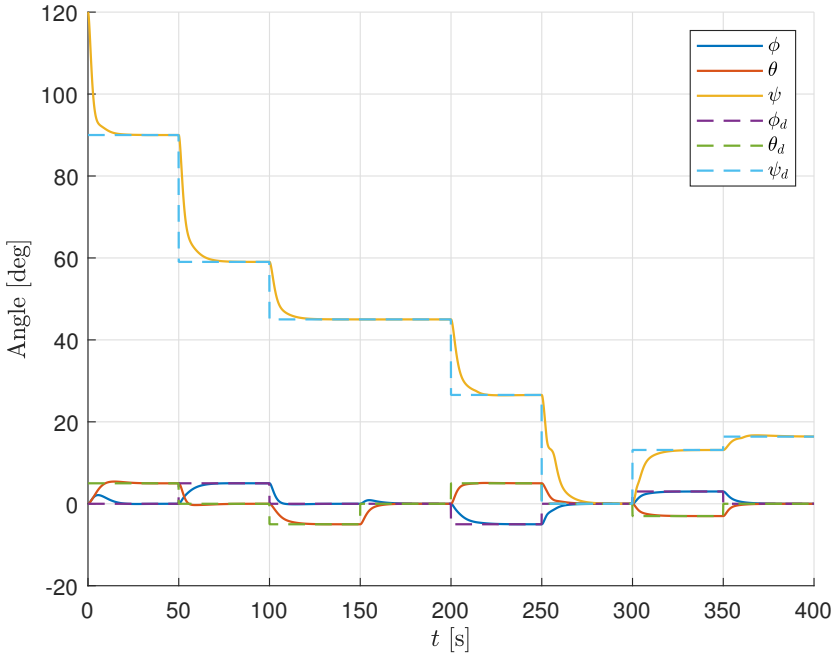
(d) Distance from the end-effector to the spherical obstacle.



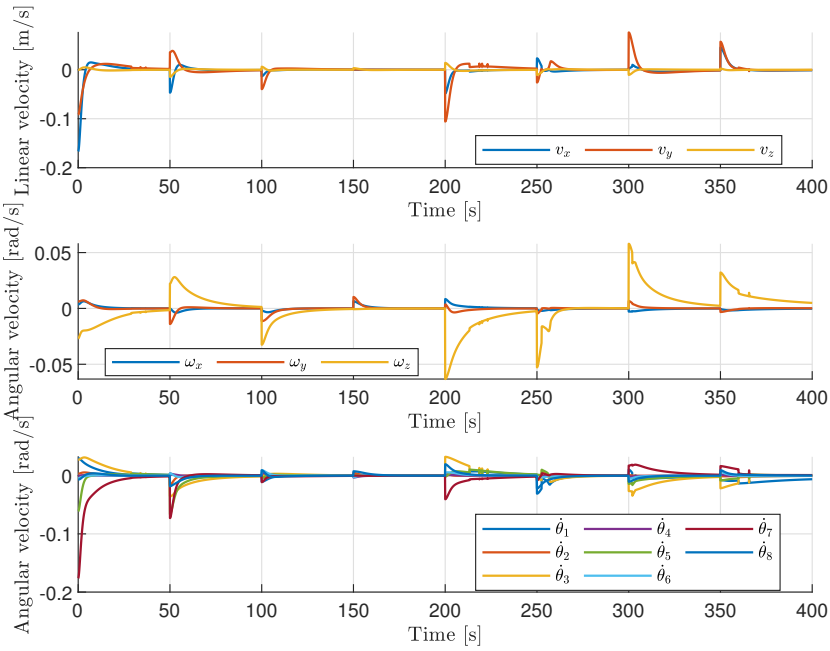
(e) The mode represents which set-based tasks are active at any given time.



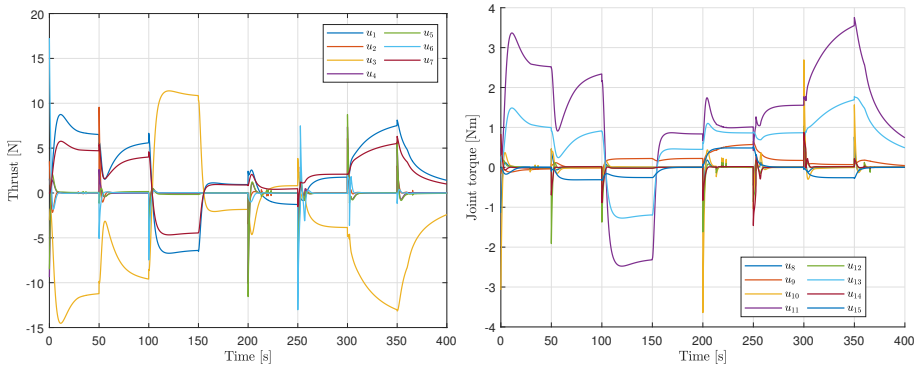
(f) End-effector, desired end-effector, base and desired base positions.



(g) End-effector orientation.



(h) Velocity references generated by the kinematic controller.



(i) Thruster input generated by the control allocation algorithm.

(j) Joint torque inputs generated by the control allocation algorithm

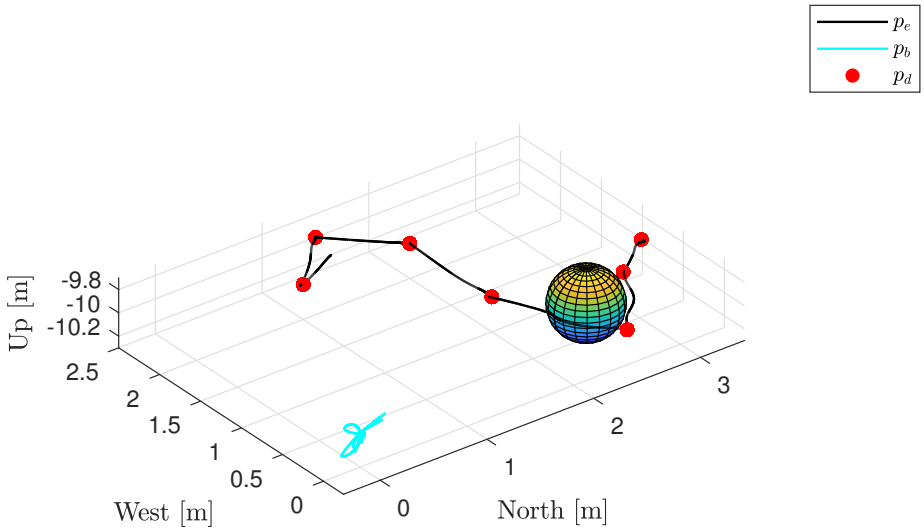
Figure 7.1: Simulation results for kinematic velocity control within the set-based SRMTP framework.

Smoothing the velocity references The velocity references generated by the kinematic controller can be smoothened as discussed in Section 6.5.1.1. The control parameters and the activation thresholds for the set-based tasks are shown in Table 7.4. The set based gain matrix $\mathbf{\Lambda}_a$ is a diagonal 10×10 matrix

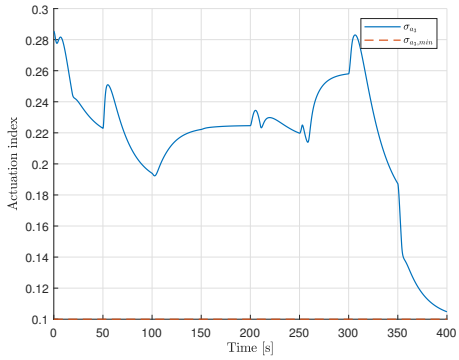
Table 7.4: Smoothened SRMTP task space gains $\mathbf{\Lambda}$ and activation thresholds β.

	σ_{a_1}	$\boldsymbol{\sigma}_{a_2}$	σ_{a_3}	$\boldsymbol{\sigma}_1$	$\boldsymbol{\sigma}_2$
$\mathbf{\Lambda}$	0	0.05	$0.05\boldsymbol{I}_8$	$0.2\boldsymbol{I}_6$	$0.2\boldsymbol{I}_3$
β	0.1 m	0.05	5°		

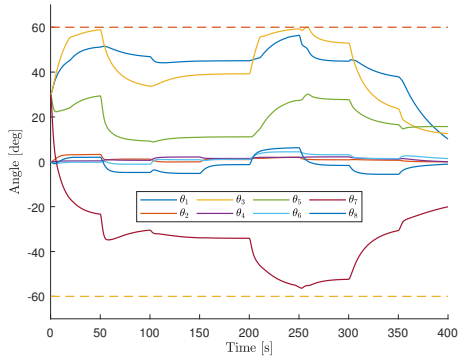
where the first diagonal element corresponds to the collision avoidance task, the following eight correspond to the joint limit avoidance task, and the final tenth element corresponds to the actuation index task. Simulation results are shown in Figure 7.2.



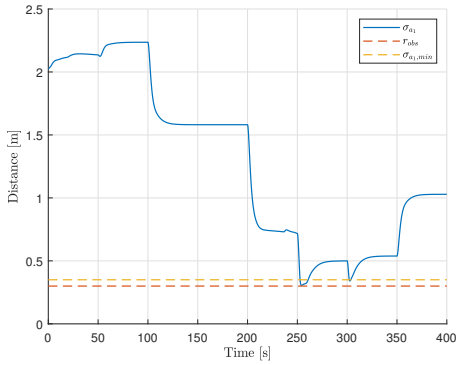
(a) North-West-Up plot. p_e, p_b and p_d represents the end-effector, base and the desired end-effector position, respectively.



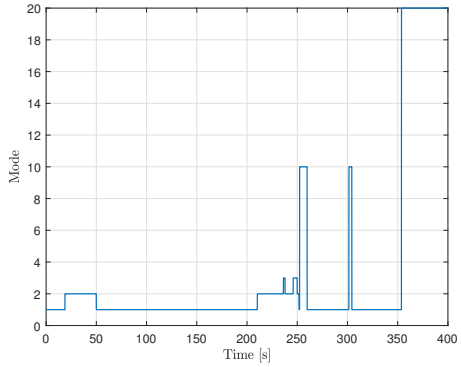
(b) The actuation index and its minimum value.



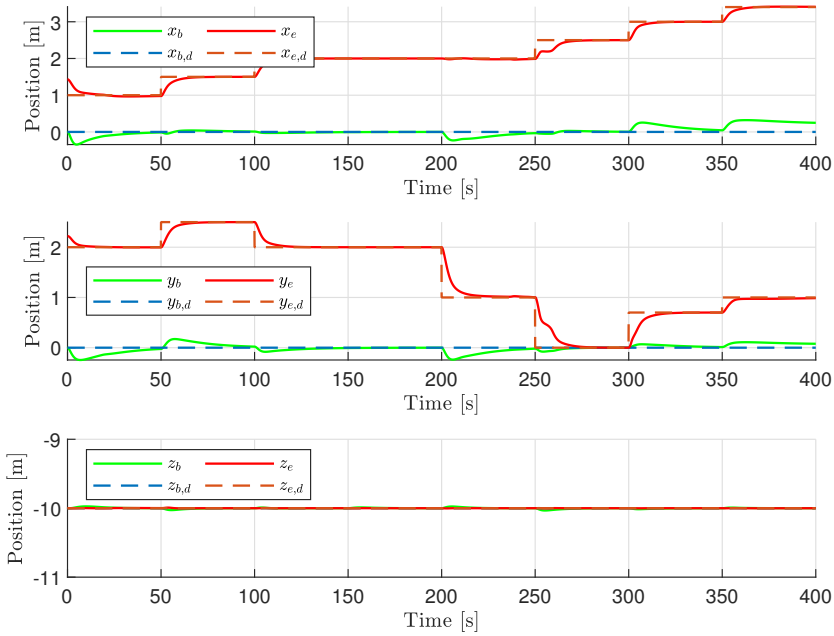
(c) Joint angles and their maximum and minimum values.



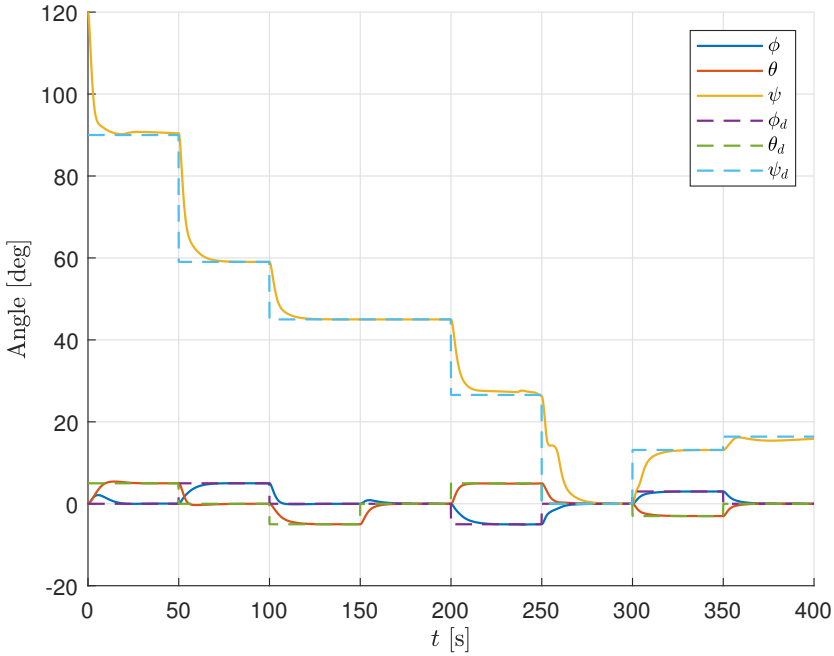
(d) Distance from the end-effector to the spherical obstacle.



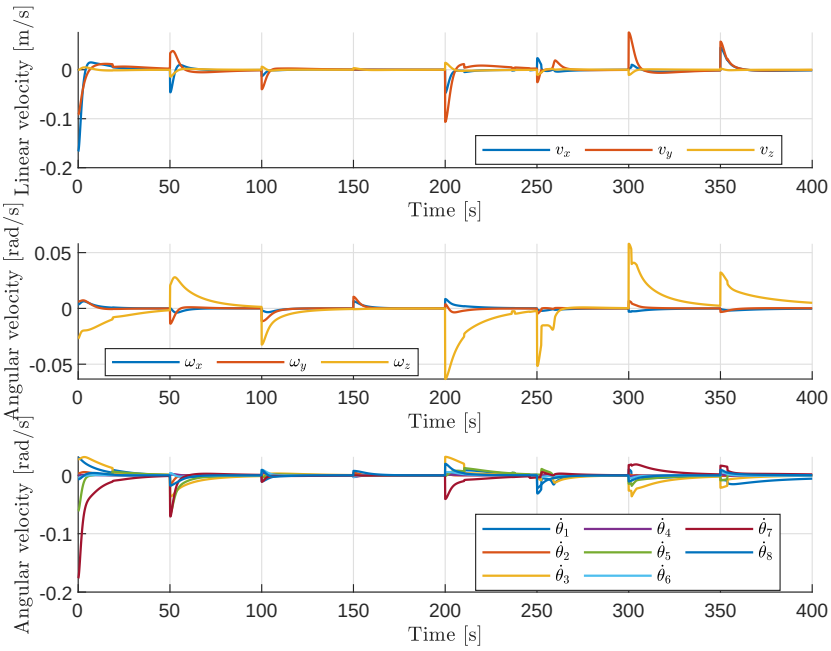
(e) The mode represents which set-based tasks are active at any given time.



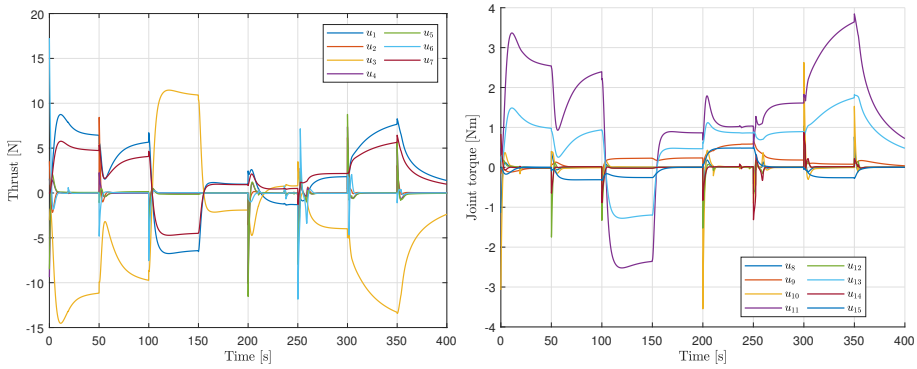
(f) End-effector, desired end-effector, base and desired base position.



(g) End-effector orientation.



(h) Velocity references generated by the kinematic controller.



(i) Thruster input generated by the control allocation algorithm.

(j) Joint torque inputs generated by the control allocation algorithm

Figure 7.2: Simulation results for the smoothened version of the set-based SRMTP kinematic control framework.

7.3.1.2 iCAT framework

The assigned feedback reference rates given by (3.15) and the activation thresholds β are computed according to Table 7.5 for the set-based tasks, where σ_a is the collision avoidance task, σ_b is the actuation index task and $\boldsymbol{\sigma}_c$ is the joint limit avoidance task. The i-subscript on the joint limit avoidance task indicates that every joint has the same gain γ and reference σ^*. The highest priority

Table 7.5: iCAT control parameters for the set-based tasks.

	σ_a	σ_b	$\boldsymbol{\sigma}_{c_i}$
σ^*	$r_{obs} + 0.1$	$\sigma_{b_{\min}} + 0.05$	0
γ	0.15	0.25	0.1
β	0.1 m	0.05	5°

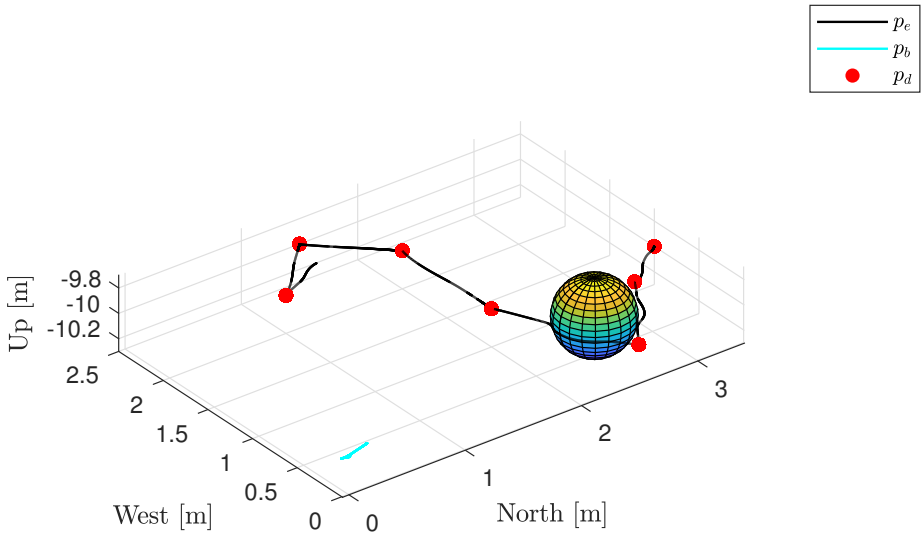
equality task is the end-effector configuration task $\boldsymbol{\sigma}_1$ with gain

$$\Gamma_1 = \begin{bmatrix} 0.1\boldsymbol{I}_{3\times3} & \boldsymbol{0}_{3\times3} \\ \boldsymbol{0}_{3\times3} & 0.1\boldsymbol{I}_{3\times3} \end{bmatrix}, \tag{7.4}$$

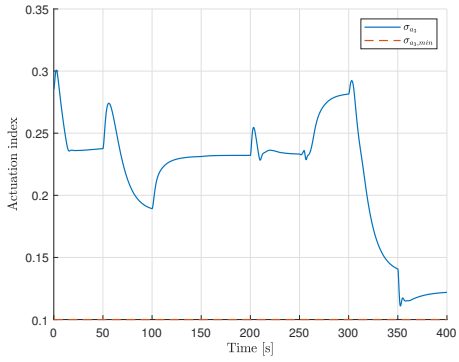
followed by the base positioning task $\boldsymbol{\sigma}_2$ with gain

$$\Gamma_2 = 0.1\boldsymbol{I}_{3\times3}. \tag{7.5}$$

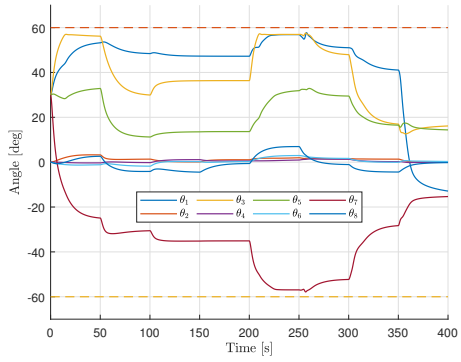
Moreover, the standard deviation of the diagonal elements in $\boldsymbol{P}$ used in (3.21) is set to $s = 40$.



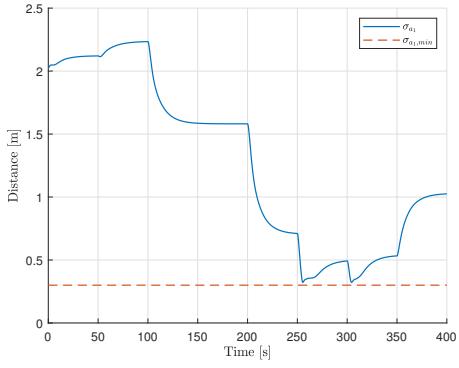
(a) North-West-Up plot. p_e, p_b and p_d represents the end-effector, base and the desired end-effector position, respectively.



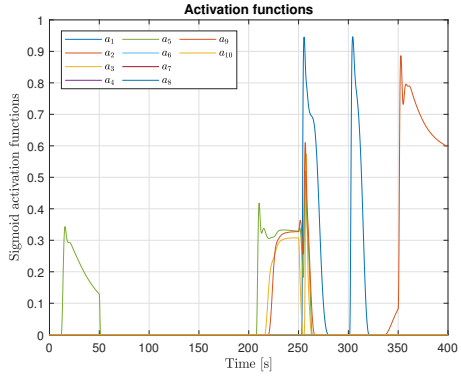
(b) The actuation index and its minimum value.



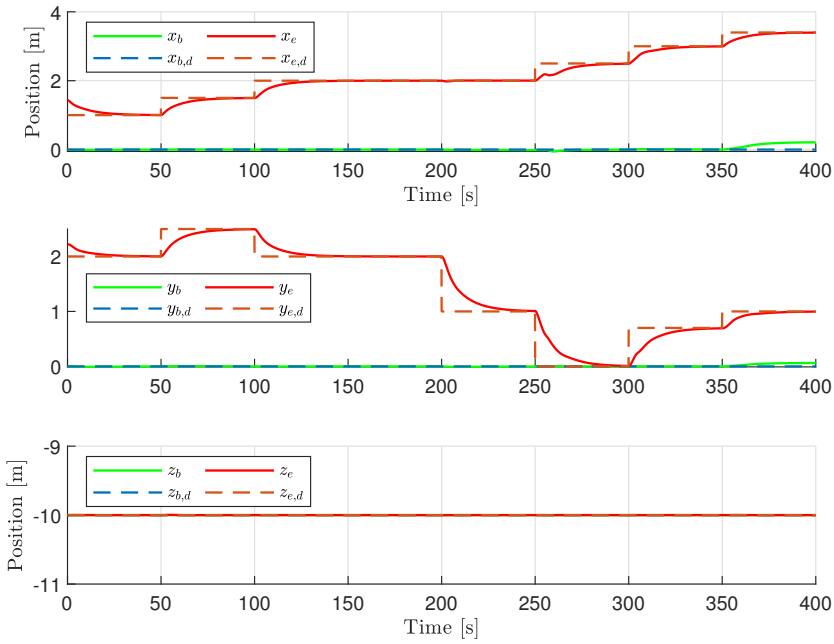
(c) Joint angles and their maximum and minimum values.



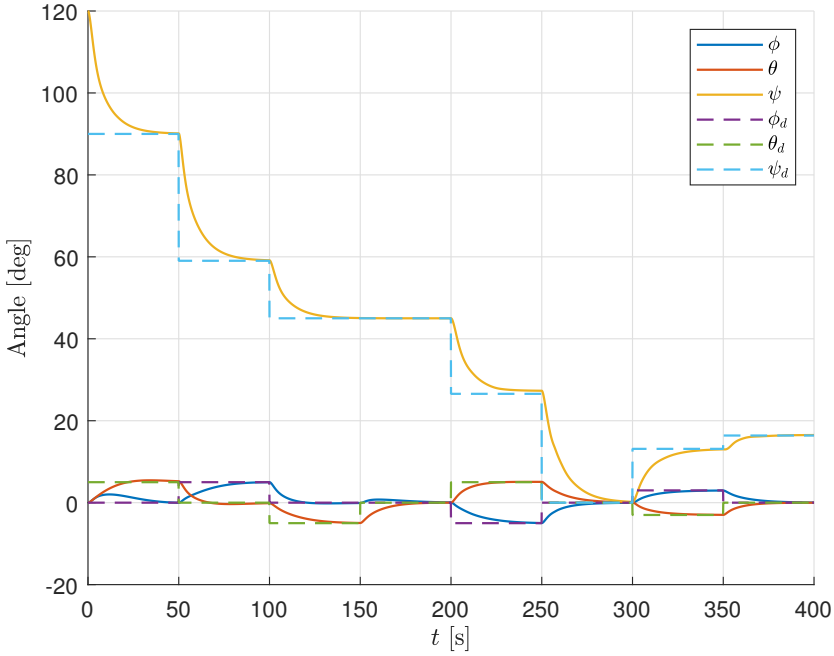
(d) Distance from the end-effector to the spherical obstacle.



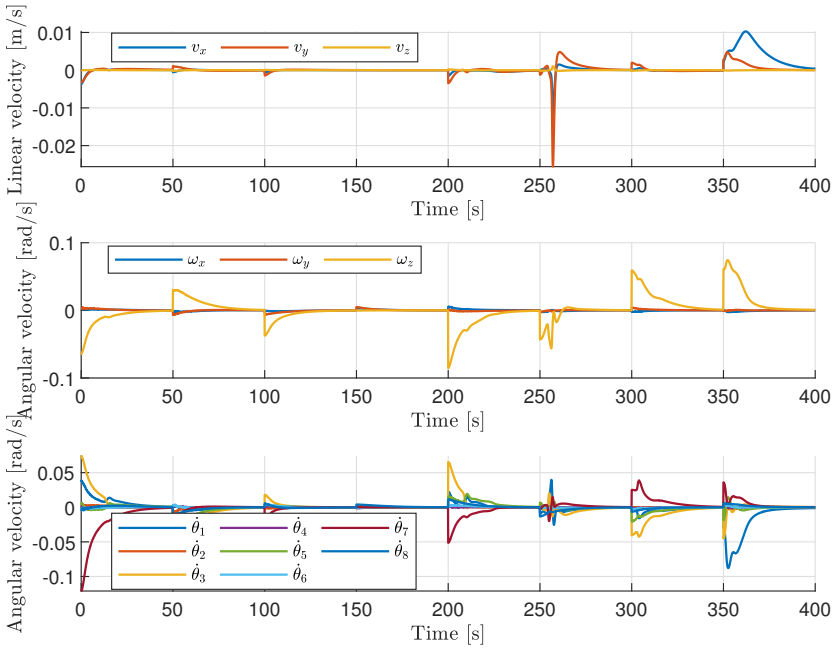
(e) The sigmoid activation functions representing how actively controlled set-based tasks are.



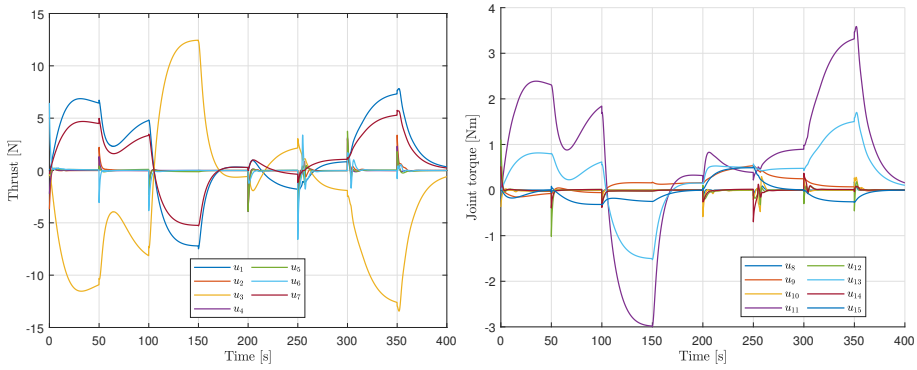
(f) End-effector, desired end-effector, base and desired base position.



(g) End-effector orientation.



(h) Velocity references generated by the kinematic controller.



(i) Thruster input generated by the control allocation algorithm.

(j) Joint torque inputs generated by the control allocation algorithm

Figure 7.3: Simulation results for the iCAT framework.

7.3.2 Set-based acceleration control

From the initial simulations it was evident that a non-zero proportional gain for the set-based tasks resulting in excessive activation and deactivation of tasks as depicted in Figure 7.4. Therefore, all proportional gains for the augmented

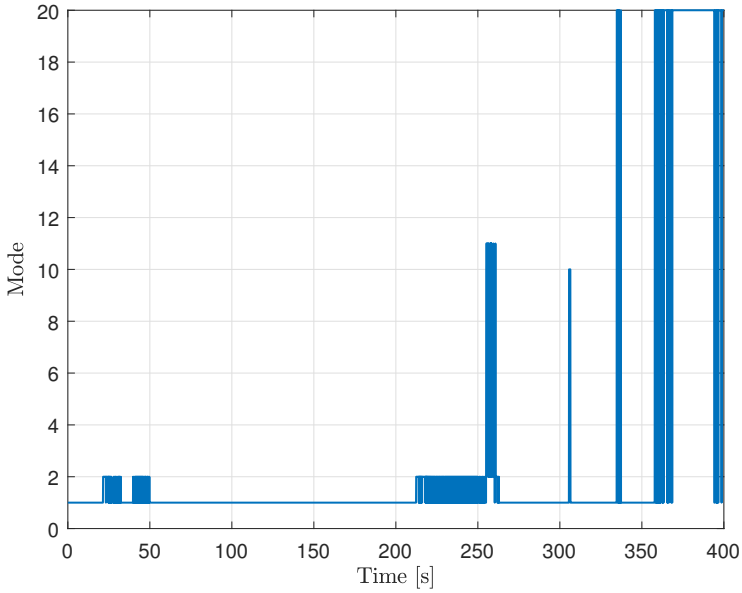


Figure 7.4: Excessive activation and deactivation of tasks, represented by what mode that is currently active.

set-based task are set to zero. The derivative gains are given by

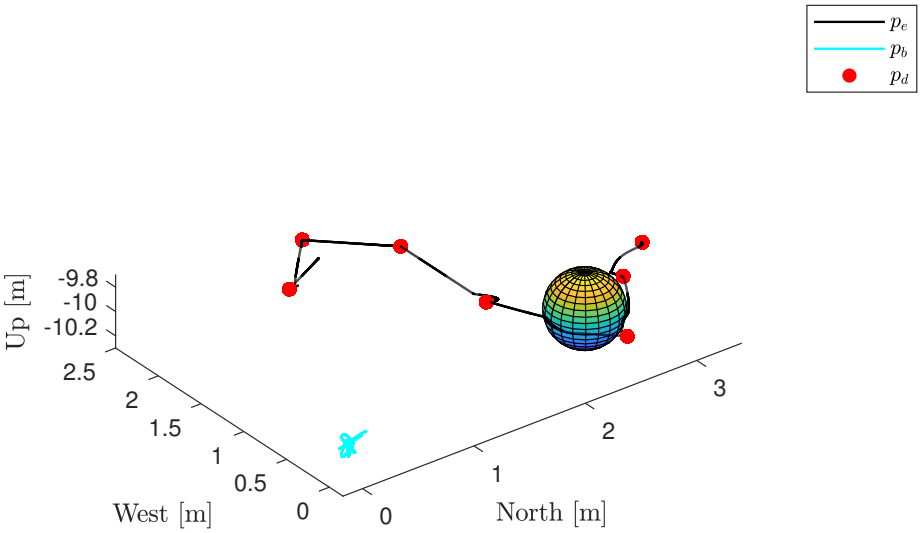
$$\boldsymbol{K}_{d_a} = \text{diag}\left\{k_{a_1}, k_{a_2}, k_{a_2}, k_{a_2}, k_{a_2}, k_{a_2}, k_{a_2}, k_{a_2}, k_{a_2}, k_{a_3}\right\}, \tag{7.6}$$

where the collision avoidance, joint limit and actuation index task gains are $k_{a_1} = 4.5, k_{a_2} = 2.5$ and $k_{a_3} = 4$, respectively. The equality task gains are

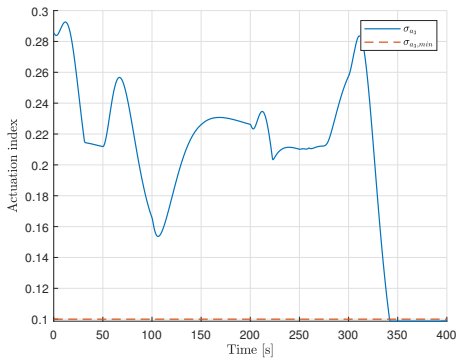
$$\boldsymbol{K}_{p_1} = \begin{bmatrix} \lambda_1^2 \boldsymbol{I}_3 & \boldsymbol{0}_{3\times 3} \\ \boldsymbol{0}_{3\times 3} & \lambda_{\text{att}}^2 \boldsymbol{I}_3 \end{bmatrix}, \quad \boldsymbol{K}_{p_2} = \lambda_2^2 \boldsymbol{I}_3, \tag{7.7}$$

$$\boldsymbol{K}_{d_1} = \begin{bmatrix} 2\lambda_1 \boldsymbol{I}_3 & \boldsymbol{0}_{3\times 3} \\ \boldsymbol{0}_{3\times 3} & 2\lambda_{\text{att}} \boldsymbol{I}_3 \end{bmatrix}, \quad \boldsymbol{K}_{d_2} = 2\lambda_2 \boldsymbol{I}_3, \quad \boldsymbol{K}_{d_3} = k_{\boldsymbol{\zeta}} \boldsymbol{I}_{14}, \tag{7.8}$$

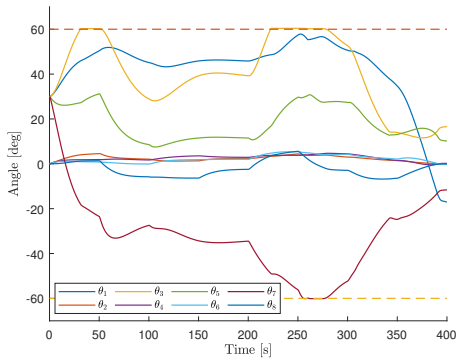
where $\lambda_1 = 0.15$, $\lambda_{\text{att}} = 0.3, \lambda_2 = 0.1$ and $k_{\boldsymbol{\zeta}} = 0.6$. Simulation results with these parameters are shown in Figure 7.5.



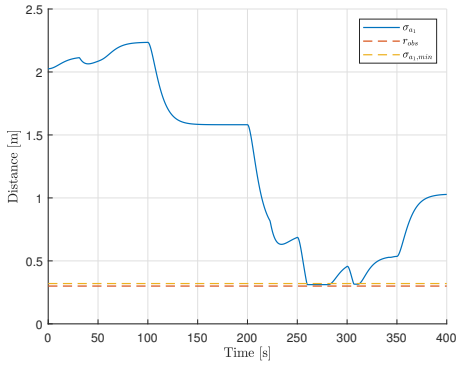
(a) North-West-Up plot. p_e, p_b and p_d represents the end-effector, base and the desired end-effector position, respectively.



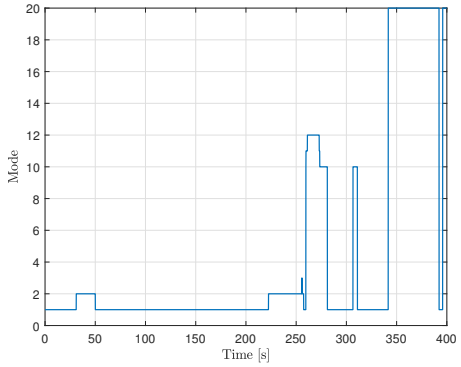
(b) The actuation index and its minimum value.



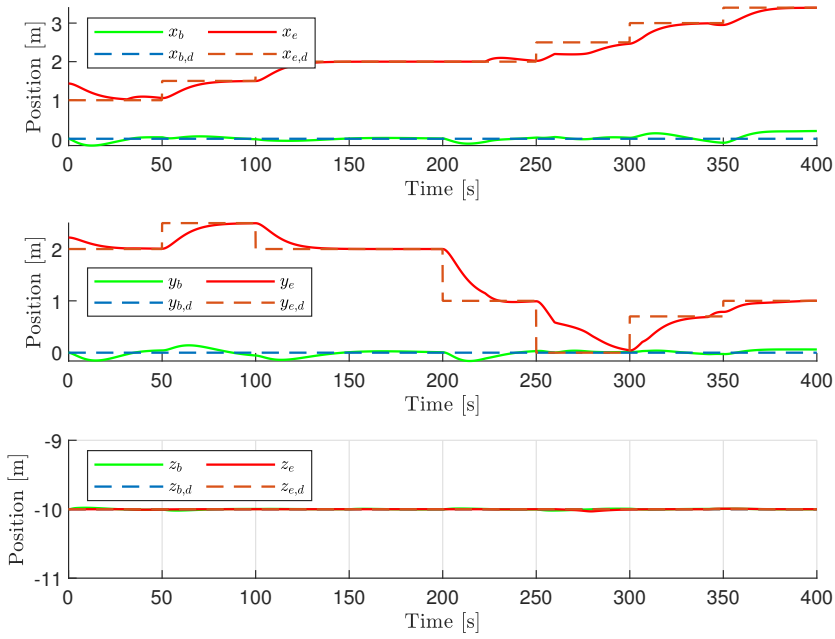
(c) Joint angles and their maximum and minimum values.



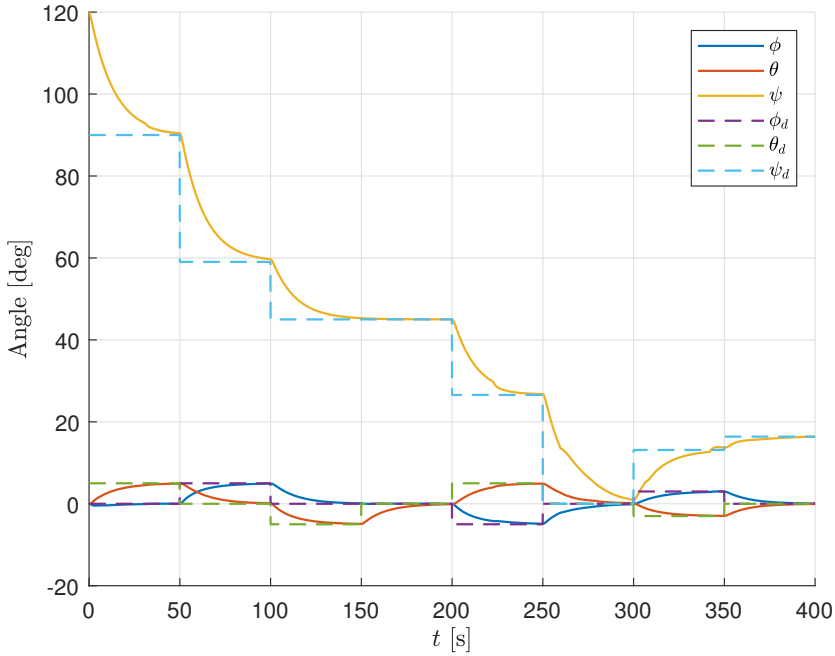
(d) Distance from the end-effector to the spherical obstacle.



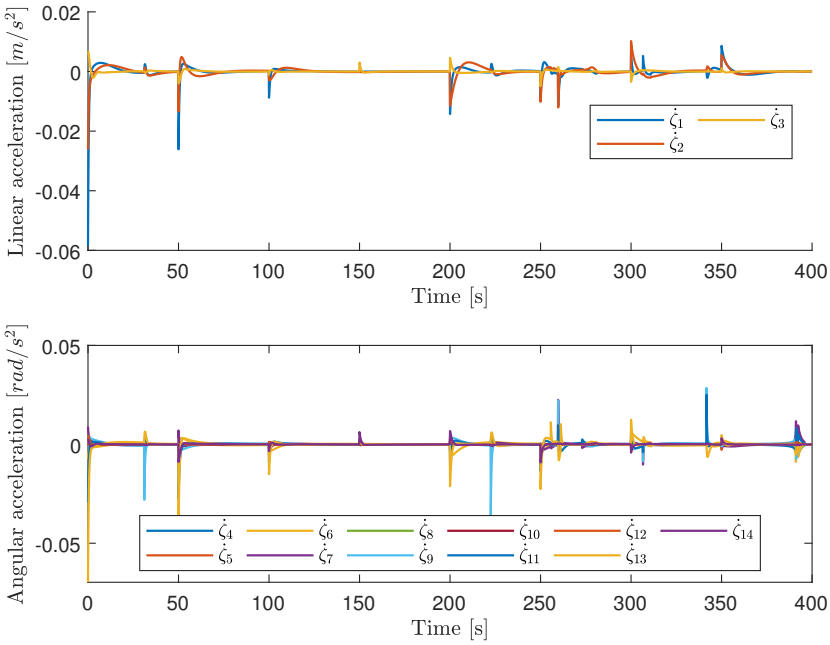
(e) The mode represents which set-based tasks are active at any given time.



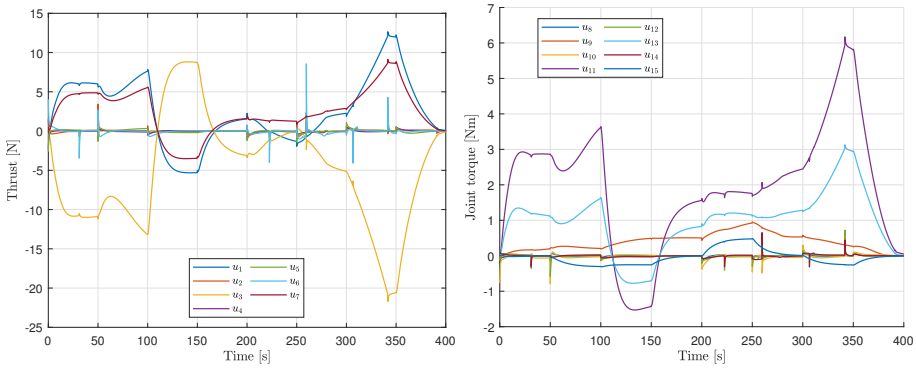
(f) End-effector, desired end-effector, base and desired base position.



(g) End-effector orientation.



(h) Acceleration references generated by the kinematic controller.



(i) Thruster input generated by the control allocation algorithm.

(j) Joint torque inputs generated by the control allocation algorithm

Figure 7.5: Simulation results for acceleration-level set-based SRMTP kinematic control.

7.3.2.1 Smoothing the acceleration references

The acceleration references can be smoothened as discussed in Section 6.6.1. The equality task gains are given by (7.7) and (7.8) with $\lambda_1 = 0.2$, $\lambda_{\text{att}} = 0.35$, $\lambda_2 = 0.1$ and $k_{\boldsymbol{\zeta}} = 0.6$. The activation thresholds β for the set-based tasks are set according to Table 7.6. Whenever the activation thresholds are reached,

Table 7.6: Activation thresholds for set-based tasks.

	σ_{a_1}	$\boldsymbol{\sigma}_{a_2}$	σ_{a_3}
β	0.05 m	5°	0.03

the set-based tasks are slowly controlled toward the boundary of their valid sets. This is achieved by having a small proportional gain for the set-based task, combined with a significantly larger derivative gain. The proportional and derivative gain matrices are given by

$$\boldsymbol{K}_{p_a} = \text{diag}\left\{\gamma_{a_1}, \gamma_{a_2}, \gamma_{a_2}, \gamma_{a_2}, \gamma_{a_2}, \gamma_{a_2}, \gamma_{a_2}, \gamma_{a_2}, \gamma_{a_2}, \gamma_{a_3}\right\}, \tag{7.9}$$

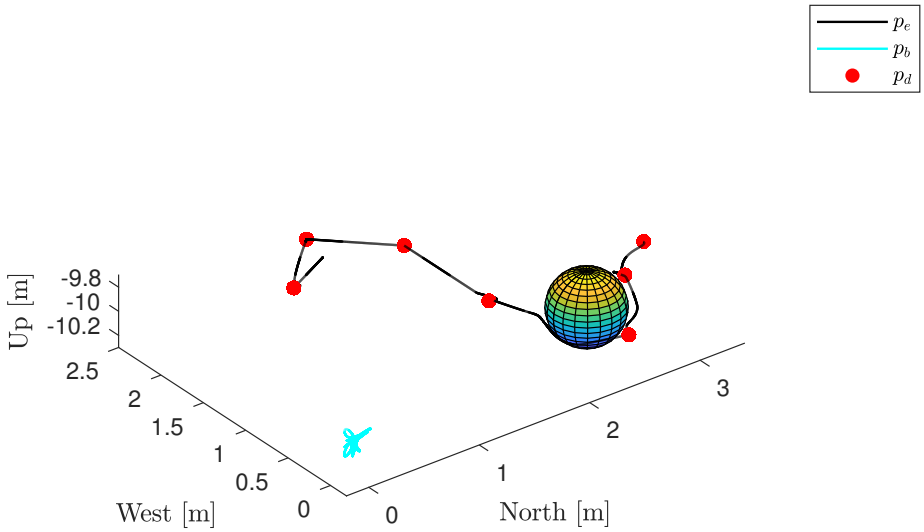
$$\boldsymbol{K}_{d_a} = \text{diag}\left\{k_{a_1}, k_{a_2}, k_{a_2}, k_{a_2}, k_{a_2}, k_{a_2}, k_{a_2}, k_{a_2}, k_{a_2}, k_{a_3}\right\}, \tag{7.10}$$

where the task specific gains are shown in Table 7.7. Simulation results are

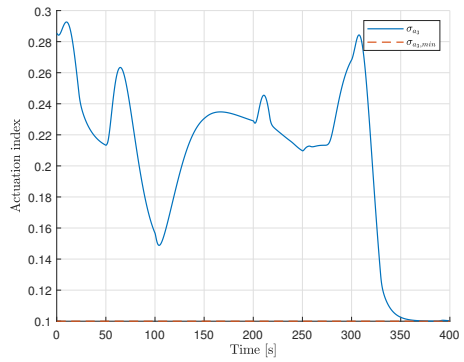
Table 7.7: Proportional and derivative gains for the augmented set-based task.

	σ_{a_1}	$\boldsymbol{\sigma}_{a_2}$	σ_{a_3}
γ	0	0.1	0.1
k	1.2	1	1

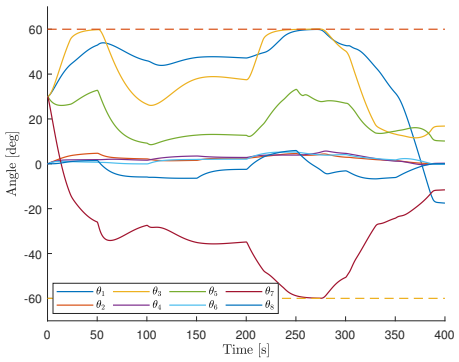
shown in Figure 7.6.



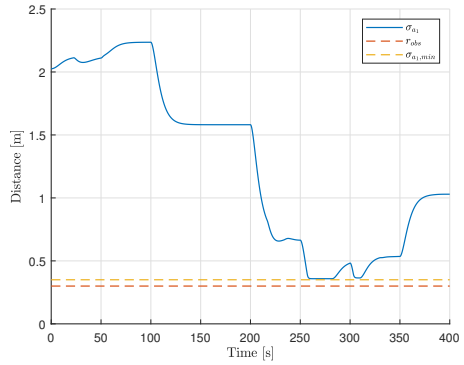
(a) North-West-Up plot. p_e, p_b and p_d represents the end-effector, base and the desired end-effector position, respectively.



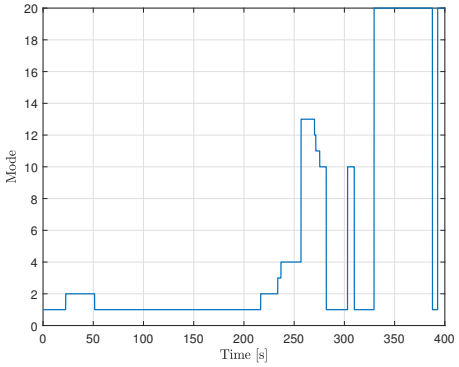
(b) The actuation index and its minimum value.



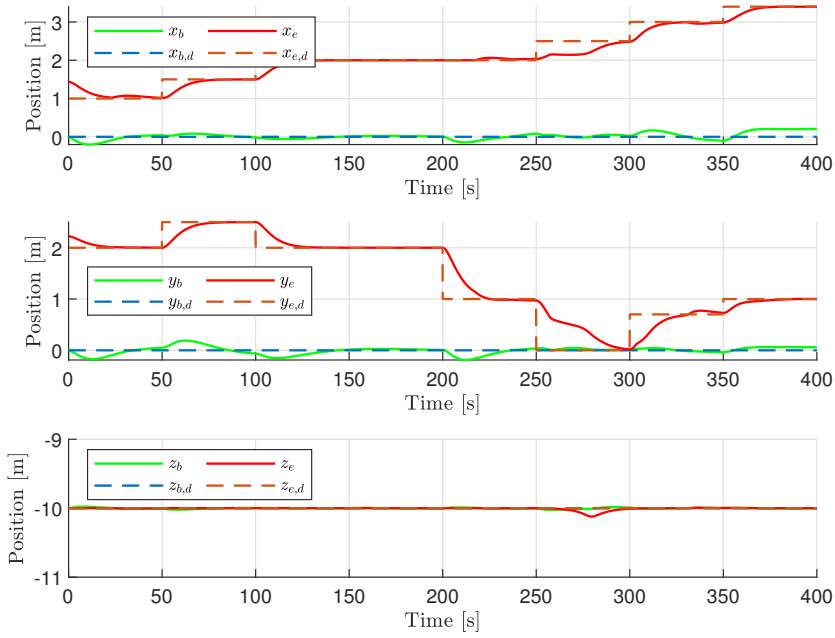
(c) Joint angles and their maximum and minimum values.



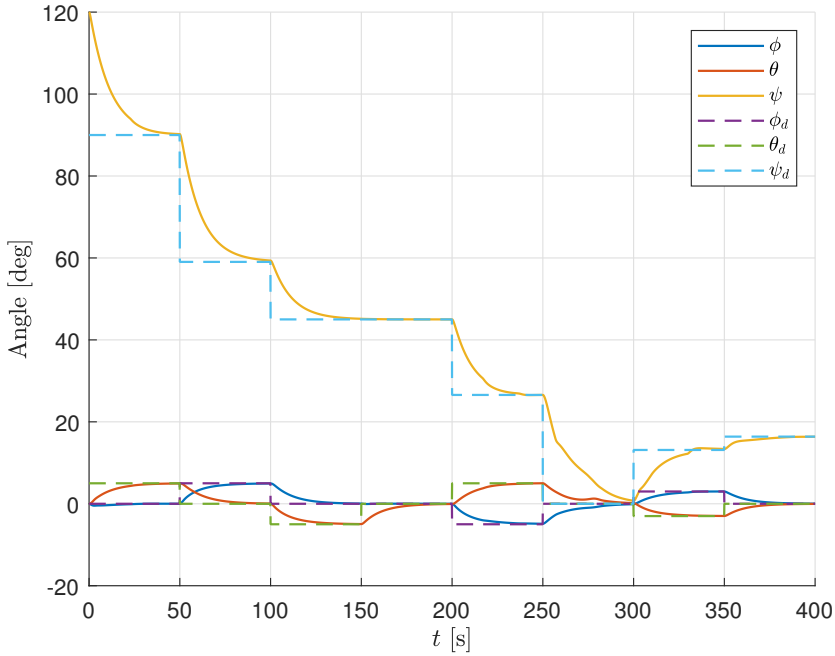
(d) Distance from the end-effector to the spherical obstacle.



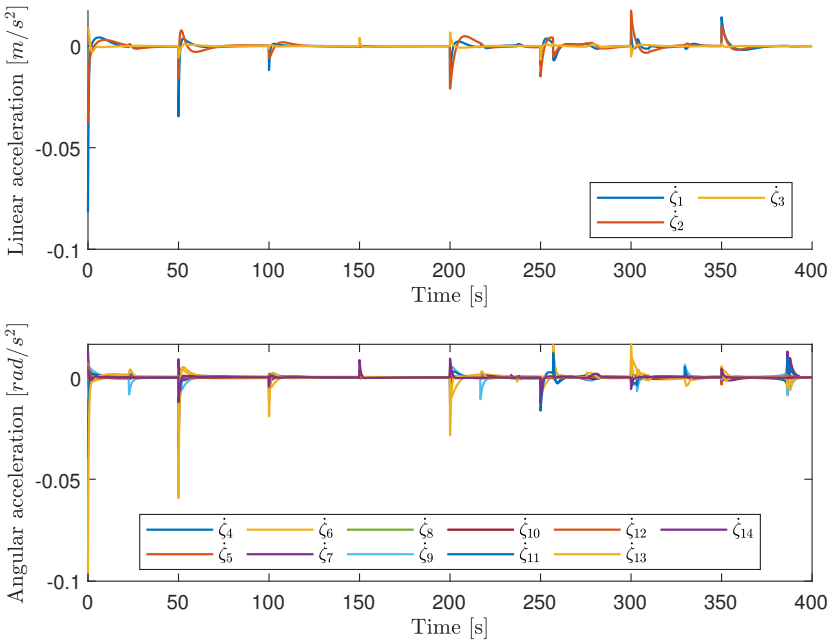
(e) The mode represents which set-based tasks are active at any given time.



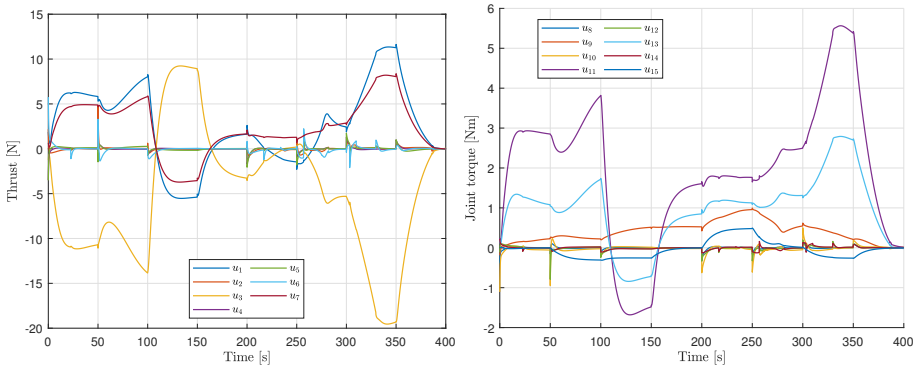
(f) End-effector, desired end-effector, base and desired base position.



(g) End-effector orientation.



(h) Acceleration references generated by the kinematic controller.



(i) Thruster input generated by the control allocation algorithm.

(j) Joint torque inputs generated by the control allocation algorithm

Figure 7.6: Simulation results for acceleration-level set-based SRMTP kinematic control with acceleration reference smoothing.

7.4 Operational Space Control

Since the inertia matrix is included in the computation of the null space operators, uncertainties in the inertia matrix may distort the null space projections such that interference from lower priority tasks affect higher priority tasks. The null space operator is statically consistent for any values of the inertia matrix, as long as it is positive definite. However, it is not dynamically consistent whenever the inertia matrix is not fully known. It is therefore of interest to investigate how the various operational space control schemes from Section 6.7 perform in simulations when the inertia matrix is not perfectly known. This is done by assuming perfect knowledge of the rigid-body inertia matrix, while the added mass matrix is unknown.

7.4.1 Control parameters

The gain matrices from (6.76), (6.77), (6.78) and (6.79) are diagonal matrices given by

$$\boldsymbol{K}_{p,a} = \mathbf{0}_{10\times 10} \tag{7.11}$$

$$\boldsymbol{K}_{d,a} = \operatorname{diag}\left\{k_{a_1}, k_{a_2}, k_{a_2}, k_{a_2}, k_{a_2}, k_{a_2}, k_{a_2}, k_{a_2}, k_{a_2}, k_{a_3}\right\}, \tag{7.12}$$

$$\boldsymbol{K}_{p_1} = \begin{bmatrix} \lambda_1^2 \boldsymbol{I}_3 & \mathbf{0}_{3\times 3} \\ \mathbf{0}_{3\times 3} & \lambda_{\text{att}}^2 \boldsymbol{I}_3 \end{bmatrix}, \quad \boldsymbol{K}_{p_2} = \lambda_2^2 \boldsymbol{I}_3, \tag{7.13}$$

$$\boldsymbol{K}_{d_1} = \begin{bmatrix} 2\lambda_1 \boldsymbol{I}_3 & \mathbf{0}_{3\times 3} \\ \mathbf{0}_{3\times 3} & 2\lambda_{\text{att}} \boldsymbol{I}_3 \end{bmatrix}, \quad \boldsymbol{K}_{d_2} = 2\lambda_2 \boldsymbol{I}_3, \tag{7.14}$$

$$\boldsymbol{K}_{d_3} = k_{\boldsymbol{\zeta}} \boldsymbol{I}_{11}. \tag{7.15}$$

Note that the proportional gains for all set-based tasks are set to zero to avoid rapid activation/deactivation of tasks around the boundaries of their active sets as discussed in 7.3.2.

7.4.1.1 Smoothing

When smoothing is employed, activation thresholds for the set-based tasks have to be defined. Moreover, non-zero proportional gains no longer result in excessive chattering in the activation and deactivation of tasks. Therefore, small proportional gains for the set-based tasks are defined such that the set-based tasks are slowly controlled toward the boundary of their valid sets after the activation thresholds are reached. The activation thresholds β for the set-based tasks are set according to Table 7.8, while the proportional and derivative gain matrices are given by

$$\boldsymbol{K}_{p,a} = \operatorname{diag}\left\{\gamma_{a_1}, \gamma_{a_2}, \gamma_{a_2}, \gamma_{a_2}, \gamma_{a_2}, \gamma_{a_2}, \gamma_{a_2}, \gamma_{a_2}, \gamma_{a_2}, \gamma_{a_3}\right\}, \tag{7.16}$$

$$\boldsymbol{K}_{d,a} = \operatorname{diag}\left\{k_{a_1}, k_{a_2}, k_{a_2}, k_{a_2}, k_{a_2}, k_{a_2}, k_{a_2}, k_{a_2}, k_{a_2}, k_{a_3}\right\}. \tag{7.17}$$

Table 7.8: Activation thresholds for set-based tasks.

	σ_{a_1}	$\boldsymbol{\sigma}_{a_2}$	σ_{a_3}
β	0.07 m	5°	0.03

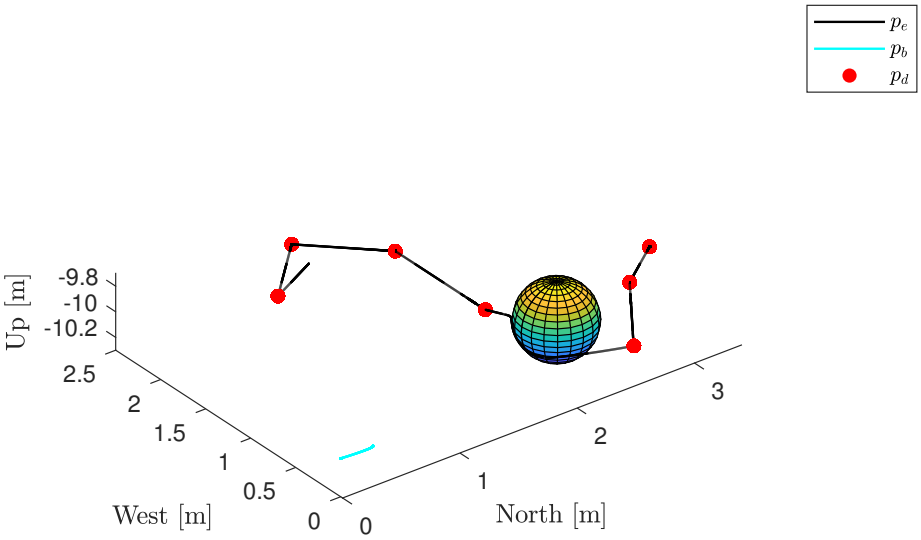
7.4.2 Method 1: Fully linearized task dynamics

The control parameters for this approach are given in Table 7.9.

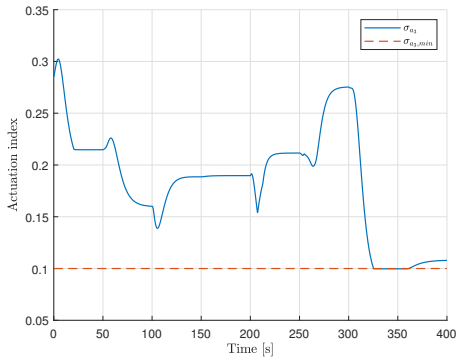
Table 7.9: Control parameters for the set-based operational space controllers.

k_{a_1}	k_{a_2}	k_{a_3}	λ_1	λ_{att}	λ_2	$k_{\boldsymbol{\zeta}}$
1.5	5	12	0.2	0.45	0.1	0.9

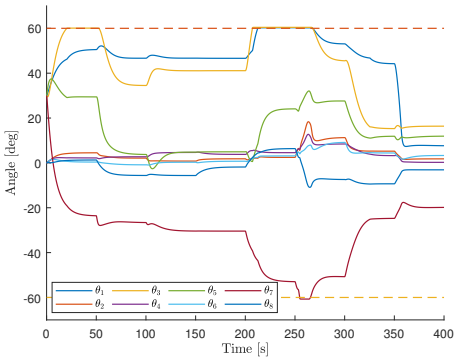
Simulation results are presented in Figure 7.7.



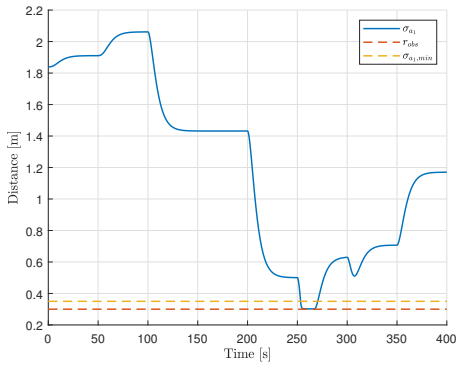
(a) North-West-Up plot. p_e, p_b and p_d represents the end-effector, base and the desired end-effector position, respectively.



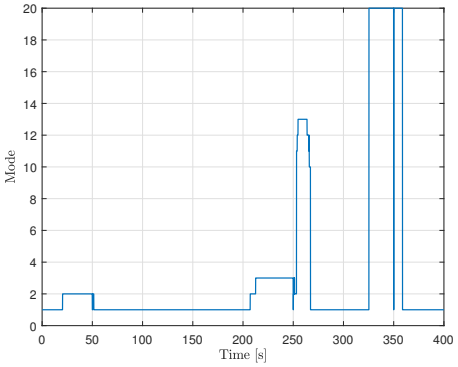
(b) The actuation index and its minimum value.



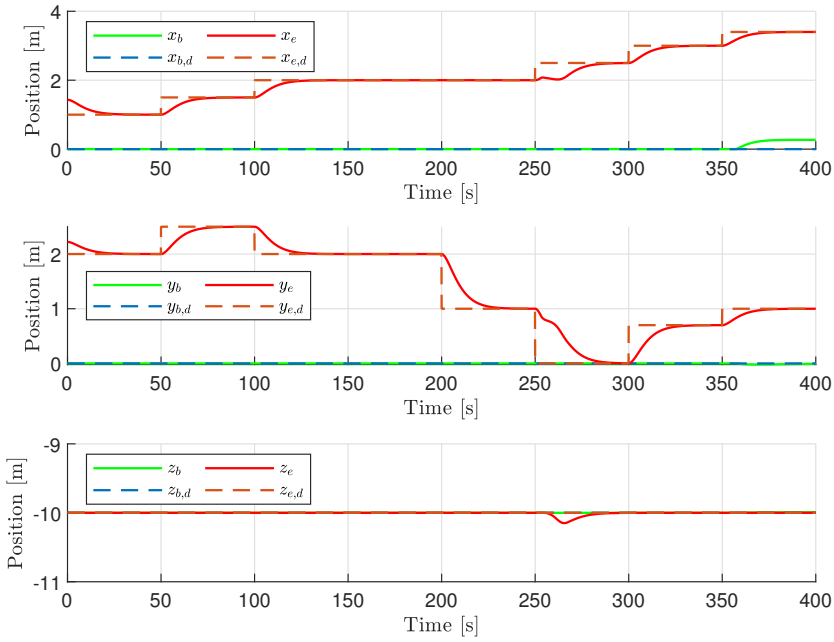
(c) Joint angles and their maximum and minimum values.



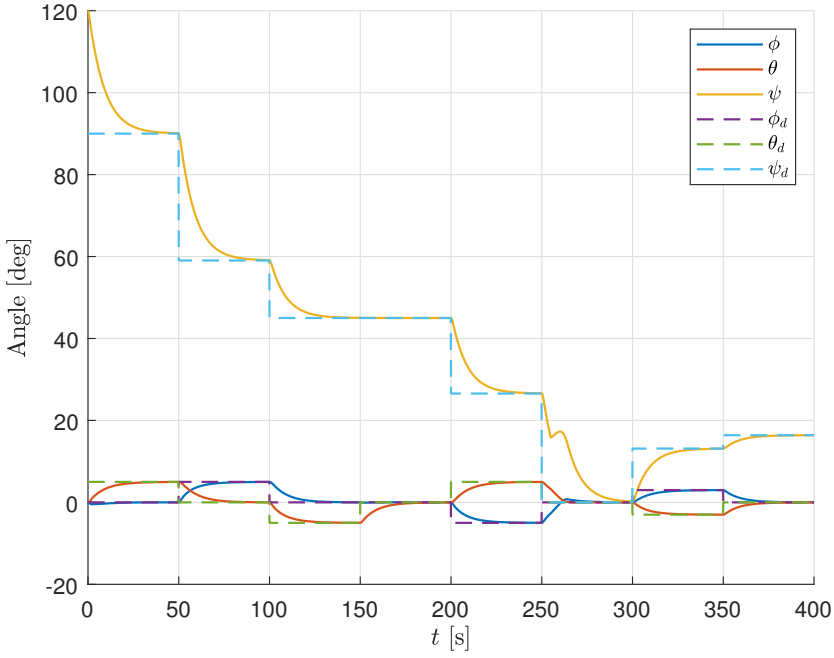
(d) Distance from the end-effector to the spherical obstacle.



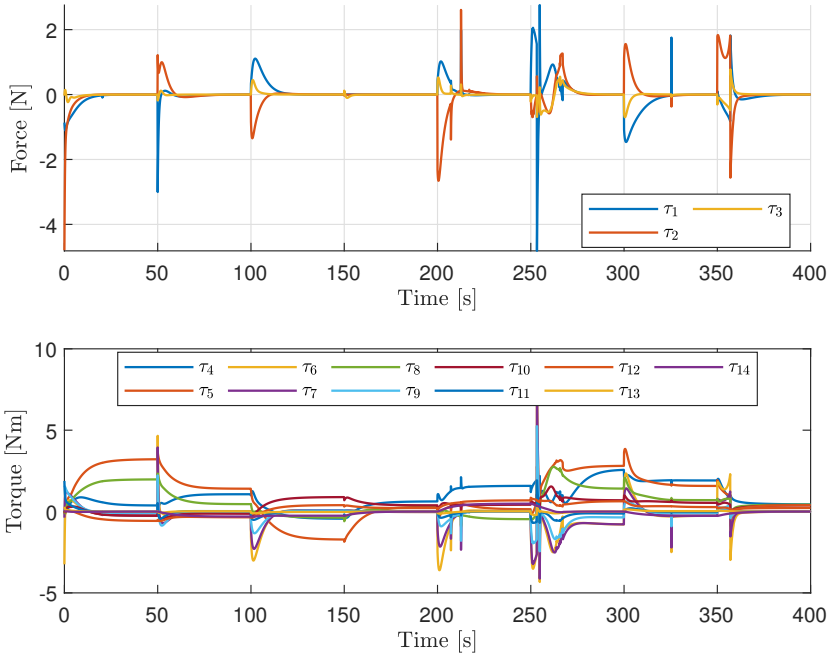
(e) The mode represents which set-based tasks are active at any given time.



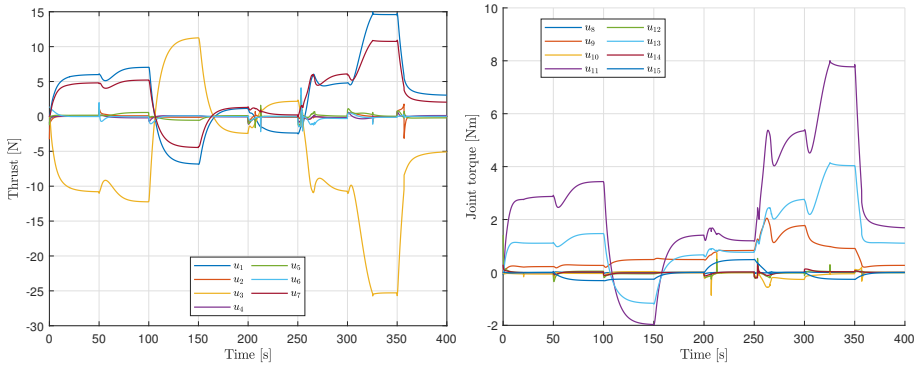
(f) End-effector, desired end-effector, base and desired base position.



(g) End-effector orientation.



(h) Commanded forces and torques.



(i) Thruster input generated by the control allocation algorithm.

(j) Joint torque inputs generated by the control allocation algorithm

Figure 7.7: Simulation results for the set-based operational space controller from Section 6.7.1

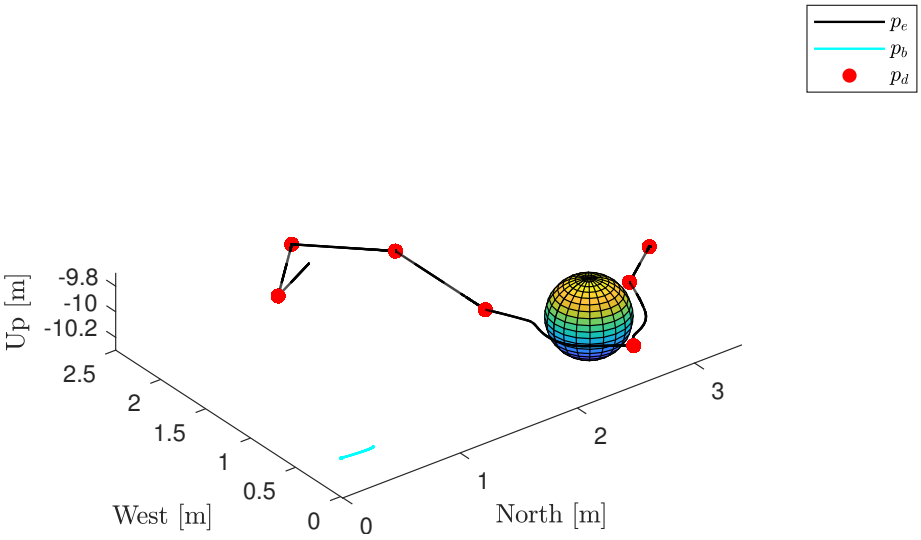
7.4.2.1 Smoothing the control torques

The equality task gains, λ_1, λ_{att}, λ_2 and $k_{\boldsymbol{\zeta}}$ remain the same as in the non-smoothed case, given by Table 7.9. The control parameters for the set-based tasks are modified, and a small proportional gain is added. The set-based control parameters are shown in Table 7.12.

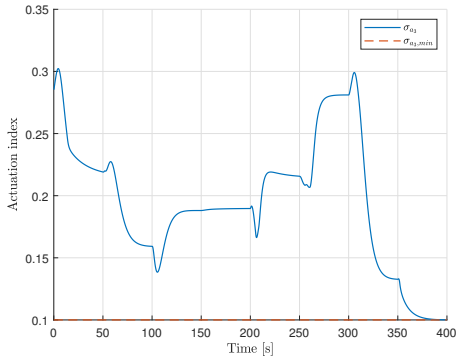
Table 7.10: Proportional and derivative gains for the augmented set-based task.

	σ_{a_1}	$\boldsymbol{\sigma}_{a_2}$	σ_{a_3}
γ	0.0	0.05	0.1
k	1.4	1	1

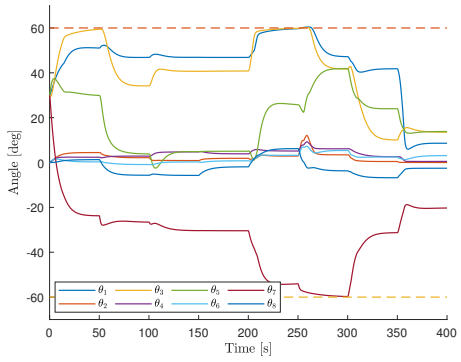
Simulation results are shown in Figure 7.8.



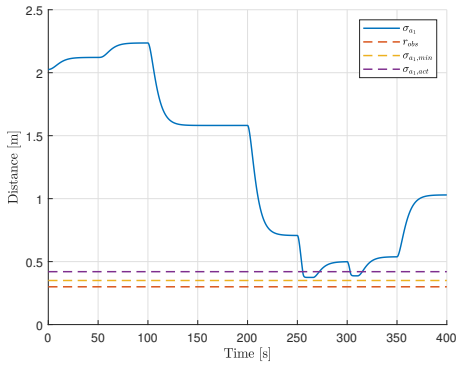
(a) North-West-Up plot. p_e, p_b and p_d represents the end-effector, base and the desired end-effector position, respectively.



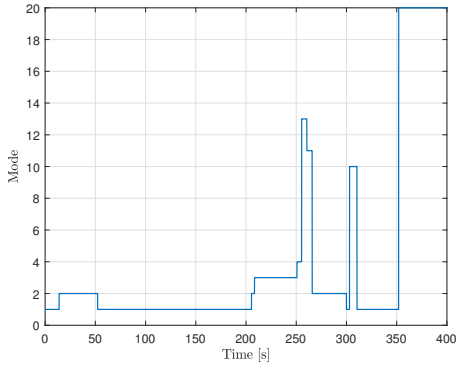
(b) The actuation index and its minimum value.



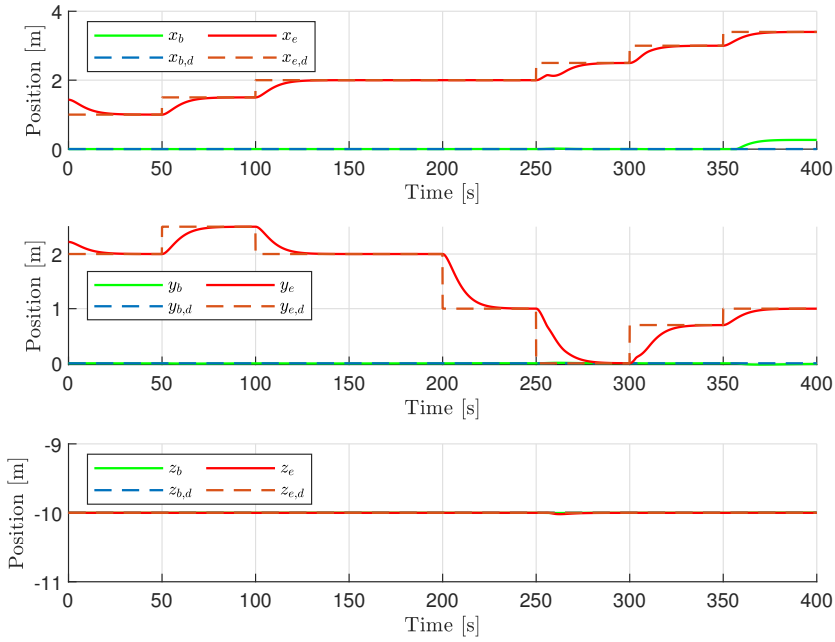
(c) Joint angles and their maximum and minimum values.



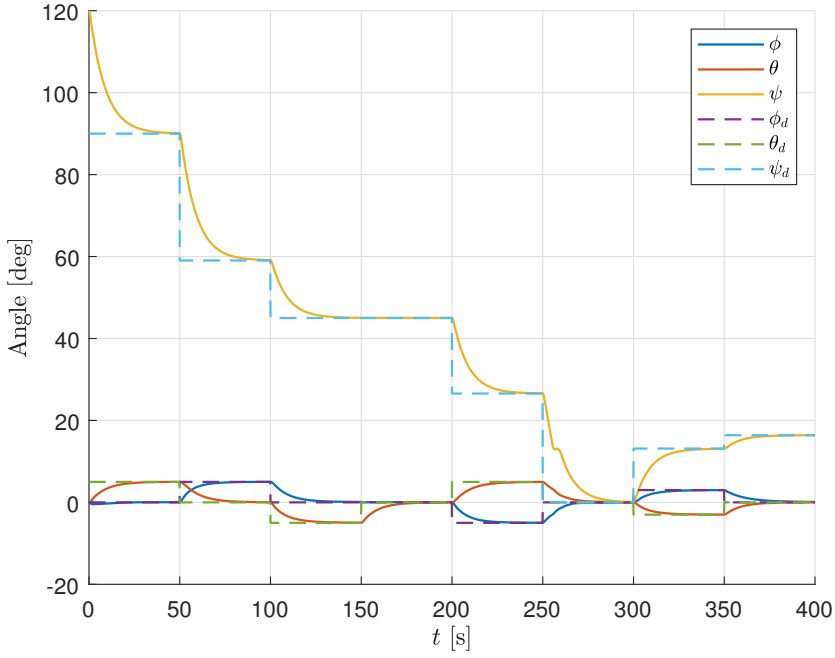
(d) Distance from the end-effector to the spherical obstacle.



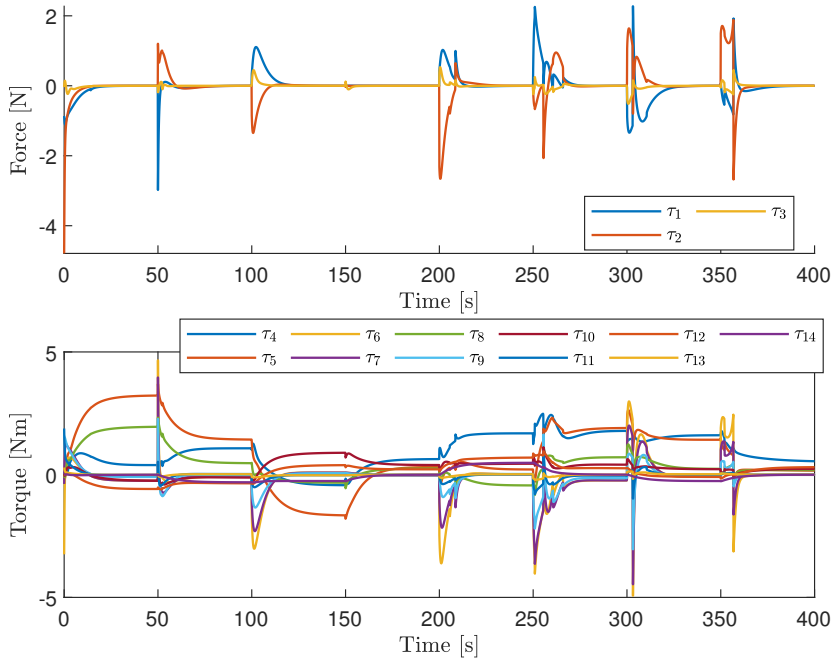
(e) The mode represents which set-based tasks are active at any given time.



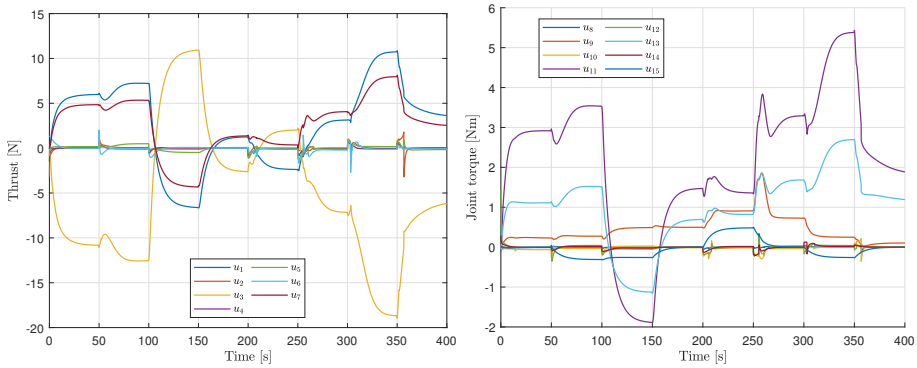
(f) End-effector, desired end-effector, base and desired base position.



(g) End-effector orientation.



(h) Commanded forces and torques.



(i) Thruster input generated by the control allocation algorithm.

(j) Joint torque inputs generated by the control allocation algorithm

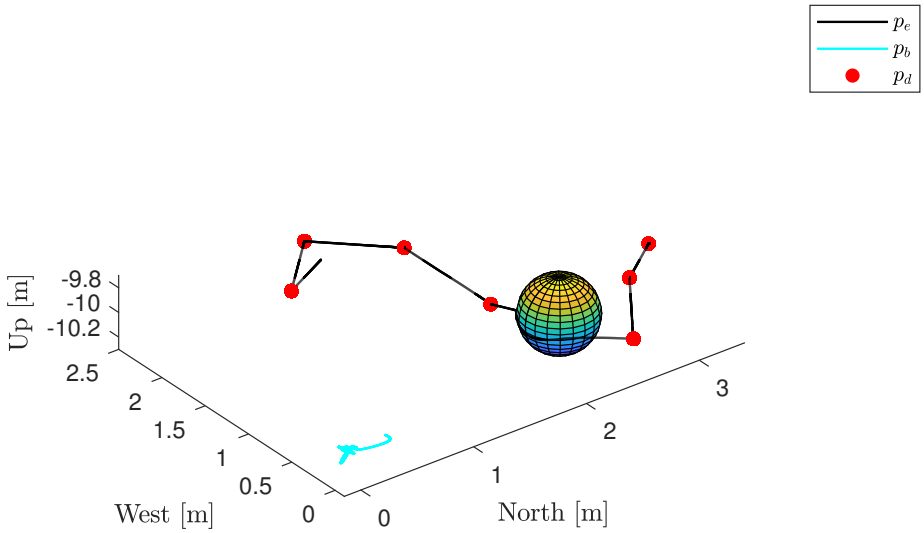
Figure 7.8: Simulation results for the set-based operational space controller from Section 6.7.1 with control torque smoothing.

7.4.3 Method 2: No compensation of higher priority tasks

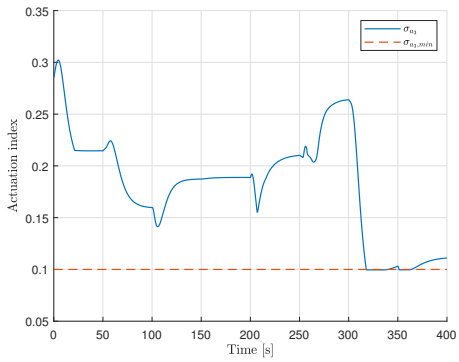
The control parameters for this approach are given in Table 7.11. Simulation results are presented in Figure 7.9.

Table 7.11: Control parameters for the set-based operational space controllers.

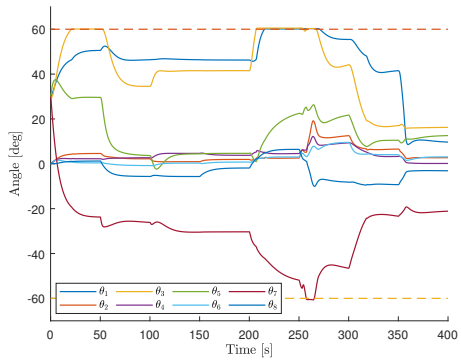
k_{a_1}	k_{a_2}	k_{a_3}	λ_1	λ_{att}	λ_2	$k_{\boldsymbol{\zeta}}$
1.5	5	12	0.2	0.45	0.1	0.9



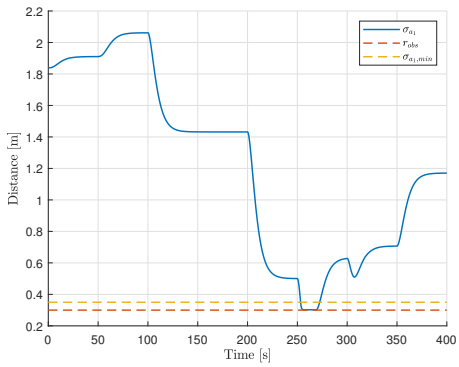
(a) North-West-Up plot. p_e, p_b and p_d represents the end-effector, base and the desired end-effector position, respectively.



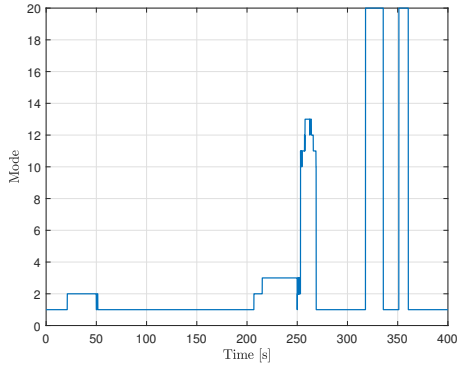
(b) The actuation index and its minimum value.



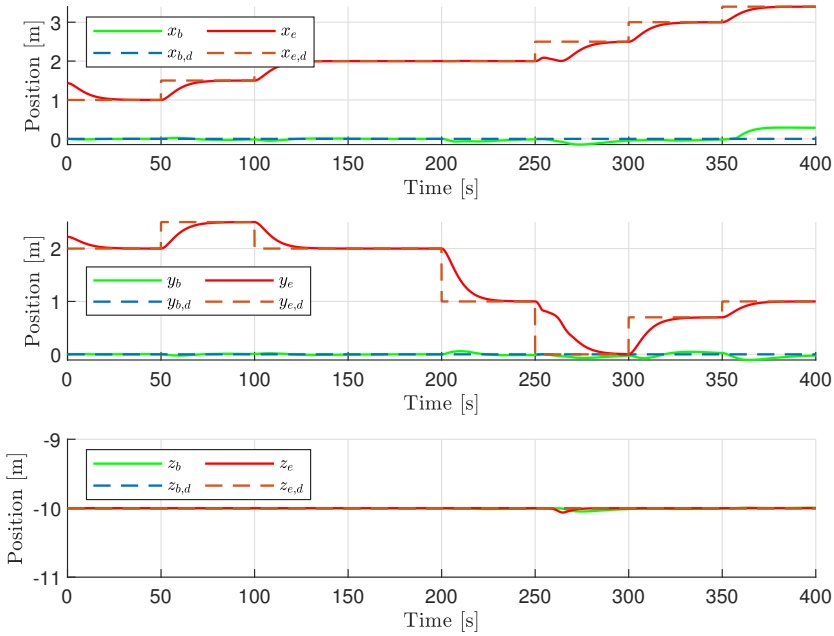
(c) Joint angles and their maximum and minimum values.



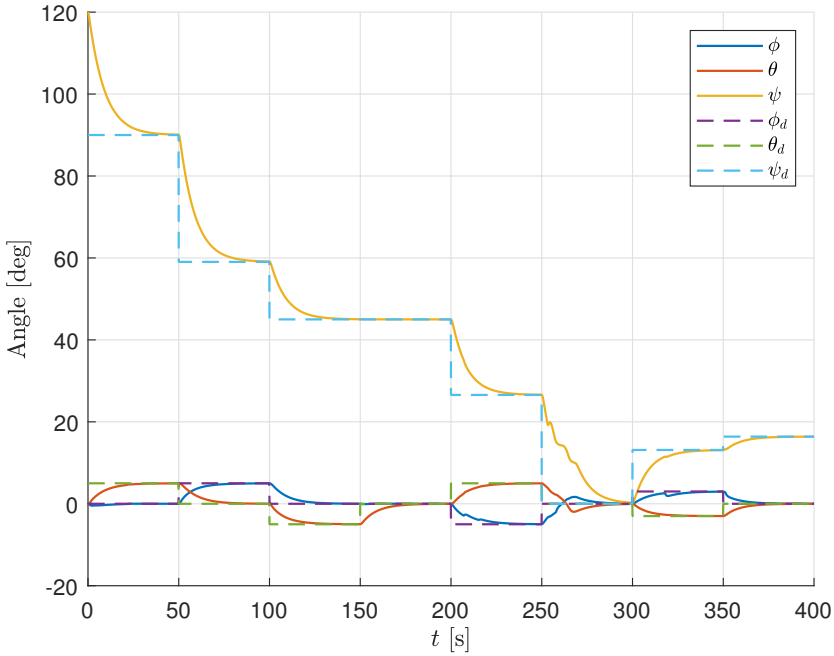
(d) Distance from the end-effector to the spherical obstacle.



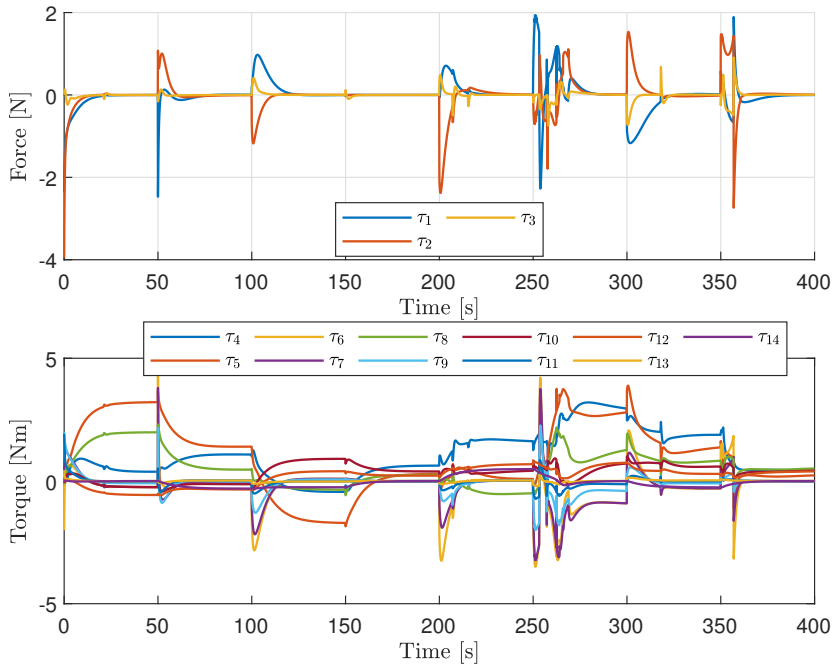
(e) The mode represents which set-based tasks are active at any given time.



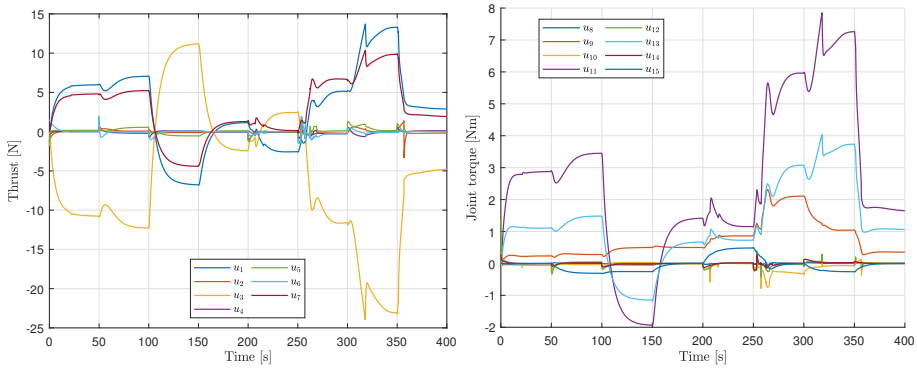
(f) End-effector, desired end-effector, base and desired base position.



(g) End-effector orientation.



(h) Commanded forces and torques.



(i) Thruster input generated by the control allocation algorithm.

(j) Joint torque inputs generated by the control allocation algorithm

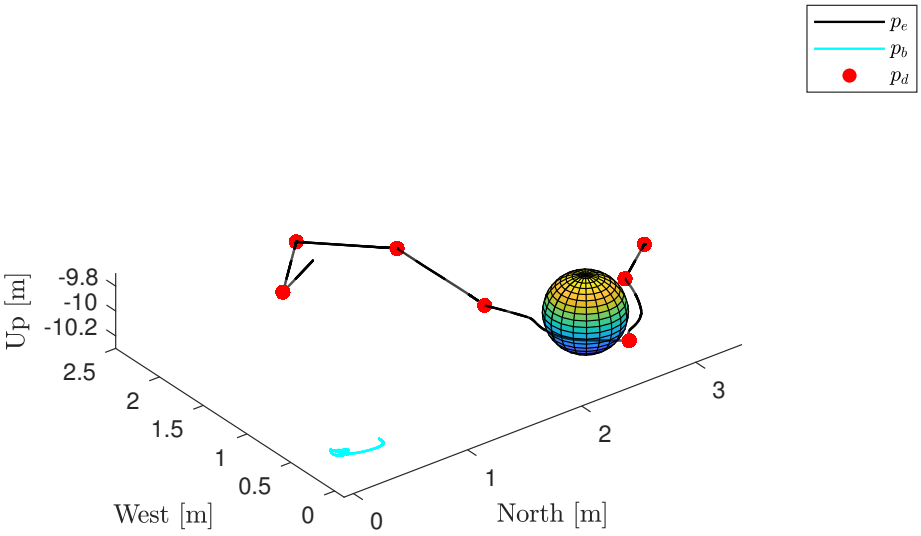
Figure 7.9: Simulation results for the set-based operational space controller from Section 6.7.2.

7.4.3.1 Smoothing the control torques

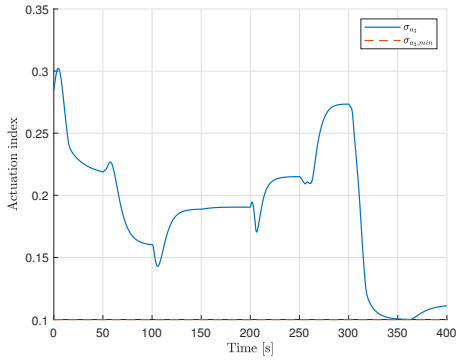
The equality task gains, λ_1, λ_{att}, λ_2 and $k_{\boldsymbol{\zeta}}$ remain the same as in the non-smoothed case, given by Table 7.9. The control parameters for the set-based tasks are modified, and a small proportional gain is added. The set-based control parameters are shown in Table 7.12. Simulation results are presented in Figure 7.10.

Table 7.12: Proportional and derivative gains for the augmented set-based task.

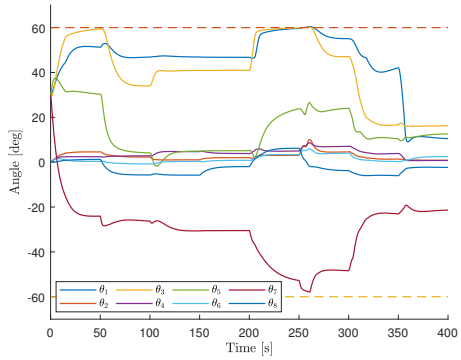
	σ_{a_1}	$\boldsymbol{\sigma}_{a_2}$	σ_{a_3}
γ	0.0	0.05	0.1
k	1.4	1	1



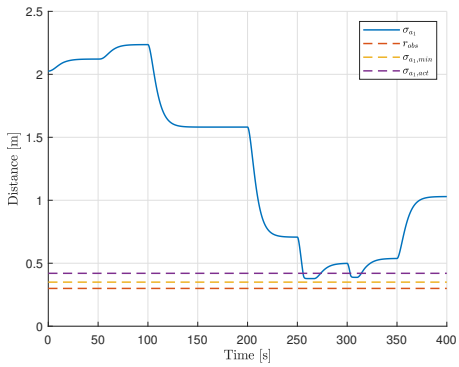
(a) North-West-Up plot. p_e, p_b and p_d represents the end-effector, base and the desired end-effector position, respectively.



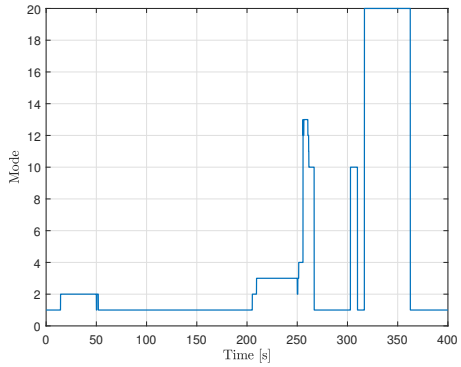
(b) The actuation index and its minimum value.



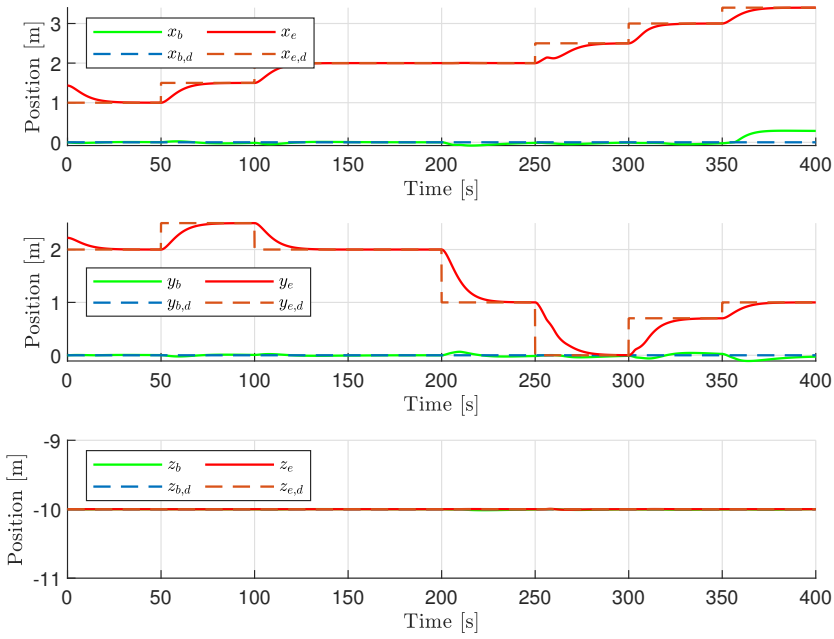
(c) Joint angles and their maximum and minimum values.



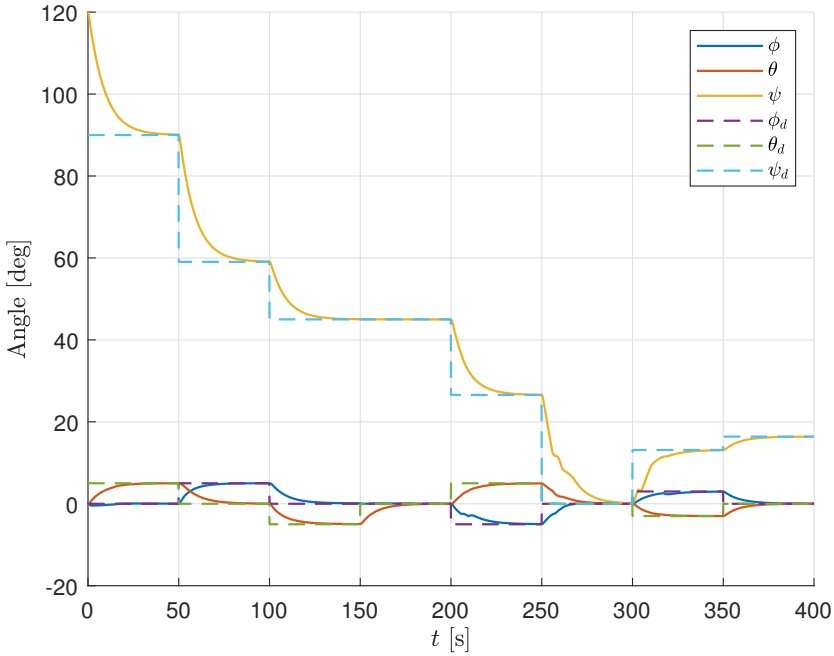
(d) Distance from the end-effector to the spherical obstacle.



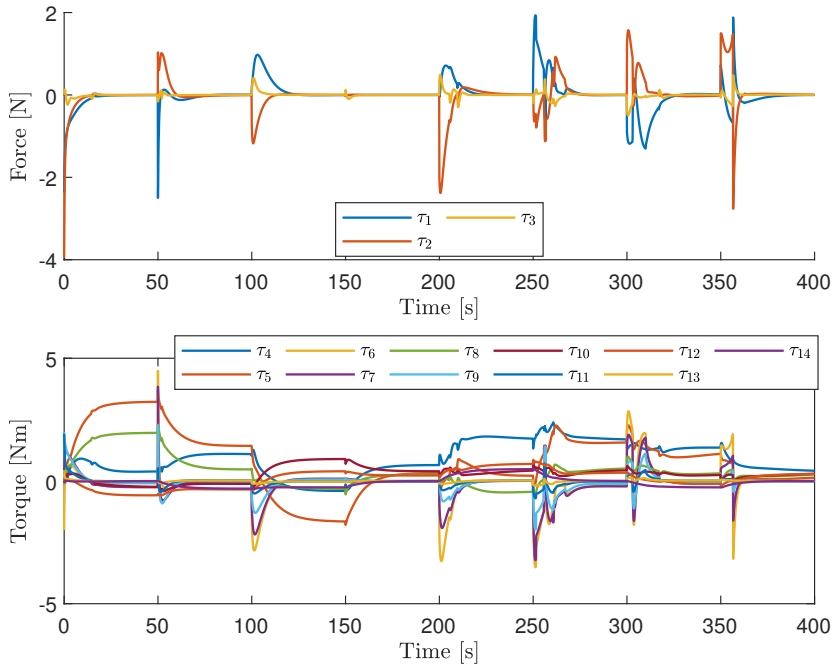
(e) The mode represents which set-based tasks are active at any given time.



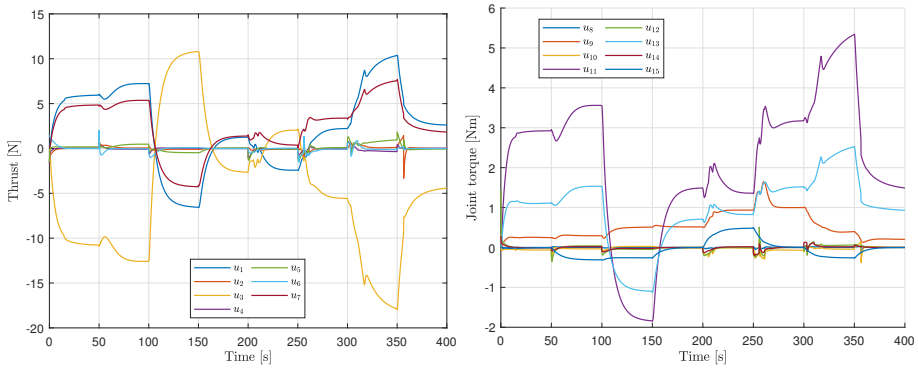
(f) End-effector, desired end-effector, base and desired base position.



(g) End-effector orientation.



(h) Commanded forces and torques.



(i) Thruster input generated by the control allocation algorithm.

(j) Joint torque inputs generated by the control allocation algorithm

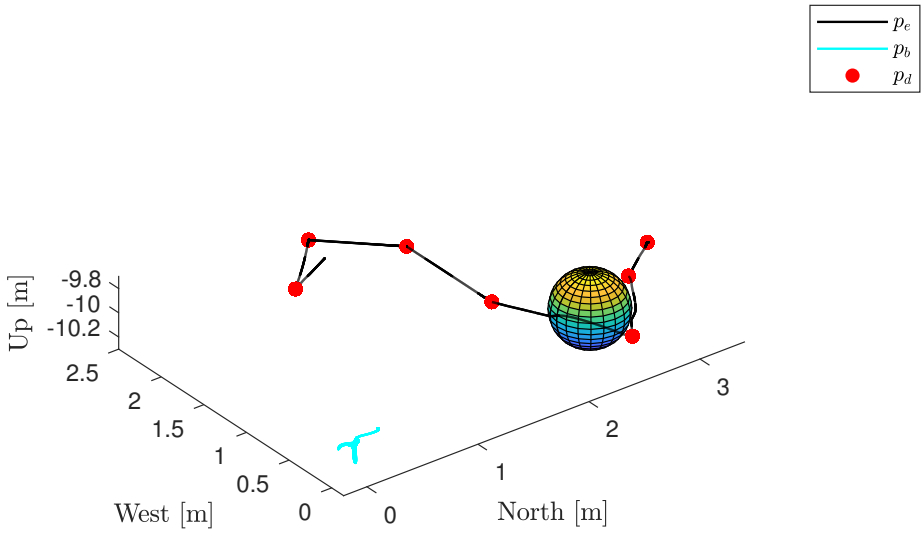
Figure 7.10: Simulation results for the set-based operational space controller from Section 6.7.2 with control torque smoothing.

7.4.4 Method 3: Omitting the null space operator within the task specific inertia matrix

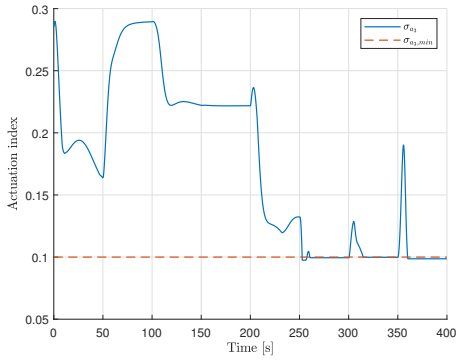
The control parameters for this approach are given in Table 7.11. Simulation results are presented in Figure 7.9.

Table 7.13: Control parameters for the set-based operational space controllers.

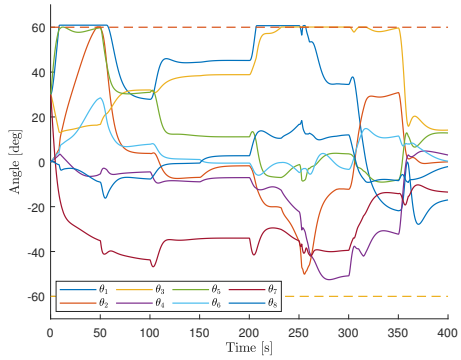
k_{a_1}	k_{a_2}	k_{a_3}	λ_1	λ_{att}	λ_2	$k_{\boldsymbol{\zeta}}$
1.5	5	12	0.2	0.5	0.15	0.9



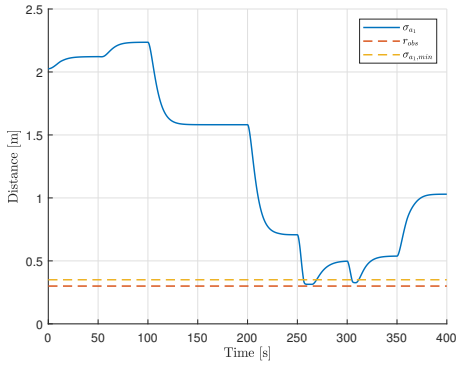
(a) North-West-Up plot. p_e, p_b and p_d represents the end-effector, base and the desired end-effector position, respectively.



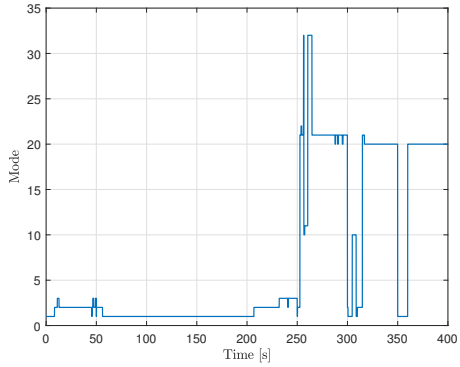
(b) The actuation index and its minimum value.



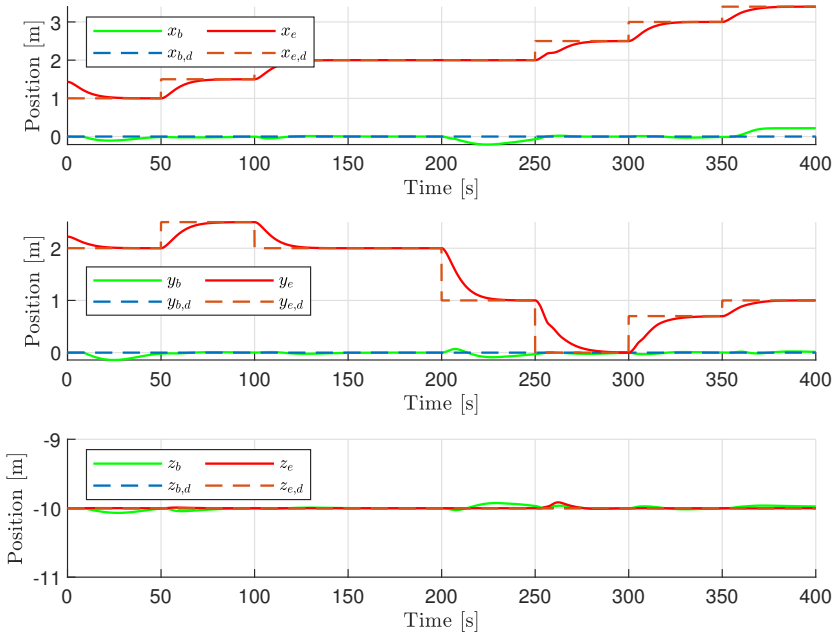
(c) Joint angles and their maximum and minimum values.



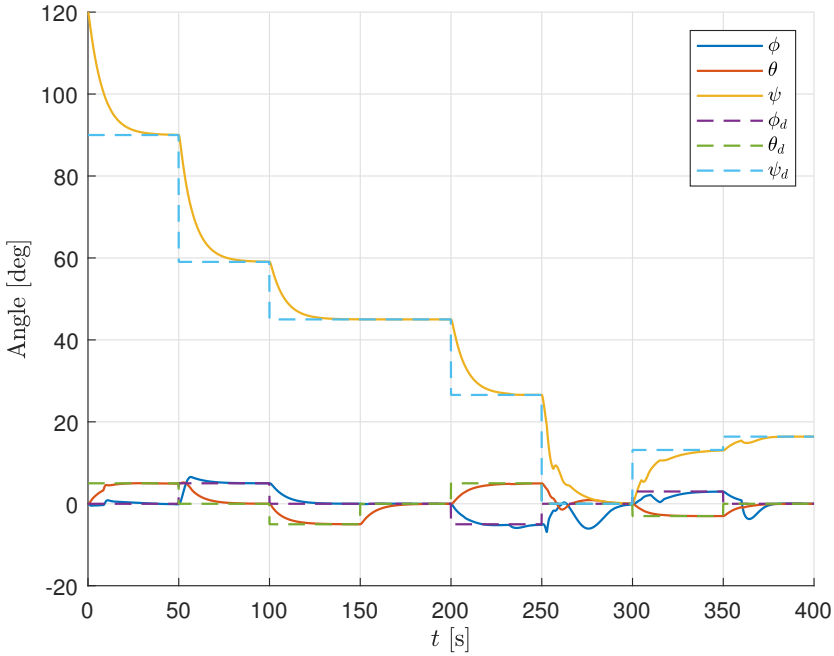
(d) Distance from the end-effector to the spherical obstacle.



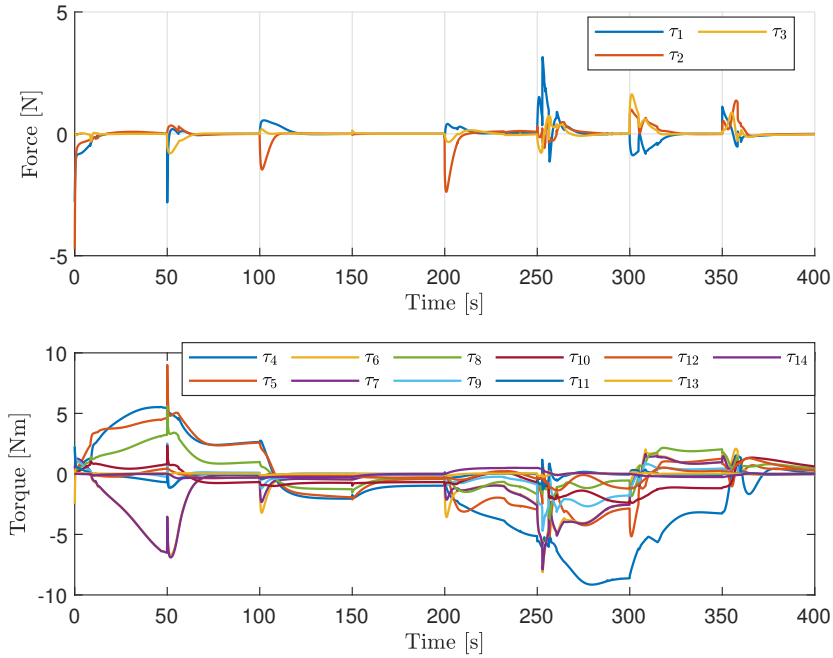
(e) The mode represents which set-based tasks are active at any given time.



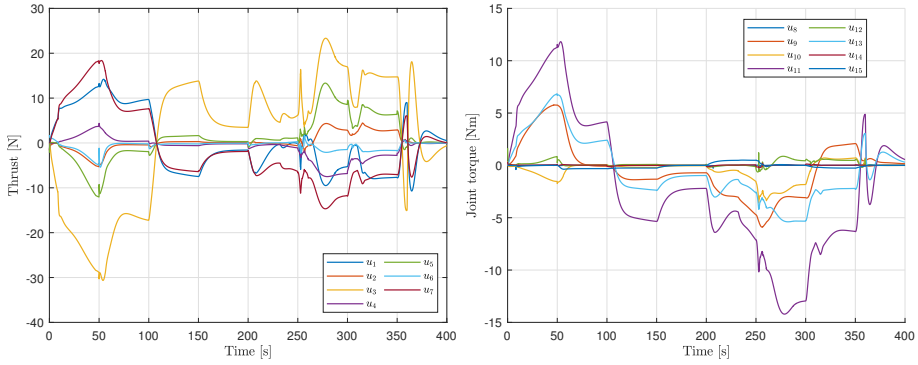
(f) End-effector, desired end-effector, base and desired base position.



(g) End-effector orientation.



(h) Commanded forces and torques.



(i) Thruster input generated by the control allocation algorithm.

(j) Joint torque inputs generated by the control allocation algorithm

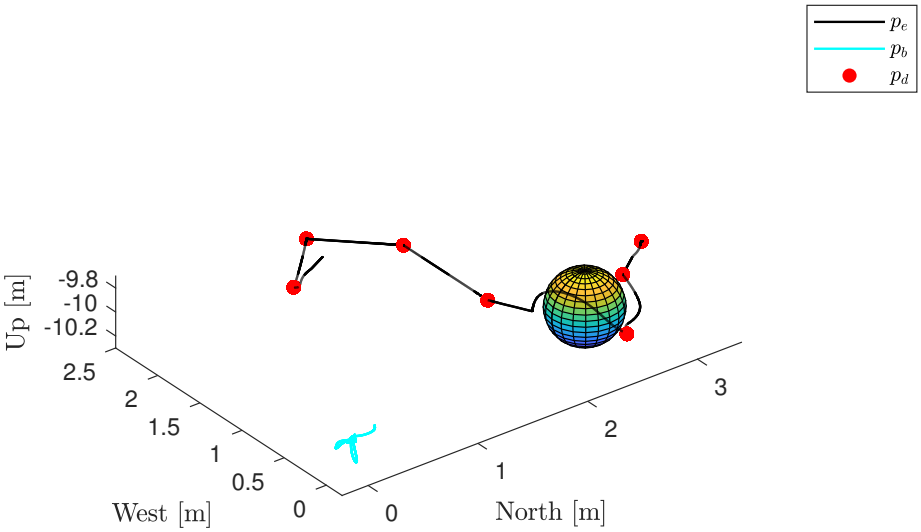
Figure 7.11: Simulation results for the set-based operational space controller from Section 6.7.3.

7.4.4.1 Smoothing the control torques

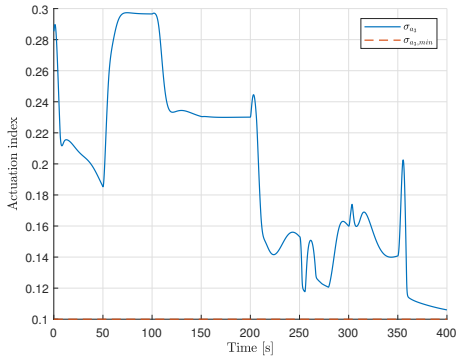
The equality task gains, λ_1, λ_{att}, λ_2 and $k_{\boldsymbol{\zeta}}$ remain the same as in the non-smoothed case, given by Table 7.13. The control parameters for the set-based tasks are modified, and a small proportional gain is added. The set-based control parameters are shown in Table 7.14. Simulation results are presented in Figure 7.12.

Table 7.14: Proportional and derivative gains for the augmented set-based task.

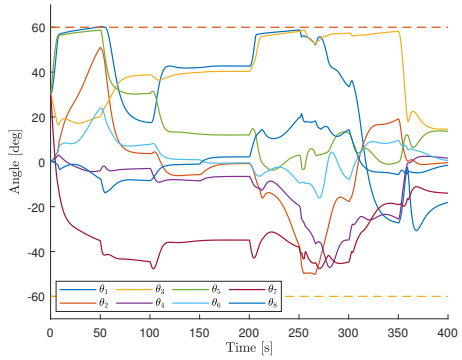
	σ_{a_1}	$\boldsymbol{\sigma}_{a_2}$	σ_{a_3}
γ	0.0	0.05	0.05
k	2.5	2.5	2.5



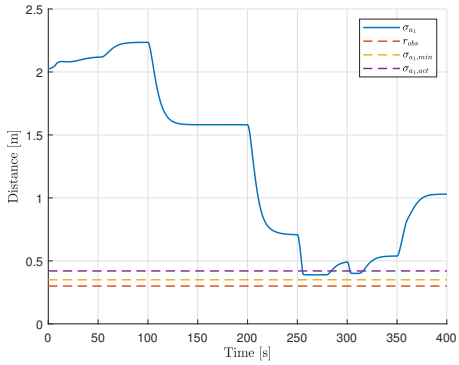
(a) North-West-Up plot. p_e, p_b and p_d represents the end-effector, base and the desired end-effector position, respectively.



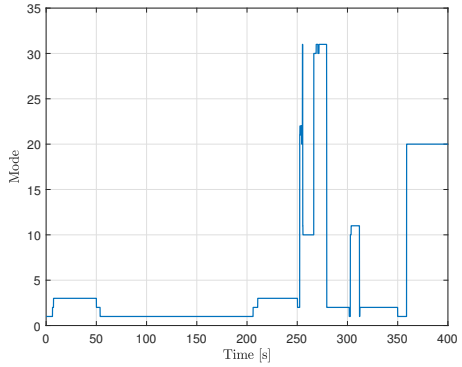
(b) The actuation index and its minimum value.



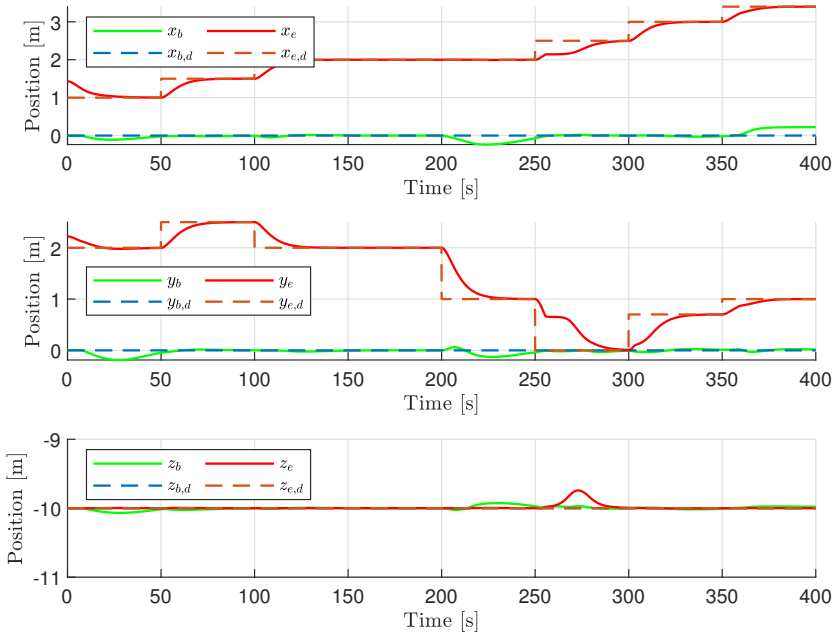
(c) Joint angles and their maximum and minimum values.



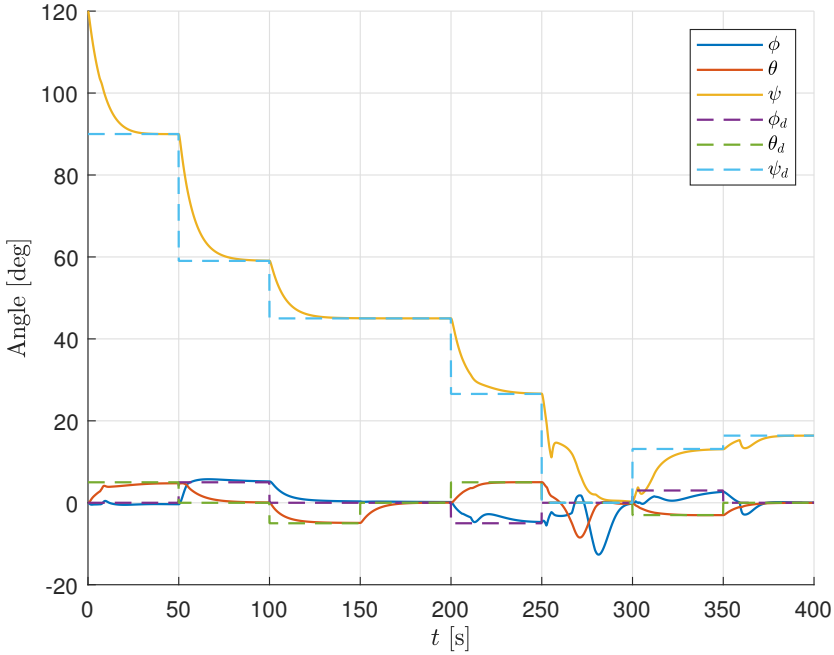
(d) Distance from the end-effector to the spherical obstacle.



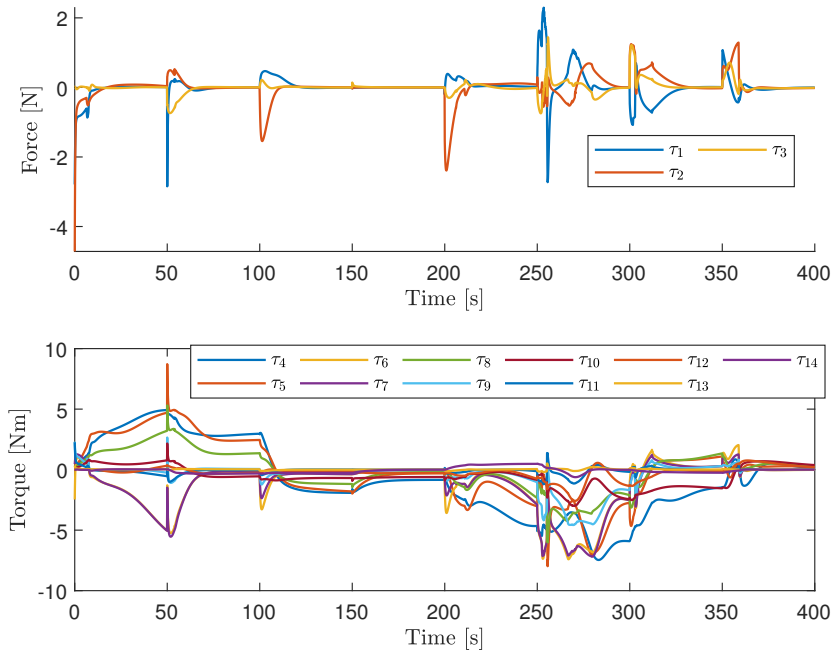
(e) The mode represents which set-based tasks are active at any given time.



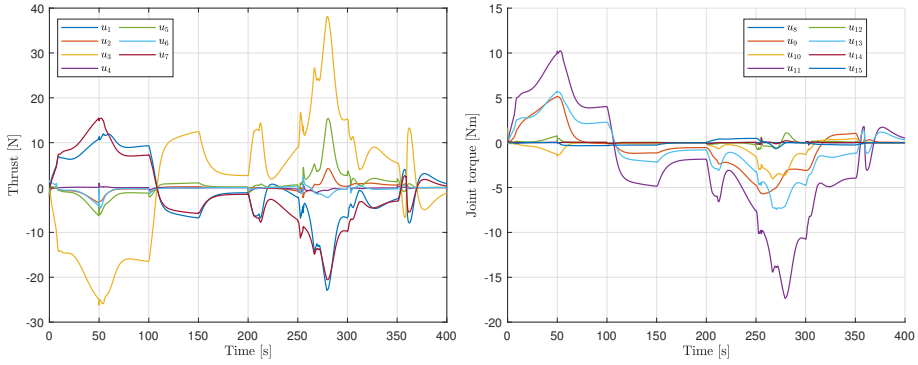
(f) End-effector, desired end-effector, base and desired base position.



(g) End-effector orientation.



(h) Commanded forces and torques.



(i) Thruster input generated by the control allocation algorithm.

(j) Joint torque inputs generated by the control allocation algorithm

Figure 7.12: Simulation results for the set-based operational space controller from Section 6.7.3 with control torque smoothing.

7.4.5 Uncertainty in the dynamic parameters

As discussed in Section 5.3, the inertia matrix consists of the rigid-body inertia matrix and the added mass matrix, viz.

$$\boldsymbol{M} = \boldsymbol{M}_{RB} + \boldsymbol{M}_{A}. \tag{7.18}$$

To investigate the effects of large uncertainties in the inertia matrix, only the rigid body inertia matrix is used in the following simulations. In order to illustrate how significant the added mass inertia is with respect to the total inertia matrix, the 2-norm of the inertia matrix, added mass matrix and rigid body inertia matrix is shown in Figure 7.13. The figure is obtained from the simulation in

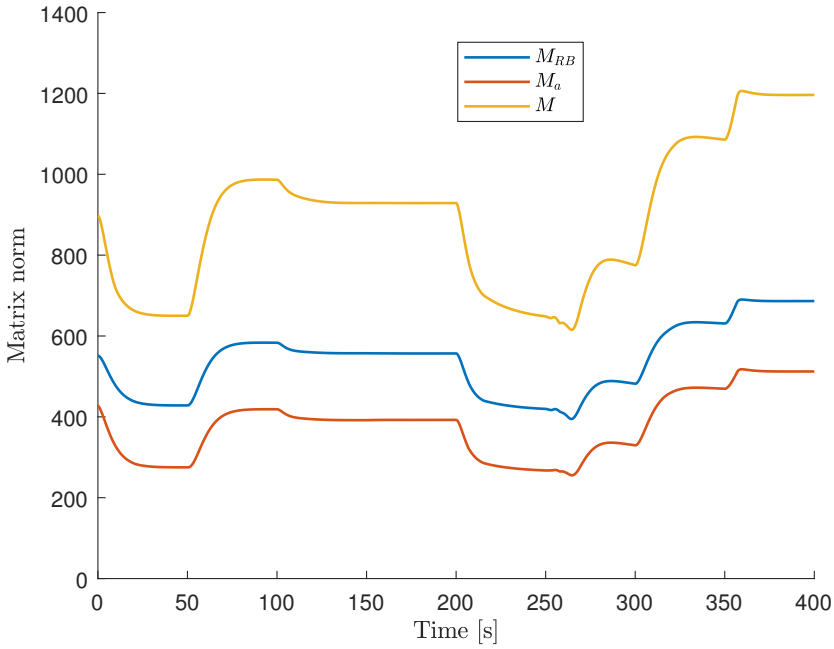
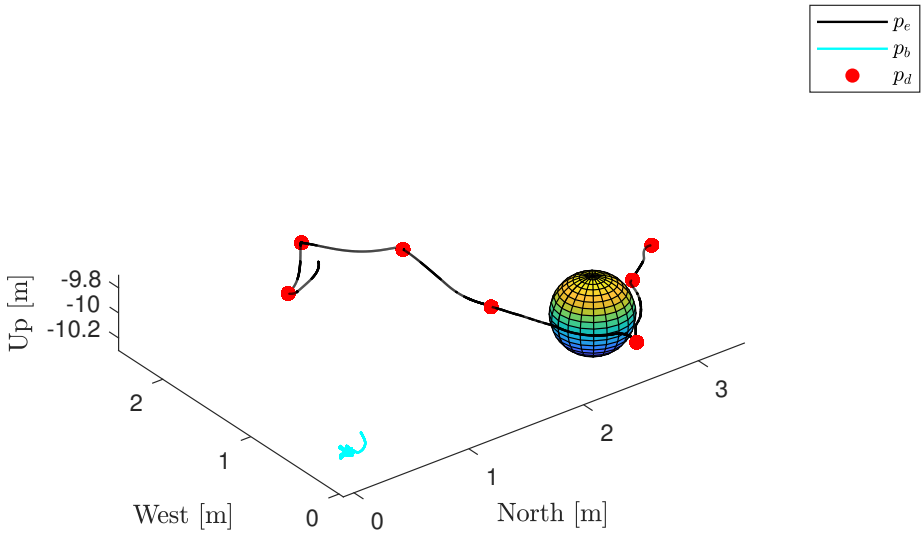


Figure 7.13: The 2-norm of the rigid body inertia matrix, the added mass matrix and the inertia matrix. The 2-norm of a matrix is equal to the largest singular value of the matrix.

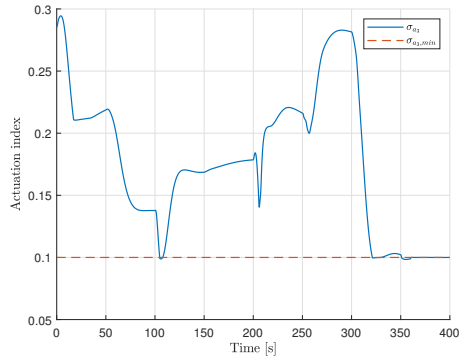
Section 7.4.2. From this distance measure, it is clear that the added mass inertia represents a considerable part of the total inertia.

7.4.5.1 Method 1

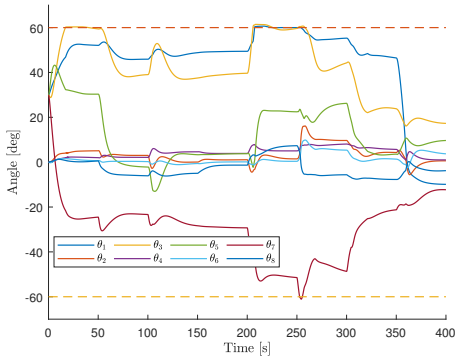
The control parameters are given by Table 7.11, with the exception of the derivative gain for the collision avoidance and joint limit avoidance tasks, which are changed to $k_{a_1} = 2$ and $k_{a_2} = 7$, respectively. Simulation results are presented in Figure 7.14.



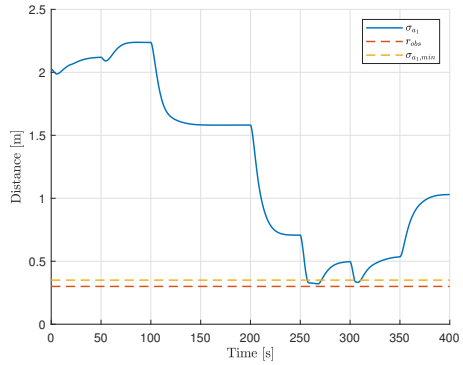
(a) North-West-Up plot. p_e, p_b and p_d represents the end-effector, base and the desired end-effector position, respectively.



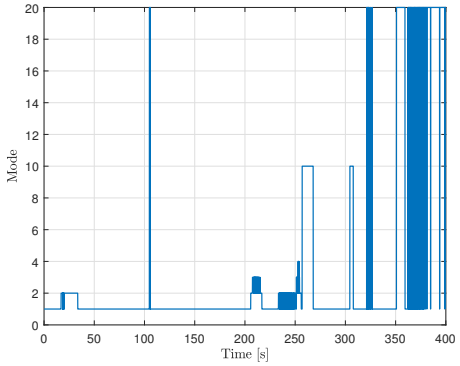
(b) The actuation index and its minimum value.



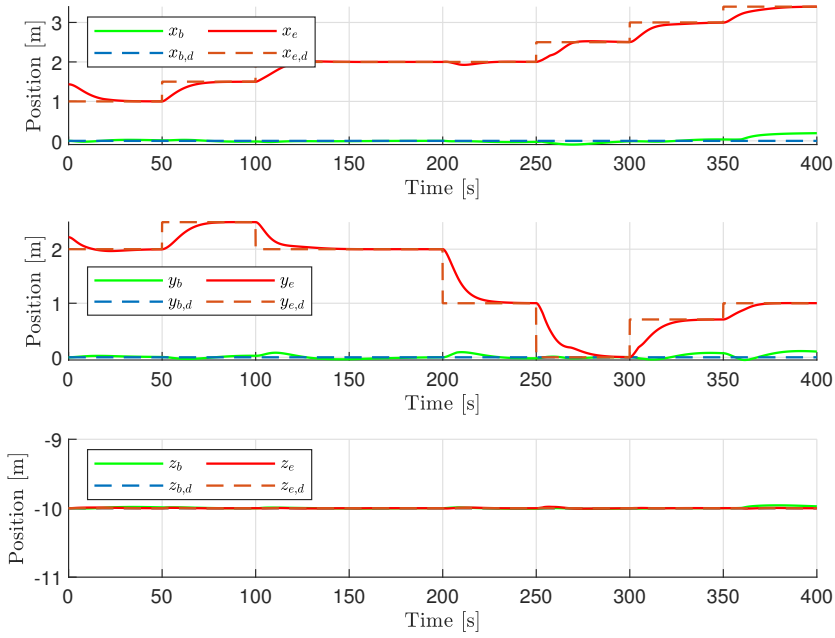
(c) Joint angles and their maximum and minimum values.



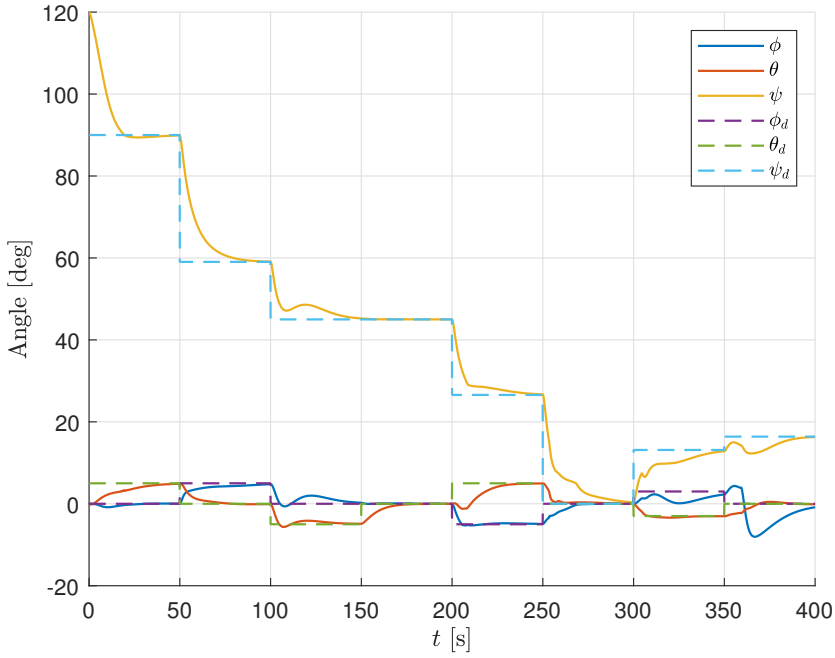
(d) Distance from the end-effector to the spherical obstacle.



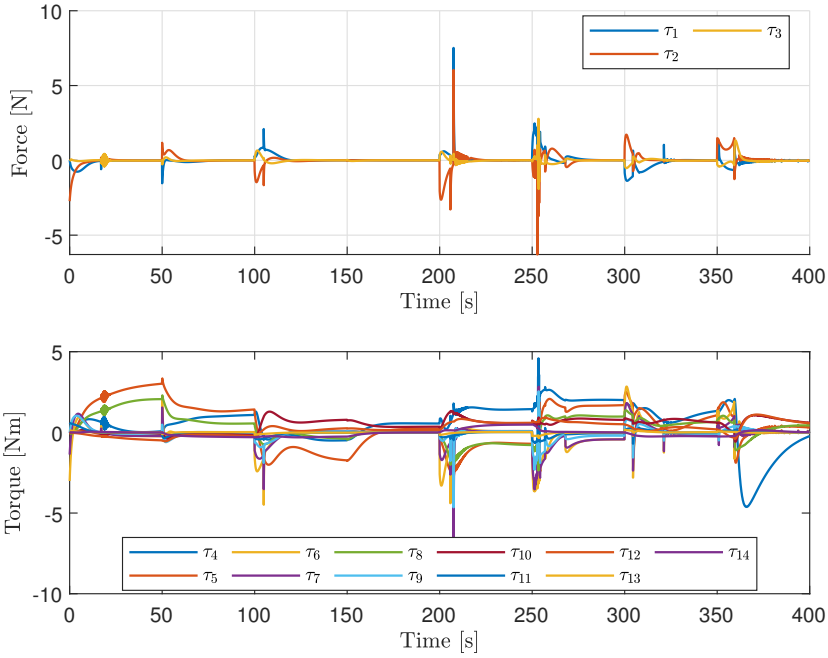
(e) The mode represents which set-based tasks are active at any given time.



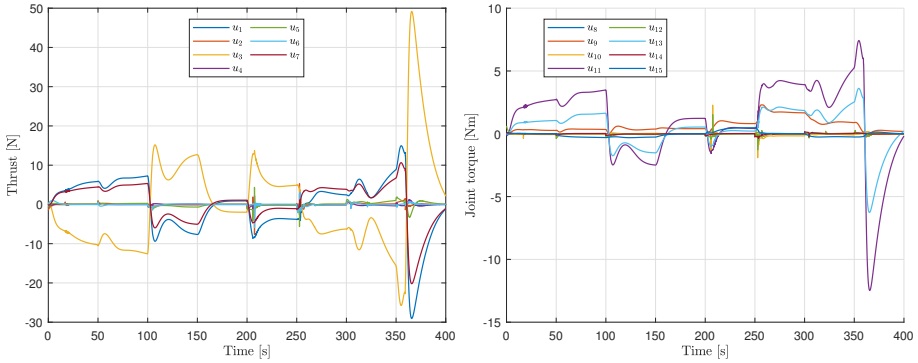
(f) End-effector, desired end-effector, base and desired base position.



(g) End-effector orientation.



(h) Commanded forces and torques.



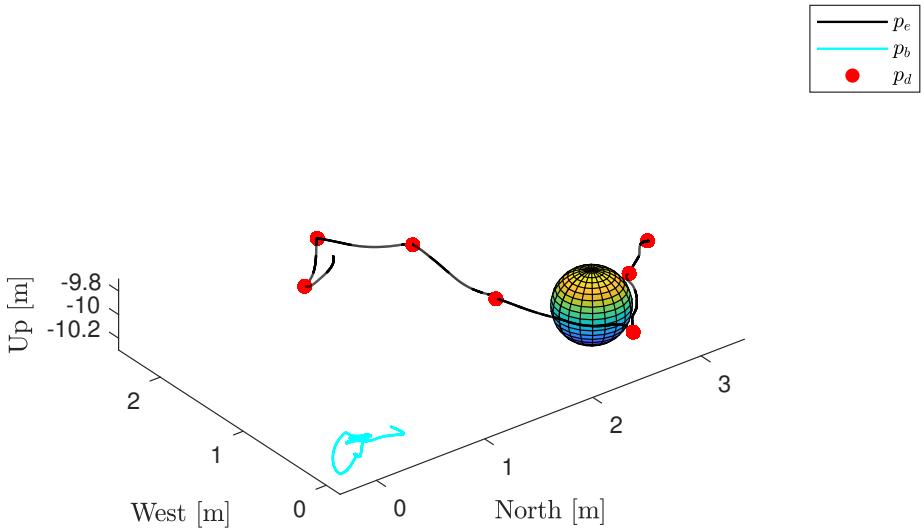
(i) Thruster input generated by the control allocation algorithm.

(j) Joint torque inputs generated by the control allocation algorithm

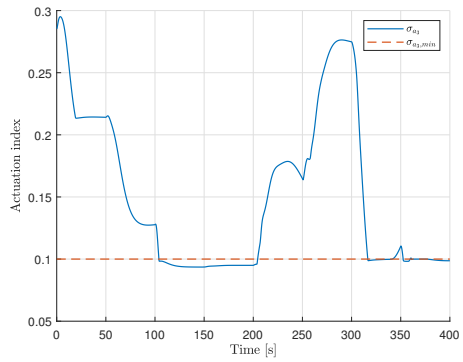
Figure 7.14: Simulation results for the set-based operational space controller from Section 6.7.1 where all added mass terms have been omitted from the inertia matrix.

7.4.5.2 Method 2

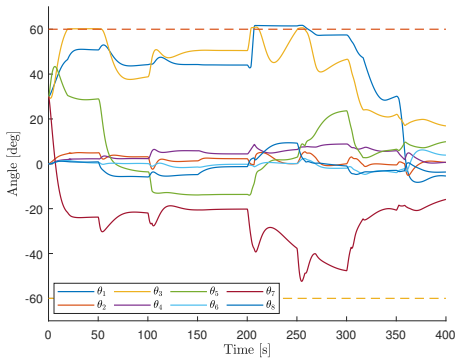
The control parameters are given by Table 7.11, with the exception of the derivative gain for the collision avoidance and joint limit avoidance tasks, which are changed to $k_{a_1} = 2$ and $k_{a_2} = 7$, respectively. Simulation results are presented in Figure 7.15.



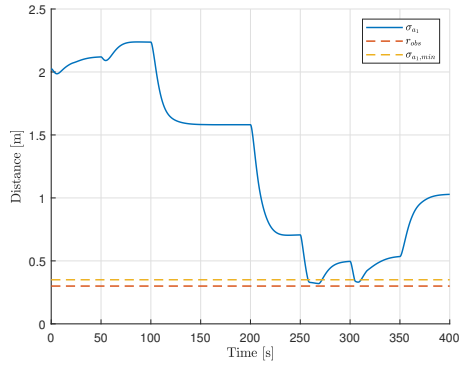
(a) North-West-Up plot. p_e, p_b and p_d represents the end-effector, base and the desired end-effector position, respectively.



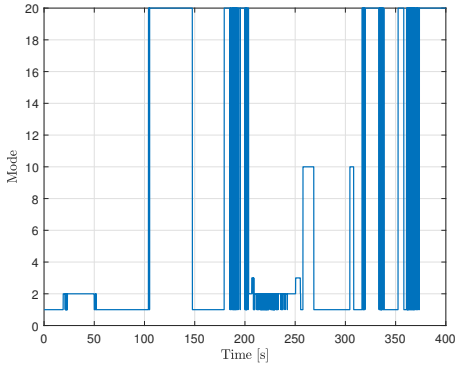
(b) The actuation index and its minimum value.



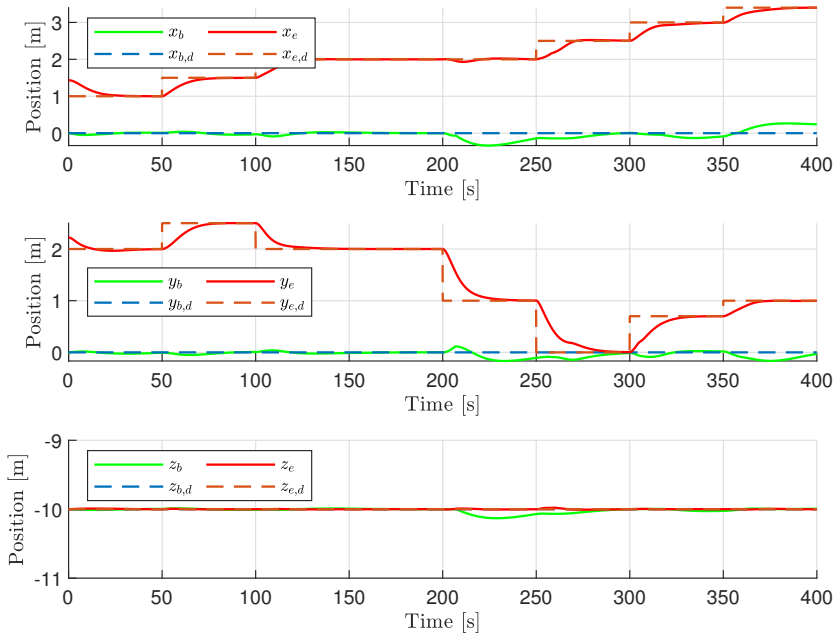
(c) Joint angles and their maximum and minimum values.



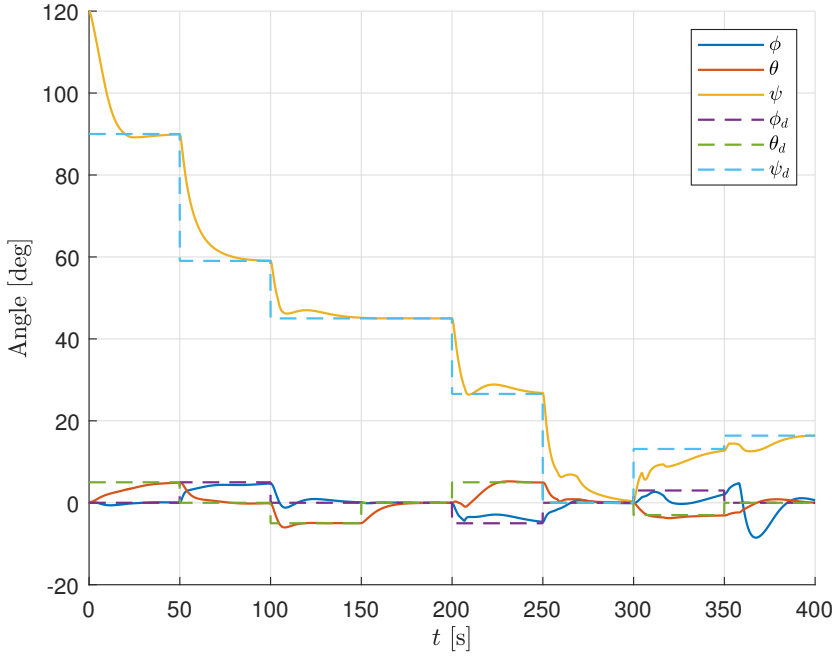
(d) Distance from the end-effector to the spherical obstacle.



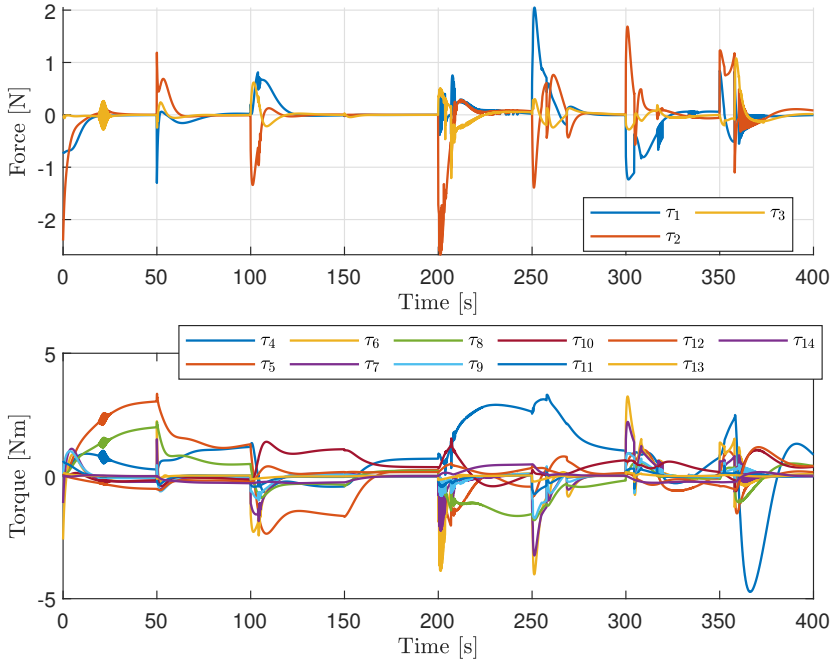
(e) The mode represents which set-based tasks are active at any given time.



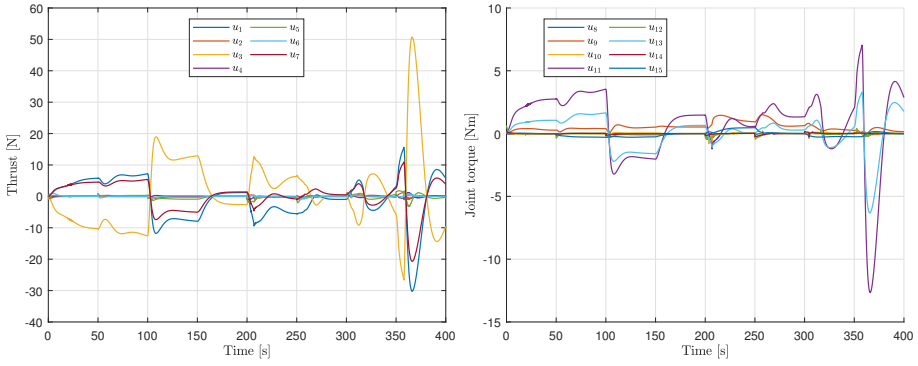
(f) End-effector, desired end-effector, base and desired base position.



(g) End-effector orientation.



(h) Commanded forces and torques.



(i) Thruster input generated by the control allocation algorithm.

(j) Joint torque inputs generated by the control allocation algorithm

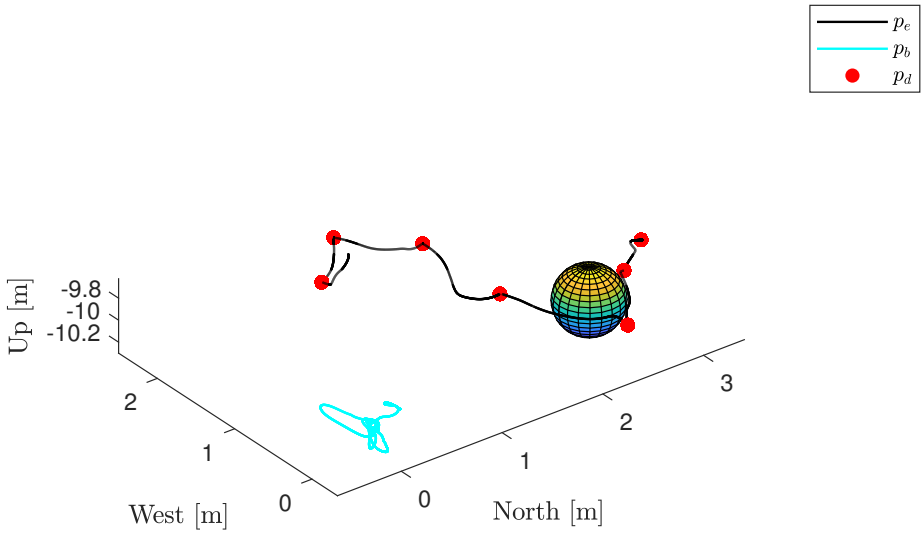
Figure 7.15: Simulation results for the set-based operational space controller from Section 6.7.2 where all added mass terms have been omitted from the inertia matrix.

7.4.5.3 Method 3

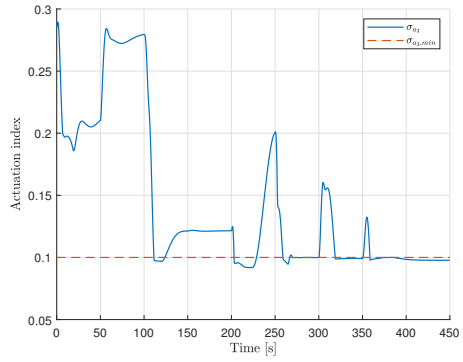
The control parameters are given by Table 7.15. The equality task gains had to be slightly increased in order to reach the desired end-effector positions within 50 seconds. Simulations results are presented in Figure 7.16

Table 7.15: Control parameters for a set-based operational space controller.

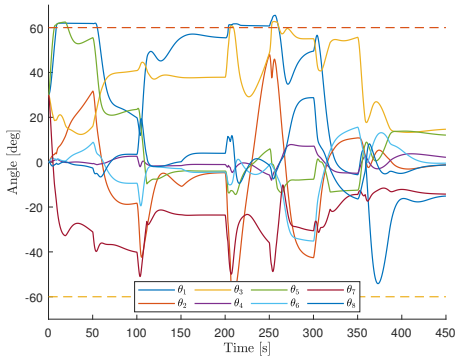
k_{a_1}	k_{a_2}	k_{a_3}	λ_1	λ_{att}	λ_2	$k_{\boldsymbol{\zeta}}$
1.5	5	12	0.25	0.5	0.15	1



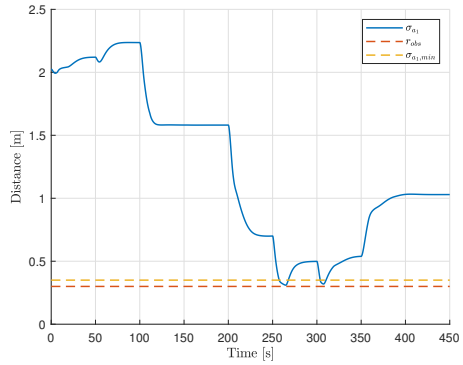
(a) North-West-Up plot. p_e, p_b and p_d represents the end-effector, base and the desired end-effector position, respectively.



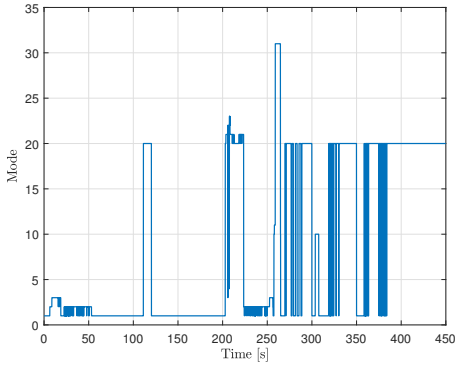
(b) The actuation index and its minimum value.



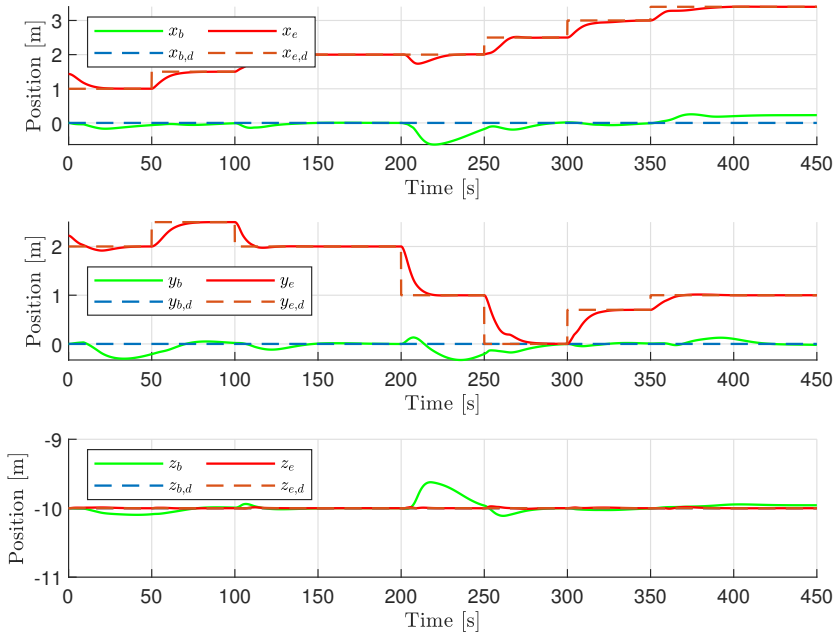
(c) Joint angles and their maximum and minimum values.



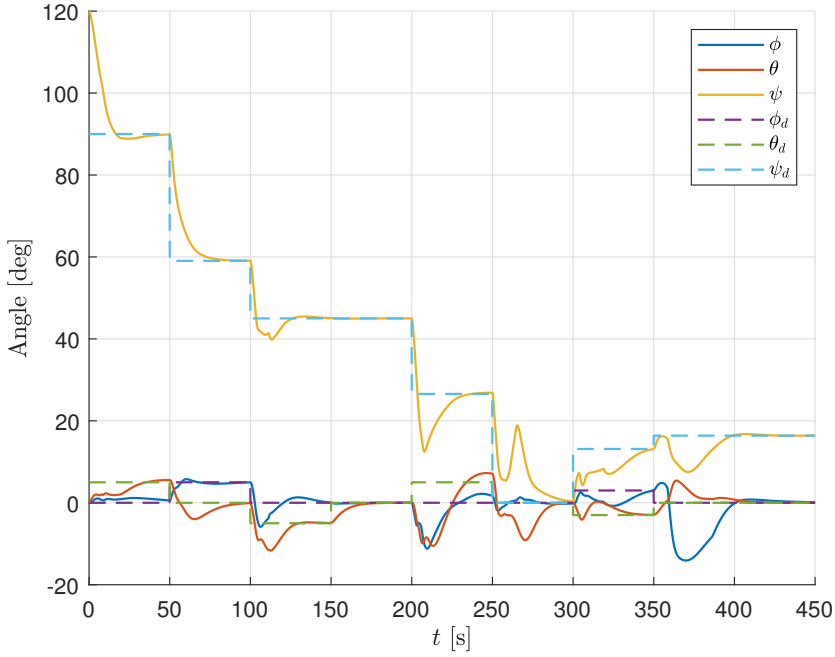
(d) Distance from the end-effector to the spherical obstacle.



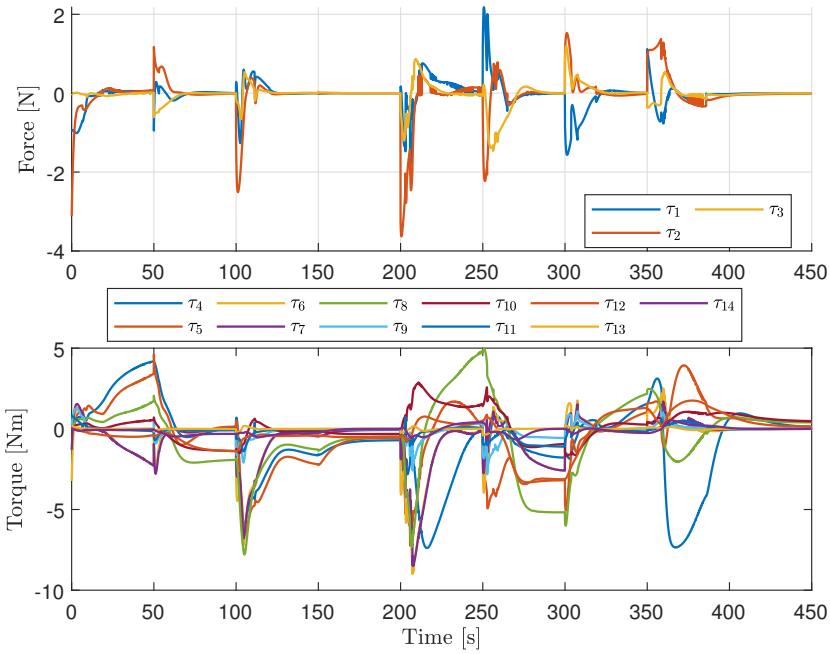
(e) The mode represents which set-based tasks are active at any given time.



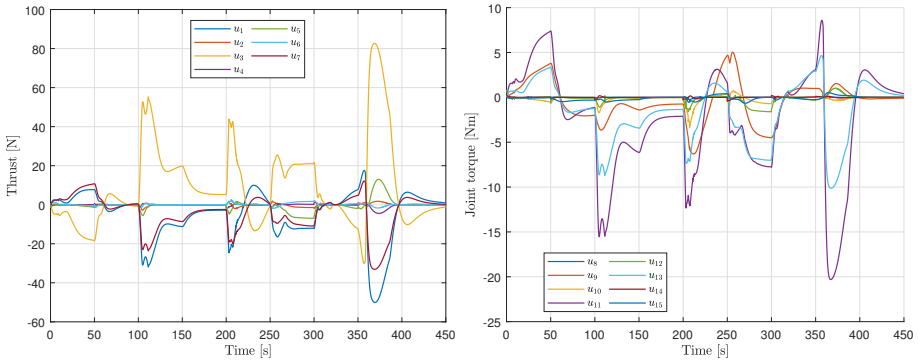
(f) End-effector, desired end-effector, base and desired base position.



(g) End-effector orientation.



(h) Commanded forces and torques.



(i) Thruster input generated by the control allocation algorithm.

(j) Joint torque inputs generated by the control allocation algorithm

Figure 7.16: Simulation results for the set-based operational space controller from Section 6.7.3 where all added mass terms have been omitted from the inertia matrix.

Chapter 8

Discussion

8.1 Kinematic Control

8.1.1 Velocity level redundancy resolution

Since the end-effector configuration set-points change every 50 seconds, the control inputs and velocity references have somewhat large jumps whenever this happens, see Figure 7.1, 7.2 and 7.3. This can be improved by employing a second order reference model which takes as input the end-effector configuration reference, and provides desired end-effector position, attitude and linear and angular velocities as outputs to be used by the kinematic controller.

8.1.1.1 Equality tasks

From Figure 7.1 and 7.2 it is apparent that the lower priority base position task is not executed optimally in the SRMTP framework. Transient errors are observed at multiple times even though the tasks are perfectly compatible. This is a known drawback with the SRMTP framework and the price to pay for avoiding algorithmic singularities. In this case, the MTP framework would obtain an optimal velocity reference with respect to the lower priority base

positioning task as long as they are compatible. This is not a problem for the iCAT framework because it computes the null space operators in a different way, ensuring singularity robustness while lower priority tasks are executed more optimally. This is easily seen by comparing Figure 7.1a and Figure 7.3a with respect to the base movement, which is reduced significantly in the iCAT approach.

After $t = 350\,\text{s}$, the end-effector desired position is no longer compatible with the desired base position. It is evident that the strict priority among the equality tasks hold since the end-effector task is executed perfectly, despite the actuation index task being active, while the base positioning task assumes a steady state error.

An important drawback with velocity level redundancy resolution is the fact that operational space acceleration references cannot be specified. This fact inevitably leads to worse tracking performance, however, this cannot really be observed from the simulations in this thesis since only set-point regulation has been considered.

8.1.1.2 Set-based tasks

Since safety related set-based tasks should be introduced as high-priority tasks, the highest priority equality task will suffer from worse transient behavior when set-based tasks are activated within the set-based SRMTP framework. Even though no differences in transient behavior due to the activation of set-based tasks are seen in Figure 7.1 and Figure 7.3, the iCAT framework should execute the highest priority equality task better when set-based tasks are activated.

From the figures it is seen that all set-based tasks are always satisfied for the iCAT framework. For the non-smoothed and smoothed SRMTP approach, the collision avoidance task is not fully satisfied at all times as seen in Figure 7.1d. This is a result of inaccuracies in the dynamic control, which are bound to occur without acceleration feedforward whenever the velocity references change significantly due to a change in end-effector configuration or the activation/deactivation of tasks. Hence, the assumption $\boldsymbol{\zeta} = \boldsymbol{\zeta}_d$ does not hold

straight after the end-effector configuration set point changes, which means that the integral error becomes non-zero before the set-based collision avoidance task is activated as seen in Figure 8.1. This can be significantly improved by employing numeric acceleration feedforward, however, since the velocity references are not continuous when the set-points change and whenever tasks are activated and deactivated, a numeric acceleration scheme will result in excessively large control inputs. Moreover, for real-life implementations, numeric differentiation is sensitive to sensor noise.

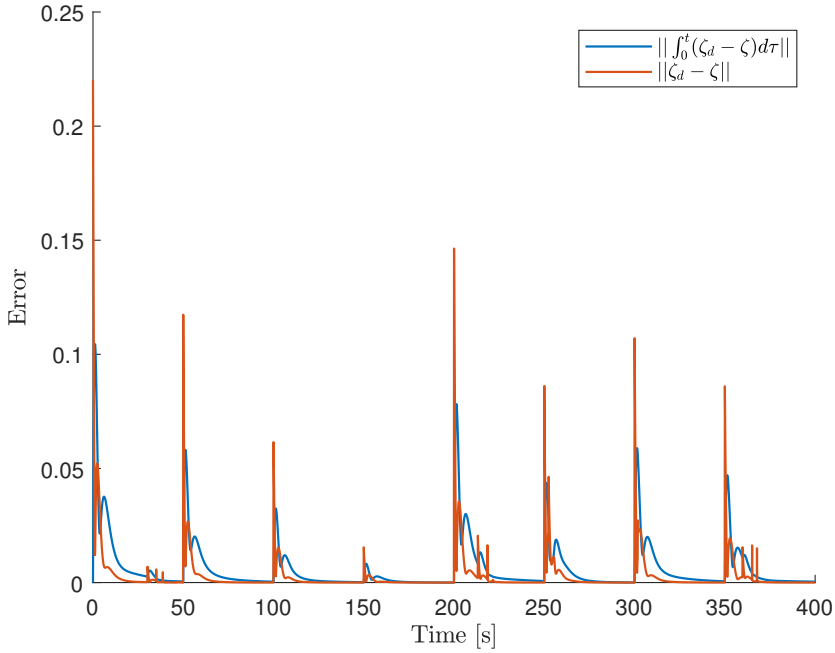


Figure 8.1: Velocity and integral error for the SRMTP framework without smoothing.

The iCAT framework has a high number of parameters, which means that the tuning procedure can be very time consuming [34]. In contrast, the set-based SRMTP framework without smoothing requires very little parameters, which

makes the tuning procedure significantly easier. However, the recursive control law formulation of the iCAT framework in Section 3.4.3 enables an arbitrary number of set-based priority levels to be effortlessly defined. For the set-based SRMTP framework however, the implementation complexity increases drastically whenever individual set-based tasks are added to their own priority levels.

In [18], it is stated that the loss of strict priority between tasks in the iCAT framework when set-based tasks are in transition can lead to undesirable behavior. However, the authors in [5] argue that the loss of strict priority during transitions is a good thing, since the set-based task in transition can share DOFs with lower priority equality task whenever the set-based task is not fully active. The author is inclined to agree with the latter since fully activating set-based tasks before they reach their boundaries would result in slightly overconstraining the system. The only possible undesirable behavior that comes to mind, could be some strange coupling effects between the set-based tasks and the equality tasks. This is not observed in the simulations however.

As noted in Section 3.4, an arbitrary point that is within the valid set, but not in the transition zone is defined for every set-based task. Whenever the set-based task enters the transition zone, it is immediately pushed back inside toward the reference point, however, lower priority tasks may continue to push it toward its boundary while in transition. How close the set-based task is to the boundary of its valid set determines how aggressively it is controlled, and strict priority between the set-based task and lower priority tasks will only hold when the set-based task has reached the boundary. In contrast, the smoothed set-based SRMTP framework attempts to slowly control the set-based task toward its boundary after the transition zone is reached. Strict priority is only lost during the transition from one velocity reference to another, which can be made arbitrarily short by the choice of smoothing function and its parameters. This effectively enables the entire valid set of the set-based task to be utilized, which cannot be said for the iCAT framework.

8.1.2 Acceleration level redundancy resolution

The only acceleration-based kinematic control approach is the proposed extension of the SRMTP framework to acceleration level redundancy. In this approach, the dynamic controller reduces to a feedback linearizing controller, where the acceleration reference is given by the kinematic controller. From Figure 7.5, it is observed that the activation/deactivation of set-based tasks results in large spikes in the joint torque and thruster inputs. This is a result of the need for relatively large derivative gains for the set-based tasks, and the resulting control inputs are obviously not feasible for AIAUVs where thruster dynamics play a key role. However, introducing activation thresholds and smoothing of the acceleration references, the control inputs are more feasible in the sense that the largest spikes are gone, as depicted in Figure 7.6.

The acceleration level SRMTP framework also suffers from non-optimal transient behavior of lower priority tasks. This is expected since the only difference between (6.66) and (6.59), is the velocity or acceleration references. When the desired end-effector position changes, transient errors occur in the lower priority base positioning task, even though the tasks are perfectly compatible. The MTP framework can also be formulated for the acceleration level [48], employing this scheme would result in optimal execution of lower priority tasks, at the expense of introducing algorithmic singularities when tasks are not compatible. Since the simulations have been designed to have conflicting tasks, some kind of singularity handling would have to incorporated.

The set-based tasks are always contained within their valid sets, however, achieving this required a large derivative gain for the collision avoidance task for the non-smoothed approach.

8.2 Operational Space Control

Method 1, which fully linearizes the task dynamics for every task whenever they are compatible achieves slightly better transient behavior with respect

to the lower priority equality tasks. This can be observed by comparing the end-effector attitude and the end-effector and base position plots in Figure 7.7, 7.9 and 7.11, while paying special attention to what the activation of higher priority set-based tasks imply for the equality tasks. However, the difference in equality task performance is essentially negligible in the two first approaches. Method 3, which results in more complicated and nonlinear task space dynamics, is subject to clear performance losses in the end-effector orientation whenever higher priority set-based tasks are activated as seen from Figure 7.11e and 7.11g. This can be explained by the task dynamics given by (4.51), where only the highest priority task has linear task dynamics. The performance losses in lower priority tasks are expected to be even more significant for trajectory tracking.

Method 1 and 2 attains the best performance with respect to the satisfaction of set-based tasks. All non-smoothed approaches struggle with the collision avoidance task, where the minimum distance to the obstacle is not satisfied for any of the approaches. Since operational space control are dynamic approaches operating at the torque level, instantaneously reducing some non-zero velocity toward an obstacle to zero proves difficult. This can be improved by increasing the derivative gain of the collision avoidance task, at the expense of requiring larger rates of change for the control inputs at the time of activation. For set-based tasks such as these, activation thresholds, physical thresholds and safety thresholds should be defined as in [18]. Activation and safety thresholds are in fact defined in every smoothing approach in this thesis, as well as for the collision avoidance task.

When the smoothing approach is employed, the set-based tasks are satisfied for all time in every approach. Once again, the first method yields the best performance with respect to the equality tasks, managing to keep the base in its desired position at all times whenever possible. Moreover, this is the only method where the effects of activating/deactivating set-based tasks cannot be seen in the end-effector orientation. These effects are most noticeable in the third method.

It is interesting to observe the differences in the commanded forces and

torques $\boldsymbol{\tau}$ when set-based tasks are activated and deactivated without smoothing. By inspecting Figures 7.7h, 7.9h and 7.11h, it is seen that method 1 exhibits the largest discontinuities due to the activation/deactivation of set-based tasks, followed by method 2 which performs slightly better, while method 3 performs significantly better. This observation makes sense given that every acceleration reference changes whenever set-based tasks are activated/deactivated within the first approach. Additionally, method 1 and 2 employ the null-space operator of all the higher priority tasks within the task specific inertia matrix, which means that the task specific inertia matrix changes for all tasks every time set-based tasks are activated/deactivated. In contrast, the expression for the task specific inertia matrix does not change depending on which tasks are active for method 3. Moreover, the acceleration references are constant for all time, which means that the discontinuities in the commanded forces and torques are only to be attributed to the change in null space operators and in the highest priority control torque.

By smoothing the commanded forces and torques, they can be made arbitrarily smooth by choice of smoothing function parameters. However, since strict priority among tasks is not satisfied whenever the control torque is transitioning from one mode to another the transition time should be minimized. Therefore, there is an inherent trade-off between strict priority and smoothness of the commanded forces and torques within this approach. The activation thresholds can be increased if longer transition times for smoother references are needed, which comes at the expense of slightly overconstraining the system since it becomes harder to deactivate set-based tasks when they are still being slowly controlled toward their boundaries as explained in Section 8.3.1.

8.2.1 Uncertainty in the dynamic parameters

Uncertainties in the inertia matrix used to calculate the null space operators imply that the null space operators are no longer dynamically consistent, which further implies that there is interference among tasks since lower priority tasks

may generate accelerations affecting higher priority tasks. This makes it harder to ensure the satisfaction of high priority set-based tasks. However, given that the results in [26] hold, such that (4.12) is statically consistent even when the inertia matrix contains modeling errors, static consistency should also hold for the generalized null space operator defined by (4.17) and (4.18). This is justified by the fact that the generalized null space operator reduces to (4.12) for the case of two tasks, as well as satisfying the same properties that was utilized in the proof of static consistency in [26].

From inspection of Figure 7.14, Figure 7.15 and Figure 7.16, it is clear that even though the transient behavior is significantly worse, the equality tasks still reach their desired values at steady-state whenever possible according to their priority. Even though the equality tasks are no longer compatible at $t = 350\,\mathrm{s}$, priority among the equality tasks is still satisfied in steady-state, since the end-effector configuration reaches its desired values while the base position suffers a constant error at the end of the simulation. Because of the excessive chattering in the activation and deactivation of the actuation index task, the base positioning task is not strictly on priority level 3 in the time interval $t \in [350, 400]$. This makes it harder to verify from simulations that tasks on priority level 3 or lower does not interfere with higher priority tasks in steady state conditions.

It is interesting to observe the difference in the thruster and joint torque inputs when the inertia matrix is fully known and when its not. In Figure 7.7, the control inputs decrease instantaneously when the equality tasks are incompatible at $t = 350\,\mathrm{s}$ and the inertia matrix is perfectly known. In Figure 7.14 the control inputs rapidly increase straight after the equality tasks are no longer compatible. The same conclusions can be drawn from comparing the control input plots for method 2 and method 3 with and without uncertainties. This can be explained by the fact that the null space operators are no longer dynamically consistent when there are uncertainties in the inertia matrix, which means that lower priority tasks can generate accelerations affecting higher priority tasks. Hence, the acceleration effects generated by lower priority tasks are not fully removed by the null space operators. Consequently, increasing the control gains of a lower

priority task will lead to larger transient errors in the higher priority task and larger control inputs when the tasks are not compatible. For instance, method 3 could have performed significantly better with respect to the base positioning task by increasing the control gains of the base positioning task, however, this results in very large control inputs when the tasks are no longer compatible at $t = 350\,\mathrm{s}$.

Method 1 still outperforms method 2 with respect to the transient behavior of the lower priority base positioning task and the end-effector orientation whenever set-based tasks are active. However, activation and deactivation of tasks leads to more abrupt changes in the commanded forces and torques for method 1 compared to method 2, which is also seen in the thruster and joint torque inputs. This was observed and discussed in the simulations with perfect knowledge of the inertia matrix as well, but it is seen more clearly here.

Method 3 performs a lot worse than both method 1 and 2 with respect to the satisfaction of set-based tasks and the transient behavior of lower priority tasks as seen in Figure 7.16. Activation and deactivation of set-based tasks may lead to smaller instantaneous changes in the commanded forces and torques. However, the thruster and joint torque inputs generally have a much larger magnitude for this method. Moreover, the exclusion of the null space operator of the higher priority tasks seem to make method 3 more sensitive to interference from lower priority tasks, making it harder to satisfy set-based control objectives while increasing the magnitude of the control inputs.

Finally, chattering in the activation and deactivation of tasks seems to be highly problematic when the inertia matrix is uncertain. A possible improvement could be to not deactivate set-based tasks until they have been active for a certain period of time. Alternatively, the dynamic model could be employed in order to predict the task velocities for several time steps into the future, only deactivating the task if the task velocity has the correct and same sign at every iteration. A completely different and more practical approach could be to to filter the output commanded forces and torques [49].

8.3 Kinematic Control vs Operational Space Control

The weakness of the SRMTP framework with respect to the optimal execution of lower priority tasks is evident from both the velocity level and acceleration level redundancy resolution schemes, the base position exhibits transient errors essentially every time the desired end-effector configuration changes. For acceleration level redundancy resolution, transient errors can clearly be seen in the end-effector task when set-based tasks are activated, even though the tasks are compatible. These effects are not observed at all in the operational space controller that dynamically decouples the tasks from Section 6.7.1, they are barely noticeable in the non-compensation approach from Section 6.7.2 while transient errors in the end-effector orientation as a result of the activation of set-based collision avoidance and actuation index tasks are clearly seen in the last operational space controller from Section 6.7.3. However, the transient errors in the base positioning task are considerably smaller for all operational space approaches compared to the acceleration level SRMTP framework. Poor execution of lower priority tasks is a bigger drawback when high-priority set-based tasks are considered, since the equality task of highest importance always suffers from poor transient behavior whenever set-based tasks are activated.

Kinematic control decomposes the control problem into a kinematic and dynamic controller. The set-based kinematic controllers only utilize kinematic relations in order to predict task velocities. However, they inherently rely on the assumption $\boldsymbol{\zeta} = \boldsymbol{\zeta}_d$ for velocity level kinematic control or $\dot{\boldsymbol{\zeta}} = \dot{\boldsymbol{\zeta}}_d$ for acceleration level kinematic control, but otherwise pay very little attention to how this is achieved by a dynamic controller. This is often a good assumption for industrial robot manipulators with fast tracking properties where the model parameters are much easier to obtain. However, for an AIAUV, thruster dynamics and hydrodynamic effects have a significant impact on the dynamics of the system. The AIAUV equations of motion are derived under several simplifying assumptions on the fluid flow [38], but the determination of the resulting model

parameters is still very challenging due to their time-varying nature. Moreover, drag forces have a much larger effect on the AIAUV base compared to a heavier UVMS base, since AIAUV base is attached to any of the links, which does not have significant inertia with respect to the rest of the system. Finally, inspection and intervention tasks are performed at low-speed, where thruster dynamics becomes essential to the control problem [50]. Consequently, designing dynamic controllers satisfying the kinematic control assumptions for AIAUVs is difficult and not straightforward. Furthermore, by employing a kinematic control scheme for an AIAUV in work mode, the contact forces with the environment can only be handled by indirect force control, which is an important drawback compared to operational space control.

When the dynamic controllers no longer accurately track the output of the kinematic controller fast and accurately enough, the overall control performance degrades significantly. This is especially true for the set-based acceleration control approach, which explicitly relies on this assumption to predict task velocities which determine the activation and deactivation of tasks. Priority among tasks can no longer be guaranteed when there are errors in the dynamic control.

One significantly challenge related to operational space control of an AIAUV is that it relies on inverse dynamic control, which requires accurate knowledge of the dynamic model. Identifying the model parameters for underwater vehicles is very difficult in general [19]. The set-based operational space approach outlined in this thesis is even more vulnerable to modeling inaccuracies since the predicted task velocities are estimated from the dynamic model as observed from (6.70) and (6.71). However, since modeling errors do not distort the operational space null-space projections in steady-state, tasks at different priority levels are not coupled in steady-state, even though the dynamic consistency property is lost and tasks are coupled during the transient phase. This legitimizes task priority schemes for operational space control even if the model parameters are not exactly known. The same cannot be said for velocity or acceleration level kinematic control, where tasks are coupled whenever the assumptions on the dynamic controller

does not hold. Modeling errors can be mitigated by utilizing sliding mode or adaptive control within an operational space task priority scheme. An adaptive control approach would increase the feasibility of the proposed approach for operational space set-based control, since the obtained estimates would improve the predicted task velocities $\dot{\boldsymbol{\sigma}}$ employed in Algorithm 2.

8.3.1 Challenges related to set-based acceleration and operational space control

When resolving redundancy at the velocity level, the set-based tasks are activated when the boundary of the valid set is reached and the sign of the task derivative entails that the task will leave its valid set if not accounted for. At this point, the set-based task error is zero, and since lower priority tasks cannot generate *velocities* that interfere with the high priority set-based task, the set-based task velocity becomes zero when lower priority tasks are projected through the null space of the Jacobian of the set-based task, without any other need for control of the set-based task. However, when set-based tasks are introduced as high priority tasks in an acceleration level or torque level redundancy resolution scheme, lower priority tasks cannot generate *accelerations* affecting higher priority set-based task. Since the set-based task must have a non-zero task velocity pushing it outside of its valid set when it was activated, projecting the lower priority tasks through the null space of the higher priority task only results in the acceleration of the set-based task being zero, while some constant velocity still continues to push the task outside of its valid set. Hence, the acceleration reference of the set-based task has to be considered, which in its general form is given by (2.27).

In order to ensure stability, the derivative gain $\boldsymbol{K}_d$ has to be non-zero. A proportional term controlling the task towards the boundary should in theory help by pushing the task variable back toward its valid set in case of overshoots. However, this leads to excessive chattering in the activation/deactivation of tasks. This can be explained by noticing that after a small overshoot above or below the boundary of the valid set, the proportional term tries to push it

back towards the boundary, creating a set-based task velocity that pushes the task toward its valid set. When the set-based task velocity of a less restrictive acceleration or torque reference is evaluated in the next time step according to

$$\dot{\boldsymbol{\sigma}}(t_k) = \boldsymbol{J}\left(\boldsymbol{\zeta}(t_{k-1}) + \Delta t\dot{\boldsymbol{\zeta}}_d(t_{k-1})\right), \tag{8.1}$$

for acceleration based control or (6.70) and (6.71) for torque level control, this acceleration or torque reference may not change the task velocity of the set-based task enough in order to change its sign. The set-based task is subsequently deactivated, only to be activated a few time steps later when the less restrictive reference has managed to change the sign of the task velocity.

The magnitude of the derivative gain of the set-based task plays a significant role in regards to the dynamics of the set-based task after it is activated. If lower priority tasks have created large set-based task velocities before the time of activation, the derivative gain of the set-based task needs to be large in order to freeze the task within a reasonable amount of time. The conclusion is that the non-smoothed approach for set-based acceleration or torque control does not perform satisfactory. Overshoots above or below the boundary of the valid set will occur every time a set based task is activated, which requires the need for large derivative gains in order to mitigate the overshoot. Moreover, proportional terms cannot be used due to excessive chattering, which means that set-based tasks are frozen slightly outside their valid sets at best.

The smoothing approach where activation thresholds are defined around the boundaries of the valid sets of the set-based tasks performs better. This approach yields continuous acceleration or torque references, while set-based tasks are less likely to violate their valid sets. However, by including proportional terms and controlling the set-based tasks toward their boundaries, the task obtains a positive or negative task velocity, depending on whether it is controlled toward an upper or lower bound, respectively. Moreover, set-based operational space control is a dynamic approach at the force/torque level, where the task velocity for a less restrictive torque reference is predicted by integrating (6.70) and pre-

multiplying the result with the task Jacobian in question. Therefore, for a task to be deactivated when it is already active and slowly being controlled toward its limit, a less restrictive torque reference needs to change the sign of the task velocity over the integration time interval. This time interval is one time step of the fixed-step ODE solver in the simulations, which is set to 0.1 s, making it very hard to deactivate tasks that have not yet reached their boundaries. In conclusion, introducing activation thresholds and proportional terms such that the set-based tasks are activated earlier and slowly controlled toward their boundaries makes it easier to satisfy the constraints on the set-based tasks, while it simultaneously becomes harder for lower priority tasks to deactivate the set-based tasks when they are still controlled toward their boundaries.

Chapter 9

Conclusions and Future Work

In this thesis, several approaches to redundancy resolution of robotic systems have been investigated, where all approaches are in the form of task priority control frameworks. A special focus has been given to set-based tasks. The set-based SRMTP framework has been extended to handle multidimensional set-based tasks and an extension to acceleration control has also been presented. The same methodology for handling set-based tasks has been utilized in order to formalize a scheme for set-based operational space control. Within this framework, the null space operator from [22] has been proven to be dynamically consistent when the inertia matrix used to compute it is fully known. Moreover, whenever the inertia matrix is uncertain, static consistency has been proven, which entails that tasks at different priority levels are not coupled at steady state conditions.

All of the schemes for set-based task priority control have been formulated and simulated for an AIAUV in set-point regulation scenarios. Simulations have shown decent performance for all of the frameworks, when perfect knowledge of

dynamic parameters are assumed, and thruster dynamics are neglected. These frameworks for set-based control all rely on activation and deactivation of set-based tasks, which may induce somewhat large and sudden changes in the control inputs. While this may work for industrial or humanoid robots with sufficiently fast actuator dynamics, the author conjectures that it is a bigger challenge for AIAUVs with significant thruster dynamics. To mitigate large and sudden changes in the references, activation thresholds can be defined in addition to the boundaries of the set-based tasks. This permits smaller control gains for set-based acceleration and operational space control, since the activation thresholds serve as a soft activation where the set-based task is slowly controlled toward its boundary as long as other tasks cannot guarantee the satisfaction of the set-based task. Control reference smoothing has been used in connection with the activation threshold approach in order to remove any discontinuities arising from activation and deactivation of tasks.

In velocity level kinematic control, operational space acceleration references cannot be accounted for, which inevitably leads to worse tracking behavior. Acceleration level kinematic control does not suffer from this drawback, but satisfying the assumption of perfect tracking of the acceleration references may be harder in practice. Furthermore, all kinematic control schemes reduce the handling of contact forces with the environment to indirect force control, which is a drawback with respect to AIAUVs in work mode.

The operational space formalism is a very promising approach for AIAUVs operating in intervention mode, where contact forces with the environment need to be considered. It permits any kind of force control scheme to be employed, while static consistency of the null space operators can always be ensured, which implies that tasks at different priority levels are not coupled in steady state, even though the operational space controllers contain modeling errors.

9.1 Future work

Simulations for a trajectory tracking scenario should be conducted. This would better highlight the difference between velocity control and acceleration/torque control, where task space accelerations can be utilized. Thruster dynamics should be taken into consideration as they are essential to the control problem at low speed regimes. More realistic behavior without modeling the full thruster dynamics could be also obtained by limiting the rate of change of the commanded thrust. Such constraints could be integrated into the control allocation problem, resulting in a constrained control allocation algorithm.

The method for activation and deactivation of set-based tasks within acceleration based kinematic control and operational space control needs some refinement. As discussed in Section 8.3.1, adding proportional control terms for set-based tasks within these approaches is not straightforward, as it often leads to chattering in the activation/deactivation of tasks. Moreover, by employing activation thresholds as done in the smoothing approaches, the set-based tasks become harder to deactivate once activated. Future work should investigate methods that does not rely on the extended tangent cone, for instance by looking at the possibility of extending the activation/deactivation scheme from [5] to operational space control.

There is a lot of future work to be done with respect to task priority operational space control. Sliding mode or adaptive controllers should be investigated within this framework, and force control schemes for interaction with the environment should be investigated for AIAUVs in intervention mode.

Robust or adaptive dynamic controllers for use in connection with kinematic control has not been investigated. Hence, the kinematic control simulations have essentially just showed the feasibility of set-based kinematic control given a perfect dynamic controller as done in [49]. Therefore, it would be of interest to consider kinematic control with a robust dynamic controller to observe the effect that would have on the kinematic controller.

Finally, a formal and rigorous stability analysis of acceleration level kinematic

control and task priority operational space control would improve the theoretical foundation of the frameworks.

Appendix A

Proofs

In [22] a null space operator for the $n-1$ higher priority tasks is defined by

$$\boldsymbol{N}_{P(n)}^{\mathrm{T}} = \boldsymbol{I} - \sum_{i=1}^{n-1} \boldsymbol{M}^{-1}\boldsymbol{N}_{P(i)}\boldsymbol{J}_i^{\mathrm{T}} \left(\boldsymbol{J}_i\boldsymbol{N}_{P(i)}^{\mathrm{T}}\boldsymbol{M}^{-1}\boldsymbol{N}_{P(i)}\boldsymbol{J}_i^{\mathrm{T}}\right)^{-1} \boldsymbol{J}_i\boldsymbol{N}_{P(i)}^{\mathrm{T}} \tag{A.1}$$

$$\boldsymbol{N}_{P(n)} = \boldsymbol{I} - \sum_{i=1}^{n-1} \boldsymbol{N}_{P(i)}\boldsymbol{J}_i^{\mathrm{T}} \left(\boldsymbol{J}_i\boldsymbol{N}_{P(i)}^{\mathrm{T}}\boldsymbol{M}^{-1}\boldsymbol{N}_{P(i)}\boldsymbol{J}_i^{\mathrm{T}}\right)^{-1} \boldsymbol{J}_i\boldsymbol{N}_{P(i)}^{\mathrm{T}}\boldsymbol{M}^{-1}, \tag{A.2}$$

where $P(n) = \{1, 2, \ldots, n-1\}$ represents the set of all higher priority tasks. The null space operator can be formulated recursively as

$$\boldsymbol{N}_{P(1)} = \boldsymbol{I}, \tag{A.3}$$

$$\boldsymbol{N}_{P(n+1)} = \boldsymbol{N}_{P(n)} \left(\boldsymbol{I} - \boldsymbol{J}_n^{\mathrm{T}} \left(\boldsymbol{J}_n\boldsymbol{N}_{P(n)}^{\mathrm{T}}\boldsymbol{M}^{-1}\boldsymbol{N}_{P(n)}\boldsymbol{J}_n^{\mathrm{T}}\right)^{-1} \boldsymbol{J}_n\boldsymbol{N}_{P(n)}^{\mathrm{T}}\boldsymbol{M}^{-1}\right). \tag{A.4}$$

Two interesting properties of the null-space projection matrix are now proven. Firstly, the null-space projector satisfies the following property

$$\boldsymbol{N}_{P(n)}^{\mathrm{T}}\boldsymbol{M}^{-1} = \boldsymbol{M}^{-1}\boldsymbol{N}_{P(n)} \tag{A.5}$$

Proof. Pre-multiplication of (A.2) by $\boldsymbol{M}^{-1}$ yields

$$\boldsymbol{M}^{-1}\boldsymbol{N}_{P(n)} = \boldsymbol{M}^{-1} - \sum_{i=1}^{n-1} \boldsymbol{M}^{-1}\boldsymbol{N}_{P(i)}\boldsymbol{J}_i^{\mathrm{T}} \left(\boldsymbol{J}_i\boldsymbol{N}_{P(i)}^{\mathrm{T}}\boldsymbol{M}^{-1}\boldsymbol{N}_{P(i)}\boldsymbol{J}_i^{\mathrm{T}}\right)^{-1} \boldsymbol{J}_i\boldsymbol{N}_{P(i)}^{\mathrm{T}}\boldsymbol{M}^{-1} \tag{A.6}$$

$$= \left(\boldsymbol{I} - \sum_{i=1}^{n-1} \boldsymbol{M}^{-1}\boldsymbol{N}_{P(i)}\boldsymbol{J}_i^{\mathrm{T}} \left(\boldsymbol{J}_i\boldsymbol{N}_{P(i)}^{\mathrm{T}}\boldsymbol{M}^{-1}\boldsymbol{N}_{P(i)}\boldsymbol{J}_i^{\mathrm{T}}\right)^{-1} \boldsymbol{J}_i\boldsymbol{N}_{P(i)}^{\mathrm{T}}\right)\boldsymbol{M}^{-1} \tag{A.7}$$

$$= \boldsymbol{N}_{P(n)}^{\mathrm{T}}\boldsymbol{M}^{-1} \tag{A.8}$$

□

Furthermore, the null-space projector obeys the idempotent property $\boldsymbol{N}_{P(n)}^2 = \boldsymbol{N}_{P(n)}$.

Proof. For $n = 2$ equation (A.4) yields

$$\begin{aligned}
\boldsymbol{N}_{P(2)}^2 &= \left(\boldsymbol{I} - \boldsymbol{J}_2^{\mathrm{T}}\left(\boldsymbol{J}_2\boldsymbol{M}^{-1}\boldsymbol{J}_2^{\mathrm{T}}\right)^{-1}\boldsymbol{J}_2\boldsymbol{M}^{-1}\right)^2 \\
&= \boldsymbol{I} - 2\boldsymbol{J}_2^{\mathrm{T}}\left(\boldsymbol{J}_2\boldsymbol{M}^{-1}\boldsymbol{J}_2^{\mathrm{T}}\right)^{-1}\boldsymbol{J}_2\boldsymbol{M}^{-1} \\
&\quad + \boldsymbol{J}_2^{\mathrm{T}}\left(\boldsymbol{J}_2\boldsymbol{M}^{-1}\boldsymbol{J}_2^{\mathrm{T}}\right)^{\mathrm{T}}\boldsymbol{J}_2\boldsymbol{M}^{-1}\boldsymbol{J}_2^{\mathrm{T}}\left(\boldsymbol{J}_2\boldsymbol{M}^{-1}\boldsymbol{J}_2^{\mathrm{T}}\right)^{\mathrm{T}}\boldsymbol{J}_2\boldsymbol{M}^{-1} \\
&= \boldsymbol{I} - \boldsymbol{J}_2^{\mathrm{T}}\left(\boldsymbol{J}_2\boldsymbol{M}^{-1}\boldsymbol{J}_2^{\mathrm{T}}\right)^{\mathrm{T}}\boldsymbol{J}_2\boldsymbol{M}^{-1} \\
&= \boldsymbol{N}_{P(2)}
\end{aligned} \tag{A.9}$$

Assume that $\boldsymbol{N}_{P(n)}^2 = \boldsymbol{N}_{P(n)}$ holds for any n. Consider the case with $n+1$

$$\begin{aligned}
\boldsymbol{N}_{P(n+1)}^2 &= \boldsymbol{N}_{P(n)}^2 \left(\boldsymbol{I} - \boldsymbol{J}_n^{\mathrm{T}} \left(\boldsymbol{J}_n \boldsymbol{N}_{P(n)}^{\mathrm{T}} \boldsymbol{M}^{-1} \boldsymbol{N}_{P(n)} \boldsymbol{J}_n^{\mathrm{T}} \right)^{-1} \boldsymbol{J}_n \boldsymbol{N}_{P(n)}^{\mathrm{T}} \boldsymbol{M}^{-1} \right)^2 \\
&= \boldsymbol{N}_{P(n)} \left(\boldsymbol{I} - \boldsymbol{J}_n^{\mathrm{T}} \left(\boldsymbol{J}_n \boldsymbol{M}^{-1} \boldsymbol{N}_{P(n)}^2 \boldsymbol{J}_n^{\mathrm{T}} \right)^{-1} \boldsymbol{J}_n \boldsymbol{M}^{-1} \boldsymbol{N}_{P(n)} \right)^2 \\
&= \boldsymbol{N}_{P(n)} \left(\boldsymbol{I} - \boldsymbol{J}_n^{\mathrm{T}} \left(\boldsymbol{J}_n \boldsymbol{M}^{-1} \boldsymbol{N}_{P(n)} \boldsymbol{J}_n^{\mathrm{T}} \right)^{-1} \boldsymbol{J}_n \boldsymbol{M}^{-1} \boldsymbol{N}_{P(n)} \right)^2 \\
&= \boldsymbol{N}_{P(n)} \left(\boldsymbol{I} - 2\boldsymbol{J}_n^{\mathrm{T}} \left(\boldsymbol{J}_n \boldsymbol{M}^{-1} \boldsymbol{N}_{P(n)} \boldsymbol{J}_n^{\mathrm{T}} \right)^{-1} \boldsymbol{J}_n \boldsymbol{M}^{-1} \boldsymbol{N}_{P(n)} \right. \\
&+ \left. \boldsymbol{J}_n^{\mathrm{T}} \left(\boldsymbol{J}_n \boldsymbol{M}^{-1} \boldsymbol{N}_{P(n)} \boldsymbol{J}_n^{\mathrm{T}} \right)^{-1} \boldsymbol{J}_n \boldsymbol{M}^{-1} \boldsymbol{N}_{P(n)} \boldsymbol{J}_n^{\mathrm{T}} \left(\boldsymbol{J}_n \boldsymbol{M}^{-1} \boldsymbol{N}_{P(n)} \boldsymbol{J}_n^{\mathrm{T}} \right)^{-1} \boldsymbol{J}_n \boldsymbol{M}^{-1} \boldsymbol{N}_{P(n)} \right) \\
&= \boldsymbol{N}_{P(n)} \left(\boldsymbol{I} - \boldsymbol{J}_n^{\mathrm{T}} \left(\boldsymbol{J}_n \boldsymbol{M}^{-1} \boldsymbol{N}_{P(n)} \boldsymbol{J}_n^{\mathrm{T}} \right)^{-1} \boldsymbol{J}_n \boldsymbol{M}^{-1} \boldsymbol{N}_{P(n)} \right) \\
&= \boldsymbol{N}_{P(n+1)}
\end{aligned} \tag{A.10}$$

□

Combining these properties the recursive null-space projection matrix equations may be rewritten

$$\boldsymbol{N}_{P(1)} = \boldsymbol{I} \tag{A.11}$$

$$\boldsymbol{N}_{P(n+1)} = \boldsymbol{N}_{P(n)} \left(\boldsymbol{I} - \boldsymbol{J}_n^{\mathrm{T}} \left(\boldsymbol{J}_n \boldsymbol{M}^{-1} \boldsymbol{N}_{P(n)} \boldsymbol{J}_n^{\mathrm{T}} \right)^{-1} \boldsymbol{J}_n \boldsymbol{M}^{-1} \boldsymbol{N}_{P(n)} \right) \tag{A.12}$$

$$\boldsymbol{N}_{P(n+1)} = \left(\boldsymbol{I} - \boldsymbol{N}_{P(n)} \boldsymbol{J}_n^{\mathrm{T}} \left(\boldsymbol{J}_n \boldsymbol{M}^{-1} \boldsymbol{N}_{P(n)} \boldsymbol{J}_n^{\mathrm{T}} \right)^{-1} \boldsymbol{J}_n \boldsymbol{M}^{-1} \right) \boldsymbol{N}_{P(n)} \tag{A.13}$$

By defining

$$\boldsymbol{M}_i = \left(\boldsymbol{J}_i \boldsymbol{M}^{-1} \boldsymbol{N}_{P(i)} \boldsymbol{J}_i^{\mathrm{T}} \right)^{-1} \tag{A.14}$$

$$\overline{\boldsymbol{J}}_i = \boldsymbol{M}^{-1} \boldsymbol{J}_i^{\mathrm{T}} \boldsymbol{M}_i \tag{A.15}$$

the null-space projection equations may be rewritten

$$\boldsymbol{N}_{P(1)} = \boldsymbol{I} \tag{A.16}$$

$$\boldsymbol{N}_{P(n+1)} = \boldsymbol{N}_{P(n)} \left(\boldsymbol{I} - \boldsymbol{J}_n^{\mathrm{T}} \overline{\boldsymbol{J}}_n^{\mathrm{T}} \boldsymbol{N}_{P(n)} \right) \tag{A.17}$$

$$\boldsymbol{N}_{P(n+1)} = \left(\boldsymbol{I} - \boldsymbol{N}_{P(n)} \boldsymbol{J}_n^{\mathrm{T}} \overline{\boldsymbol{J}}_n^{\mathrm{T}} \right) \boldsymbol{N}_{P(n)} \tag{A.18}$$

Note that for $n = 2$, (A.1) reduces to

$$\boldsymbol{N}_{P(2)}^{\mathrm{T}} = \boldsymbol{I} - \boldsymbol{M}^{-1} \boldsymbol{N}_{P(1)} \boldsymbol{J}_1^{\mathrm{T}} \left(\boldsymbol{J}_1 \boldsymbol{N}_{P(1)}^{\mathrm{T}} \boldsymbol{M}^{-1} \boldsymbol{N}_{P(1)} \boldsymbol{J}_1^{\mathrm{T}} \right)^{-1} \boldsymbol{J}_1 \boldsymbol{N}_{P(1)} \tag{A.19}$$

$$= \boldsymbol{I} - \boldsymbol{M}^{-1} \boldsymbol{I} \boldsymbol{J}_1^{\mathrm{T}} \left(\boldsymbol{J}_1 \boldsymbol{I} \boldsymbol{M}^{-1} \boldsymbol{I} \boldsymbol{J}_1^{\mathrm{T}} \right)^{-1} \boldsymbol{J}_1 \boldsymbol{I} \tag{A.20}$$

$$= \boldsymbol{I} - \boldsymbol{M}^{-1} \boldsymbol{J}_1^{\mathrm{T}} \boldsymbol{M}_1 \boldsymbol{J}_1 \tag{A.21}$$

$$= \boldsymbol{I} - \overline{\boldsymbol{J}}_1 \boldsymbol{J}_1 \tag{A.22}$$

which is the same solution as (4.12) since

$$\left(\boldsymbol{N}_{P(2)}^{\mathrm{T}} \right)^{\mathrm{T}} = \boldsymbol{N}_{P(2)} = \boldsymbol{I} - \boldsymbol{J}_1^{\mathrm{T}} \overline{\boldsymbol{J}_1}^{\mathrm{T}} \tag{A.23}$$

The purpose of the null-space operator defined in (A.1) is to ensure that lower-priority tasks have no acceleration effects in all preceding objectives, which is to say that the null space operator is dynamically consistent. A null space operator is dynamically consistent if it satisfies the following property [51]

$$\boldsymbol{J}_i \boldsymbol{M}^{-1} \boldsymbol{N}_{P(n)} = \boldsymbol{0}, \quad \forall\, i \in P(n) \tag{A.24}$$

Proof. Consider the case with $P(n+1)$ and $i = n$, pre-multiplying (A.12) by

$\boldsymbol{J}_n\boldsymbol{M}^{-1}$ yields

$$\begin{aligned}
\boldsymbol{J}_n\boldsymbol{M}^{-1}\boldsymbol{N}_{P(n+1)} &= \boldsymbol{J}_n\boldsymbol{M}^{-1}\boldsymbol{N}_{P(n)}\left(\boldsymbol{I} - \boldsymbol{J}_n^{\mathrm{T}}\left(\boldsymbol{J}_n\boldsymbol{M}^{-1}\boldsymbol{N}_{P(n)}\boldsymbol{J}_n^{\mathrm{T}}\right)^{-1}\boldsymbol{J}_n\boldsymbol{M}^{-1}\boldsymbol{N}_{P(n)}\right) \\
&= \left(\boldsymbol{J}_n - \boldsymbol{J}_n\boldsymbol{M}^{-1}\boldsymbol{N}_{P(n)}\boldsymbol{J}_n^{\mathrm{T}}\left(\boldsymbol{J}_n\boldsymbol{M}^{-1}\boldsymbol{N}_{P(n)}\boldsymbol{J}_n^{\mathrm{T}}\right)^{-1}\boldsymbol{J}_n\right)\boldsymbol{M}^{-1}\boldsymbol{N}_{P(n)} \\
&= \left(\boldsymbol{J}_n - \boldsymbol{J}_n\right)\boldsymbol{M}^{-1}\boldsymbol{N}_{P(n)} \\
&= \boldsymbol{0}
\end{aligned} \tag{A.25}$$

Let $i \in P(n{+}1)$ and define $\boldsymbol{X}_n = \left(\boldsymbol{I} - \boldsymbol{J}_n^{\mathrm{T}}\left(\boldsymbol{J}_n\boldsymbol{M}^{-1}\boldsymbol{N}_{P(n)}\boldsymbol{J}_n^{\mathrm{T}}\right)^{-1}\boldsymbol{J}_n\boldsymbol{M}^{-1}\boldsymbol{N}_{P(n)}\right)$

$$\begin{aligned}
\boldsymbol{J}_i\boldsymbol{M}^{-1}\boldsymbol{N}_{P(n+1)} &= \boldsymbol{J}_i\boldsymbol{M}^{-1}\boldsymbol{N}_{P(n)}\boldsymbol{X}_n && \text{(A.26)} \\
&= \boldsymbol{J}_i\boldsymbol{M}^{-1}\boldsymbol{N}_{P(n-1)}\boldsymbol{X}_{n-1}\boldsymbol{X}_n && \text{(A.27)} \\
&= \underbrace{\boldsymbol{J}_i\boldsymbol{M}^{-1}\boldsymbol{N}_{P(i+1)}}_{=\boldsymbol{0}}\boldsymbol{X}_{i+1}\cdots\boldsymbol{X}_{n-1}\boldsymbol{X}_n && \text{(A.28)} \\
&= \boldsymbol{0} && \text{(A.29)}
\end{aligned}$$

□

The following also holds for the generalized null space operator

$$\boldsymbol{N}_{P(n)}\boldsymbol{J}_i^{\mathrm{T}} = \boldsymbol{0}, \quad \forall\, i \in P(n) \tag{A.30}$$

Proof. Consider the case with $P(n+1)$ and $i = n$, equation (A.13) yields

$$\begin{aligned}
\boldsymbol{N}_{P(n+1)}\boldsymbol{J}_n^{\mathrm{T}} &= \left(\boldsymbol{I} - \boldsymbol{N}_{P(n)}\boldsymbol{J}_n^{\mathrm{T}}\left(\boldsymbol{J}_n\boldsymbol{M}^{-1}\boldsymbol{N}_{P(n)}\boldsymbol{J}_n^{\mathrm{T}}\right)^{-1}\boldsymbol{J}_n\boldsymbol{M}^{-1}\right)\boldsymbol{N}_{P(n)}\boldsymbol{J}_n^{\mathrm{T}} \\
&= \boldsymbol{N}_{P(n)}\boldsymbol{J}_n^{\mathrm{T}} - \boldsymbol{N}_{P(n)}\boldsymbol{J}_n^{\mathrm{T}}\left(\boldsymbol{J}_n\boldsymbol{M}^{-1}\boldsymbol{N}_{P(n)}\boldsymbol{J}_n^{\mathrm{T}}\right)^{-1}\boldsymbol{J}_n\boldsymbol{M}^{-1}\boldsymbol{N}_{P(n)}\boldsymbol{J}_n^{\mathrm{T}} \\
&= \boldsymbol{N}_{P(n)}\boldsymbol{J}_n^{\mathrm{T}} - \boldsymbol{N}_{P(n)}\boldsymbol{J}_n^{\mathrm{T}} \\
&= \boldsymbol{0}
\end{aligned} \tag{A.31}$$

Let $i \in P(n{+}1)$ and define $\boldsymbol{Y}_n = \left(\boldsymbol{I} - \boldsymbol{N}_{P(n)}\boldsymbol{J}_n^{\mathrm{T}}\left(\boldsymbol{J}_n\boldsymbol{M}^{-1}\boldsymbol{N}_{P(n)}\boldsymbol{J}_n^{\mathrm{T}}\right)^{-1}\boldsymbol{J}_n\boldsymbol{M}^{-1}\right)$, equation (A.13) yields

$$\begin{aligned}
\boldsymbol{N}_{P(n+1)}\boldsymbol{J}_i^{\mathrm{T}} &= \boldsymbol{Y}_n\boldsymbol{N}_{P(n)}\boldsymbol{J}_i^{\mathrm{T}} \\
&= \boldsymbol{Y}_n\boldsymbol{Y}_{n-1}\boldsymbol{N}_{P(n-1)}\boldsymbol{J}_i^{\mathrm{T}} \\
&= \boldsymbol{Y}_n\boldsymbol{Y}_{n-1}\cdots\boldsymbol{Y}_{i+1}\underbrace{\boldsymbol{N}_{P(i+1)}\boldsymbol{J}_i^{\mathrm{T}}}_{=\boldsymbol{0}} \\
&= \boldsymbol{0}.
\end{aligned} \tag{A.32}$$

□

Note that the result above holds for all values of the inertia matrix used to compute the null space operator, as long as the inertia matrix is positive definite.

References

[1] J. Sverdrup-Thygeson, E. Kelasidi, K. Y. Pettersen, and J. T. Gravdahl, "The underwater swimming manipulator—a bioinspired solution for subsea operations," *IEEE Journal of Oceanic Engineering*, vol. 43, no. 2, pp. 402–417, April 2018.

[2] E. I. Grøtli, J. Tjønnås, J. Azpiazu, A. A. Transeth, and M. Ludvigsen, "Towards more autonomous rov operations: Scalable and modular localization with experiment data," *IFAC-PapersOnLine*, vol. 49, no. 23, pp. 173 – 180, 2016, 10th IFAC Conference on Control Applications in Marine SystemsCAMS 2016.

[3] A. Kohl, "Redundancy resolution methods for articulated intervention-auvs," NTNU, Tech. Rep., 2018.

[4] S. Moe, G. Antonelli, A. R. Teel, K. Y. Pettersen, and J. Schrimpf, "Set-based tasks within the singularity-robust multiple task-priority inverse kinematics framework: General formulation, stability analysis, and experimental results," *Frontiers in Robotics and AI*, vol. 3, p. 16, 2016.

[5] E. Simetti and G. Casalino, "A novel practical technique to integrate inequality control objectives and task transitions in priority based control," *Journal of Intelligent & Robotic Systems*, vol. 84, no. 1, pp. 877–902, Dec 2016.

[6] H. Hanafusa, T. Yoshikawa, and Y. Nakamura, "Analysis and control of articulated robot arms with redundancy," *IFAC Proceedings Volumes*, vol. 14, no. 2, pp. 1927 – 1932, 1981, 8th IFAC World Congress on Control Science and Technology for the Progress of Society, Kyoto, Japan, 24-28 August 1981.

[7] 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, 1987.

[8] B. Siciliano and J. J. Slotine, "A general framework for managing multiple tasks in highly redundant robotic systems," *Fifth International Conference on Advanced Robotics 'Robots in Unstructured Environments*, pp. 1211–1216 vol.2, 1991.

[9] S. Chiaverini, "Singularity-robust task-priority redundancy resolution for real-time kinematic control of robot manipulators," *Robotics and Automation, IEEE Transactions on Robotics and Automation*, vol. 13, pp. 398 – 410, 07 1997.

[10] G. Antonelli, F. Arrichiello, and S. Chiaverini, "The nsb control: a behavior-based approach for multi-robot systems," *Paladyn*, vol. 1, no. 1, pp. 48–56, Mar 2010.

[11] ——, "Stability analysis for the null-space-based behavioral control for multi-robot systems," pp. 2463–2468, Dec 2008.

[12] 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.

[13] 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," *Int. J. Rob. Res.*, vol. 10, no. 4, pp. 410–425, Jul. 1991.

[14] O. Khatib, "Real-time obstacle avoidance for manipulators and mobile robots," in *Proceedings. 1985 IEEE International Conference on Robotics and Automation*, vol. 2, March 1985, pp. 500–505.

[15] Y. Koren and J. Borenstein, "Potential field methods and their inherent limitations for mobile robot navigation," in *Proceedings. 1991 IEEE International Conference on Robotics and Automation*, April 1991, pp. 1398–1404 vol.2.

[16] 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, Aug 2011.

[17] S. Moe, J. T. Gravdahl, and K. Y. Pettersen, "Set-based control for autonomous spray painting," *IEEE Transactions on Automation Science and Engineering*, vol. 15, no. 4, pp. 1785–1796, Oct 2018.

[18] P. D. Lillo, F. Arrichiello, G. Antonelli, and S. Chiaverini, "Safety-related tasks within the set-based task-priority inverse kinematics framework," pp. 6130–6135, Oct 2018.

[19] G. Antonelli, *Underwater Robots*, ser. Springer Tracts in Advanced Robotics. Springer International Publishing, 2018.

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

[21] O. Khatib, "A unified approach for motion and force control of robot manipulators: The operational space formulation," *IEEE Journal on Robotics and Automation*, vol. 3, no. 1, pp. 43–53, February 1987.

[22] L. Sentis and O. Khatib, "Prioritized multi-objective dynamics and control of robots in human environments," vol. 2, pp. 764–780 Vol. 2, Nov 2004.

[23] ——, "A whole-body control framework for humanoids operating in human environments," pp. 2641–2648, May 2006.

[24] L. Sentis, "Synthesis and control of whole-body behaviors in humanoid systems," Ph.D. dissertation, Standford University, Stanford, CA, USA, 2007, aAI3281945.

[25] A. Dietrich, C. Ott, and A. Albu-Schäffer, "An overview of null space projections for redundant, torque-controlled robots," *The International Journal of Robotics Research*, vol. 34, no. 11, pp. 1385–1400, 2015.

[26] G. Antonelli, P. Di Lillo, and C. Natale, "Modeling errors analysis in inverse dynamics approaches within a task-priority framework," 08 2018, pp. 553–558.

[27] N. Mansard, O. Khatib, and A. Kheddar, "A unified approach to integrate unilateral constraints in the stack of tasks," *IEEE Transactions on Robotics*, vol. 25, no. 3, pp. 670–685, June 2009.

[28] B. Siciliano and O. Khatib, *Springer Handbook of Robotics*, 2nd ed. Springer Publishing Company, Incorporated, 2016.

[29] A. Liegeois, "Liegeois, a.: Automatic supervisory control of the configuration and behavior of multibody mechanisms. ieee trans. syst. man cybern. 7(12), 868-871," vol. 7, pp. 868–871, 12 1977.

[30] 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, Jan 2008.

[31] L. Sciavicco, B. Siciliano, and B. Sciavicco, *Modelling and Control of Robot Manipulators*, 2nd ed. Berlin, Heidelberg: Springer-Verlag, 2000.

[32] H. Khalil, *Nonlinear Systems*, ser. Pearson Education. Prentice Hall, 2002.

[33] H. Sadeghian, L. Villani, M. Keshmiri, and B. Siciliano, "Multi-priority control in redundant robotic systems," pp. 3752–3757, Sept 2011.

[34] E. Simetti, G. Casalino, F. Wanderlingh, and M. Aicardi, "Task priority control of underwater intervention systems: Theory and applications," *Ocean Engineering*, vol. 164, pp. 40 – 54, 2018.

[35] M. Spong, S. Hutchinson, and M. Vidyasagar, *Robot Modeling and Control*. Wiley, 2005.

[36] P. Hsu, J. Hauser, and S. Sastry, "Dynamic control of redundant manipulators," in *1988 American Control Conference*, June 1988, pp. 2135–2139.

[37] G. Golub and C. Van Loan, *Matrix Computations*, ser. Johns Hopkins Studies in the Mathematical Sciences. Johns Hopkins University Press, 2013.

[38] H. Schmidt-Didlaukies, "Modeling of articulated underwater robots for simulation and control," NTNU, Tech. Rep., 2018.

[39] J. Farrell, *Aided Navigation: GPS with High Rate Sensors*, 1st ed. New York, NY, USA: McGraw-Hill, Inc., 2008.

[40] T. I. Fossen, *Handbook of Marine Craft Hydrodynamics and Motion Control*. John Wiley & Sons, Ltd, 2011.

[41] R. Murray, Z. Li, S. Sastry, and S. Sastry, *A Mathematical Introduction to Robotic Manipulation*. Taylor & Francis, 1994.

[42] J. Nakanishi, R. Cory, M. Mistry, J. Peters, and S. Schaal, "Operational space control: A theoretical and empirical comparison," *The International Journal of Robotics Research*, vol. 27, no. 6, pp. 737–757, 2008.

[43] M. Wrzos-Kaminska, "Path following control for underwater swimming manipulators moving in 3d," Master's thesis, NTNU, June 2018.

[44] T. A. Johansen and T. I. Fossen, "Control allocation - a survey," *Automatica*, vol. 49, pp. 1087–1103, 2013.

[45] G. Marani, J. Kim, J. Yuh, and W. K. Chung, "A real-time approach for singularity avoidance in resolved motion rate control of robotic manipulators," in *ICRA*, 2002.

[46] A. E. Stene, "Operational space singularity avoidance for articulated intervention auv," NTNU, Tech. Rep., 2018.

[47] V. K. Nguyen, "Partial derivative of matrix functions with respect to a vector variable," *Vietnam Journal of Mechanics*, vol. 30, 07 2012.

[48] H. Sadeghian, L. Villani, M. Keshmiri, and B. Siciliano, "Dynamic multi-priority control in redundant robotic systems," *Robotica*, vol. 31, no. 7, p. 1155–1167, 2013.

[49] J. Sverdrup-Thygeson, S. Moe, K. Y. Pettersen, and J. T. Gravdahl, "Kinematic singularity avoidance for robot manipulators using set-based manipulability tasks," pp. 142–149, Aug 2017.

[50] D. R. Yoerger, J. G. Cooke, and J. . E. Slotine, "The influence of thruster dynamics on underwater vehicle behavior and their incorporation into control system design," *IEEE Journal of Oceanic Engineering*, vol. 15, no. 3, pp. 167–178, July 1990.

[51] A. Dietrich, C. Ott, and J. Park, "The hierarchical operational space formulation: Stability analysis for the regulation case," *IEEE Robotics and Automation Letters*, vol. 3, no. 2, pp. 1120–1127, April 2018.

[52] S. Moe, "Guidance and control of robot manipulators and autonomous marine robots," Ph.D. dissertation, NTNU, 2016.

[53] T. D. Barfoot, *State Estimation for Robotics*. Cambridge University Press, 2017.