Master's thesis

NTNU
Norwegian University of Science and Technology
Faculty of Information Technology and Electrical Engineering
Department of Engineering Cybernetics

Arnt Erik Stene

Robust Control for Articulated Intervention AUVs in the Operational Space

Master's thesis in Cybernetics and Robotics
Supervisor: Kristin Y. Pettersen
June 2019

NTNU
Norwegian University of
Science and Technology

Arnt Erik Stene

Robust Control for Articulated Intervention AUVs in the Operational Space

Master's thesis in Cybernetics and Robotics
Supervisor: Kristin Y. Pettersen, Ida-Louise G. Borlaug, Anna M. Kohl
June 2019

Norwegian University of Science and Technology
Faculty of Information Technology and Electrical Engineering
Department of Engineering Cybernetics

Task description for a Masters thesis

Anna Kohl, Ida-Louise Borlaug

In todays technology there is a large potential for improving efficiency and reducing costs by increasing autonomy. Subsea inspection, maintenance and repair (IMR) operations is a field of technology with a large potential for autonomous robotics to increase the efficiency, reduce operation costs, and provide safer and more environmentally friendly solutions. This requires the development of new systems for IMR operations that are more robust, agile, and versatile than existing technology. Articulated intervention autonomous underwater vehicles (AIAUVs) have emerged from swimming snake robots are essentially snake robots equipped with thrusters. AIAUVs are considered promising to provide autonomous IMR solutions in the future.

The operational space formulation (OSF) was introduced in the 1980s and provides a strategy for computing the control torque based directly on the desired dynamics of the end-effector [Khatib, 1987]. For redundant systems several tasks can be achieved by choosing the control forces in a way that the lower priority task forces act in the dynamically consistent nullspace of the higher priority task Jacobians, a strategy that is conceptually similar to task-priority inverse kinematic control. The advantage of the OSF is that the dynamics are taken into account already in the redundancy resolution of the robot and that it is easily extendable for contact interaction and compliant control.

Task

One of the major challenges however is the dependency on model parameters that are not accurately known. The task of the Master thesis is to investigate robust control in the OSF, namely sliding mode control methods. A second possible alternative are adaptive methods.

- Literature review: Review and describe the following methods from the literature
  1. OSF: [Khatib, 1987], [Antonelli et al., 2018], [Dietrich et al., 2015], [Dietrich et al., 2018] and references therein
  2. Sliding mode control: basic theory: [Shtessel et al., 2014], stability theory: [Polyakov and Fridman, 2014], applied to AIAUVs: [Borlaug et al., 2018], [Borlaug et al., 2019]
  3. Sliding mode control in the OSF: [Herrmann et al., 2016], [Barbalata et al., 2018]
  4. optional: Adaptive control in the OSF: [Tee and Yan, 2011], [Lu and Liu, 2017]
- Control design: Design a control system for multiple-task control of an AIAUV using the methods from the literature above. What assumptions have to be made regarding the model parameters and control gains?
- Simulation study: Define some benchmark scenarios that are relevant for AIAUVs and demonstrate the performance of the algorithm(s). Discuss the advantages and challenges. A simulation environment for the free-floating robot in Matlab/Simulink is available. The simulation model can be extended to take into account contact interaction using the model in [Antonelli, 2014], similar to the way it was done in [Aalbu, 2018].

References

[Aalbu, 2018] Aalbu, S. V. (2018). Subsea inspection and intervention with underwater swimming manipulators. Master's thesis, NTNU.

[Antonelli, 2014] Antonelli, G. (2014). Underwater robots. In *Springer Tracts in Advanced Robotics*, volume 96. Springer International Publishing, 3 edition.

[Antonelli et al., 2018] Antonelli, G., Lillo, P. D., and Natale, C. (2018). Modeling errors analysis in inverse dynamics approaches within a task-priority framework. In *Proc. IEEE Conf. Control Technology and Applications*, Copenhagen, Denmark.

[Barbalata et al., 2018] Barbalata, C., Dunnigan, M. W., and Petillot, Y. (2018). Position/force operational space control for underwater manipulation. *Robotics and Autonomous Systems*, 100:150–159.

[Borlaug et al., 2018] Borlaug, I.-L. G., Pettersen, K. Y., and Gravdahl, J. T. (2018). Trajectory tracking for an articulated intervention AUV using a super-twisting algorithm in 6 DOF. In *11th IFAC Conference on Control Applications in Marine Systems, Robotics, and Vehicles (Accepted)*, Opatija, Croatia.

[Borlaug et al., 2019] Borlaug, I.-L. G., Pettersen, K. Y., and Gravdahl, J. T. (2019). Tracking control of an articulated intervention-AUV in 6DOF using the generalized super-twisting algorithm. In *Submitted to 2019 American Control Conference*, Philadelphia, USA.

[Dietrich et al., 2015] Dietrich, A., Ott, C., and Albu-Schäffer, A. (2015). An overview of null space projections for redundant, torque-controlled robots. *The International Journal of Robotics Research*, 34(11):1385–1400.

[Dietrich et al., 2018] Dietrich, A., Ott, C., and Park, J. (2018). The Hierarchical Operational Space Formulation: Stability Analysis for the Regulation Case. *IEEE Robotics and Automation Letters*, 3(2):1120–1127.

[Herrmann et al., 2016] Herrmann, G., Jalani, J., Mahyuddin, M. N., Khan, S. G., and Melhuish, C. R. (2016). Robotic hand posture and compliant grasping control using operational space and integral sliding mode control. *Robotica*, 34(10):2163–2185.

[Khatib, 1987] Khatib, O. (1987). A unified approach for motion and force control of robot manipulators: The operational space formulation. *Robotics and Automation, IEEE Journal of*, 3(1):43–53.

[Lu and Liu, 2017] Lu, W. and Liu, D. (2017). Active Task Design in Adaptive Control of Redundant Robotic Systems. In *Proc. Australasian Conf. Robotics and Automation 2017*.

[Polyakov and Fridman, 2014] Polyakov, A. and Fridman, L. (2014). Stability notions and lyapunov functions for sliding mode control systems. *Journal of the Franklin Institute*, 351(4):1831–1865.

[Shtessel et al., 2014] Shtessel, Y., Edwards, C., Fridman, L., and Levant, A. (2014). *Sliding Mode Control and Observation*. Control Engineering. Springer.

[Tee and Yan, 2011] Tee, K. P. and Yan, R. (2011). Adaptive Operational Space Control of Redundant Robot Manipulators. In *Proc. American Control Conference*, San Francisco, CA.

Abstract

This thesis examines robust control methods applied within the operational space control formulation. The methods are applied to an Articulated Intervention Autonomous Underwater Vehicle (AIAUV) which require robust control to counteract uncertainties in hydrodynamic and hydrostatic parameters, unknown external disturbances, like waves and currents, and modeling errors. The AIAUV is a hyper-redundant robot manipulator which has shown great promise for efficient and cost effective operations in subsea inspection, maintenance and repair.

The thesis is focused on sliding mode controllers (SMC), where a simple PID-controller is implemented as reference and compared to the higher order, nonlinear super-twisting algorithm with adaptive gains (STA) and the generalized-super-twisting algorithm (GSTA). These methods include an integrator which cloaks the non-continuous high-frequency switching of the sliding mode control approach and ensures a continuous control signal.

The operational space formulation incorporates multiple task-priority control, which allows exploitation of the redundant degrees of freedom of the AIAUV for intuitive design of different control tasks. The thesis tackles different control tasks, namely end-effector position and orientation; joint angles; and the actuation index. The latter is a result of the specialization project and is further examined for actuator singularity avoidance in this thesis.

A simulation study is performed in realistic scenarios which the AIAUV might face at subsea. The SMCs perform on par with the PID-controller with compensation, proving they are well suited robust controllers for the AIAUV. In conclusion, the operational space formulation is well suited to implement task priorities for the AIAUV, where transient performance and stationary convergence with the robust controllers is shown through simulations. Finally, the actuation index task displays increased performance with the more advanced controllers.

Sammendrag

Denne avhandlingen undersøker robuste kontrollmetoder innenfor *operational space* formuleringen. Metodene er brukt på et artikulert intervensjons undervannskjøretøy (AIAUV) som krever robust kontroll for å motvirke usikkerheter i hydrodynamiske og hydrostatiske parametere, ukjente eksterne forstyrrelser, som bølger og strømmer, og modelleringsfeil. AIAUV er en hyper-redundant robotmanipulator som har vist potensiale for kostnadsbesparende og effektiv drift i undersjøisk inspeksjon, vedlikehold og reparasjon.

Prosjektet fokuserer på *sliding mode* kontrollere (SMC), hvor en enkel PID-regulator er implementert som referanse og sammenlignet med høyere-ordens, ulineære *super-twisting* med adaptive ledd (STA) og *generalized super-twisting* (GSTA). Disse metodene inkluderer en integrator som skjuler den ikke-kontinuerlige høyfrekvente vekslingen i kontrollsignalet som er typisk for *sliding mode* kontroll og sikrer et kontinuerlig kontrollsignal.

Operational space formuleringen muliggjør kontroll av flere oppgaver med ulik prioritet, slik at de redundante frihetsgradene i AIAUVen kan utnyttes til å intuitivt utforme forskjellige kontrolloppgaver. Avhandlingen fokuserer på tre ulike kontrolloppgaver: Styring av slangens hodeposisjon og rotasjon; vinklene i rotasjonsleddene; og *actuation index*. Sistnevnte er et resultat fra prosjektoppgaven, og blir videre brukt til å unngå singulariteter som en kontrolloppgave i simuleringer.

For å undersøke ytelsen til kontrollerne utføres en simuleringsstudie i relistiske scenarier som AIAUVen kan møte på undersjøiske installasjoner. *Sliding mode* kontrollerne presterer på samme nivå som PID-kontrolleren med kompensasjon, noe som viser at de er velegnet for robust kontroll på en AIAUV. For å konkludere, *operational space* formuleringen er velegnet til å implementere prioriteter for flere oppgaver for en AIAUV, hvor transient ytelse og stasjonær konvergens med *sliding mode* kontrollerne er vist gjennom simuleringer. I tillegg viser *actuation index* økt ytelse når avanserte *sliding mode* kontrollere brukes.

Preface

This master's thesis is submitted as a part of the requirements for the M. Sc. degree at the Department of Engineering Cybernetics at the Norwegian University of Science and Technology. The project has been completed with guidance from Prof. Kristin Y. Pettersen, PhD. candidate Ida-Louise Garmann Borlaug and Dr. Anna M. Kohl at the Department of Engineering Cybernetics, NTNU.

I chose the topic of underwater snake robots as the subject for my thesis without much prior knowledge of the subject, but I found it fascinating, being an extensive field of research at the Department of Engineering Cybernetics. As such, I would like to thank my supervisor Kristin Y. Pettersen for giving me the opportunity to work with this subject which I have had a lot of fun with this year. I would also like to thank Ida-Louise G. Borlaug and Anna M. Kohl for meeting with me bi-weekly since August and providing me much needed professional advise and insight, and for proof-reading this thesis. Through the year I have also received help from my colleagues, and a special thanks goes to Sondre Sortland and Bjørn A. Kristiansen for sparring with me on many occasions.

This master's thesis is a continuation of a specialization project I conducted during the autumn of 2018. As is customary, the specialization project is not published. This means important background theory and methods from the project report will be restated in full throughout this report to provide the best reading experience. Below, a complete list of the material included from the specialization project is listed.

- Chapter 2 (Specifically sections 2.1, 2.2.2 and 2.2.3, with some changes to section 2.2.3)
- Chapter 3
- Chapter 5

Contributions and related work will be presented in the introduction.

During the project, I have been provided multiple tools through Anna M. Kohl and Ida-Louise G. Borlaug to be used in the work. A problem description for the

master's thesis was provided which contained multiple sources of literature that have been important during the project. A simulator with the dynamic model implemented in Matlab created by Ph.D. candidate Henrik Schmidt-Didlaukies at the Department of Marine Technology, NTNU and extended to single-task end-effector control within the operational space by Anna M. Kohl, has been the basis for most of the work. Towards the end of the project the simulator base was replaced by a simulator implementing the dynamic model and reference trajectory generator in Matlab/Simulink, which was provided by Ida-Louise G. Borlaug.

For the specialization project completed in the Autumn of 2018, a problem description for actuator singularity avoidance was provided. In addition, a MATLAB script containing symbolic computations of the actuator configuration matrix derivative was also provided.

Unless otherwise stated, all figures and illustrations have been created by the author.

Arnt Erik Stene
Trondheim June 2, 2019

Contents

List of Figures

Nomenclature

Acronyms

USM	Underwater Swimming Manipulator
USR	Underwater Snake Robot
AUV	Autonomous Underwater Vehicle
AIAUV	Articulated Intervention Autonomous Underwater Vehicle
ROV	Remotely Operated Vehicle
SMC	Sliding Mode Control
STA	Super-Twisting Algorithm with Adaptive Gains
GSTA	Generalized Super-Twisting Algorithm
AME	Absolute Maximum Error
RMSE	Root-Mean-Square Error

Model notation

n	Number of links
n_q	Number of joints
n_t	Number of thrusters
m_x	Task dimension
$\boldsymbol{\nu}_{h/i}^{h}$	Linear and angular velocities in frame O_h w.r.t frame O_i expressed in O_i

$\boldsymbol{\zeta}_{h/i}^{h}$	System velocities in frame O_h w.r.t frame O_i expressed in O_i
$\boldsymbol{B}$	Actuator configuration matrix
$\boldsymbol{M}$	Rigid-body mass matrix
$\boldsymbol{C}$	Rigid-body coriolis and centripetal forces
$\boldsymbol{D}$	Hydrodynamic damping matrix
$\boldsymbol{g}$	Hydrostatic wrench
$\boldsymbol{J}$	System Jacobian matrix
$\boldsymbol{\tau}$	Torque vector

End-effector

$\boldsymbol{\eta}_{h/i}^{i}$	End-effector position and orientation expressed with quaternions in inertial frame
${}^{e}\boldsymbol{\eta}_{h/i}^{i}$	End-effector position and orientation expressed with Euler angles in inertial frame
$\boldsymbol{t}_{h/i}^{i}$	End-effector Cartesian coordinates in inertial frame
$\boldsymbol{p}_{ih}$	End-effector orientation in quaternions between frame O_i and O_h
$\boldsymbol{\eta}_{ih}$	Scalar part of quaternion between frame O_i and O_h
$\boldsymbol{\epsilon}_{ih}$	Vectorial part of quaternion between frame O_i and O_h
$\boldsymbol{\Theta}_{ih}$	End-effector orientation in Euler angles

Actuation index

σ	Actuation index coordinate
σ_d	Actuation index reference

Joint angles

$\boldsymbol{q}$	Joint angle coordinates
$\boldsymbol{q}_d$	Joint angle reference

Operational space

$\boldsymbol{J}_x$	Task specific Jacobian for task x

$\boldsymbol{M}_x$	Operational space mass matrix
$\boldsymbol{c}_x$	Operational space coriolis and centripetal forces
$\boldsymbol{d}_x$	Operational space hydrodynamic damping
$\boldsymbol{g}_x$	Operational space hydrostatic wrench
$\boldsymbol{\tau}_x$	Task specific torque
$\boldsymbol{N}_x$	Task specific null-space projector
$\bar{\boldsymbol{J}}_x$	Task specific weighted pseudoinverse
$\boldsymbol{A}^+$	Moore Penrose pseudoinverse

PID

$\boldsymbol{K}_p$	PID proportional gain
$\boldsymbol{K}_d$	PID derivative gain
$\boldsymbol{K}_i$	PID integral gain
$K_{dis,x}$	PID dissipative gain

Super-twisting algorithm

u_{STA}	Super-twisting with adaptive gains computed force
$\boldsymbol{\beta}_{STA}$	STA adaptive controller gain
$\boldsymbol{\alpha}$	STA adaptive controller gain
$\boldsymbol{\epsilon}_1$	STA controller gain
$\boldsymbol{\omega}_1$	STA controller gain
$\boldsymbol{\lambda}_1$	STA controller gain
$\boldsymbol{\gamma}_1$	STA controller gain

Generalized super-twisting algorithm

u_{GSTA}	Generalized super-twisting computed force
$\boldsymbol{\beta}_{GSTA}$	GSTA controller gain
$\boldsymbol{\phi}_1$	GSTA controller gain
$\boldsymbol{\phi}_2$	GSTA controller gain

Chapter 1

Introduction

This chapter gives an introduction to the remainder of the thesis. First, motivation for the research is given, before related work is presented in multiple categories. These sections are then summarized in the thesis goal, before the thesis contributions are presented. Finally, an outline of the thesis is given.

1.1 Motivation

In the oil and gas industry, large subsea installations have become more common, and fully-automated plants are within reach. In order to perform inspection, maintenance and repair, the traditional remotely operated vehicle (ROV) has been used. The ROV is relatively bulky, expensive to operate and requires human intervention during operation. To perform reliable, real-time, dexterous inspection and manipulation at subsea a new solution is required.

The Articulated Intervention Autonomous Underwater Vehicle (AIAUV), also called underwater swimming manipulator (USM), introduced in [1] is a hyper-redundant robot manipulator based on biological snakes, and has great potential for efficient and cost-effective operation. Compared to the ROV, the robot is cheap to operate, dexterous and requires no human intervention during operations. In addition, the AIAUV is intended for autonomous operations, which further reduces cost and increases efficiency.

If the AIAUV is to see success and contribute to cost-efficient operations, ad-

vanced control techniques are required to take advantage of the redundant degrees of freedom. Control algorithms for hyper-redundant manipulators are often based on inverse-kinematic or inverse-dynamic approaches. The inverse-kinematic approach generates reference trajectories for the overall pose of the manipulator without taking into account the dynamics of the robot. Inverse-dynamic approaches, e.g. the operational space formulation [2], create a direct mapping from computed forces to joint and thruster torques based on the desired dynamics of the end-effector. The advantage of the operational space formulation is that the dynamics are taken into account already in the redundancy resolution of the manipulator. The operational space formulation also implements task priorities for the control system, enabling exploitation of the redundant degrees of freedom for the AIAUV and a hierarchical control structure.

One of the major challenges in control of the AIAUV and other robot applications, is the dependency on parameters which are not accurately known. For example, the AIAUV is subject to hydrodynamic and hydrostatic parameter uncertainties, in addition to unknown disturbances present at subsea. To negate the effects of parameter uncertainties, robust control methods are required.

This thesis aims to investigate robust control methods applied within the operational space formulation. Main focus is given to sliding mode control, but a PID-controller is implemented for reference. In addition, the thesis builds on the specialization project, where a scheme for avoiding singularities in the actuator configuration matrix was proposed. The specialization project pointed out the need for more advanced control methods, and some of the simulations in this thesis will examine the effects on the singularity avoidance scheme.

1.2 Related work

This section will introduce references to related work of the thesis. First, the history of the Autonomous Underwater Vehicle (AUV) and its development towards the AIAUV used in this thesis is given. Then, some background to robust control theory is presented with emphasis on the control methods used in this thesis. In addition, important resources for the implementation of the singularity avoidance scheme in the specialization project are introduced.

1.2.1 AIAUV

AUVs have been studied since SPURV [3] was developed in 1957 and are still an extensive field of research. There has been a peak of interest lately with possible applications at subsea as an enhancement to human operated ROVs. Introducing autonomous technology leads to increased efficiency and reduced costs for the oil and gas industry, and the potential for AUVs is considerable. Currently in the industry, AUVs are used for surveys of larger areas, they are torpedo shaped and without a tether. An example is Kongsberg's AUV, HUGIN [4].

Another area in robotics which has been shown interest in recent years is the development of biologically inspired snake robots. The robot creates forward propulsion through snake-like locomotion and is able to access remote and difficult locations. Shigeo Hirose introduced the first terrestrial snake robot in the 1970s [5] and since, several new application areas have been proposed. In [6], a snake robot was used as a firefighting robot which could access areas not accessible by human firefighters. Modular terrestrial snake robots have been studied by Professor Howie Choset and his research group[1] at the Carnegie Mellon University. Professor Auke Jan Ijspeert and his research group at the EPFL Biorobotics laboratory study biologically inspired amphibious robots based on snakes, salamanders, fish and centipedes[2]. The snake robot has also evolved into an AUV and is conquering the oceans through the Underwater Snake Robot (USR), where additional propulsion can be achieved through propellers. A detailed description of the USR Mamba can be found in [7].

To perform more than mere inspection, which would be required in order to compete with or replace ROVs in the future, the AUV requires a way to interact and apply forces to the environment. For example, the SAUVIM [8] has an attached manipulator arm in order to perform intervention missions. The AIAUV presented in section 1.1 has the potential of using its own body as a lever for intervention purposes, and could be a viable alternative to the ROV.

1.2.2 Controller synthesis

As mentioned in section 1.1, inverse-kinematics and inverse-dynamics are often used for manipulator control schemes. Inverse-kinematics has been used to generate reference trajectories for AIAUVs before, e.g. [9]–[11]. However, the inverse-dynamics scheme in the operational space formulation [2] has not

[1] http://biorobotics.ri.cmu.edu/projects/modsnake/index.html
[2] https://biorob.epfl.ch/research/research-amphibious/

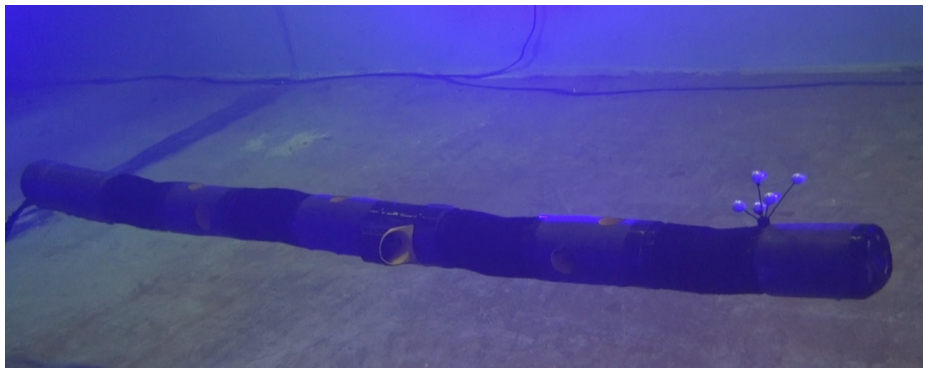

Figure 1.1: Eely AIAUV at NTNU Tyholt. Image provided by Anna M. Kohl.

previously been applied to control of an AIAUV, although it has been extensively researched and used with other systems. In [12] it was used for robotic hand posture and grasping control; in [13] it was applied to a 6DOF underwater manipulator for position and force control; in [14] a controller known as the whole-body operational space controller was used for stabilization of a point-foot bipedal robot. As can be seen in fig. 1.2, the inverse-dynamics scheme includes the dynamic controller, while the inverse-kinematic scheme assumes the dynamic controller is able to follow the generated reference perfectly.

To counteract uncertainties in parameters, modeling and disturbances several robust control methods have been developed. The predominant robust controller frameworks are adaptive controllers and sliding mode control [15]. A thorough introduction to robust adaptive control can be found in [16], and an intuitive coverage of SMC is given in [17]. Adaptive control in the operational space formulation has previously been performed in [18] and [19]. SMC has already been applied to an AIAUV in [10], [11] and [20]. The generalized super-twisting algorithm (GSTA) from [21] and [22] is applied on the AIAUV in [11]. A combination of adaptive control and SMC was used on the AIAUV in [10] and [20]; namely the super-twisting algorithm with adaptive gains (STA) [23]. In [10], [11], the inverse-kinematics approach was used to generate a reference for the controllers to follow, while in [20], the controller was fed a generated trajectory. Contrary to these papers, this thesis aims to use the SMCs within the operational space formulation.

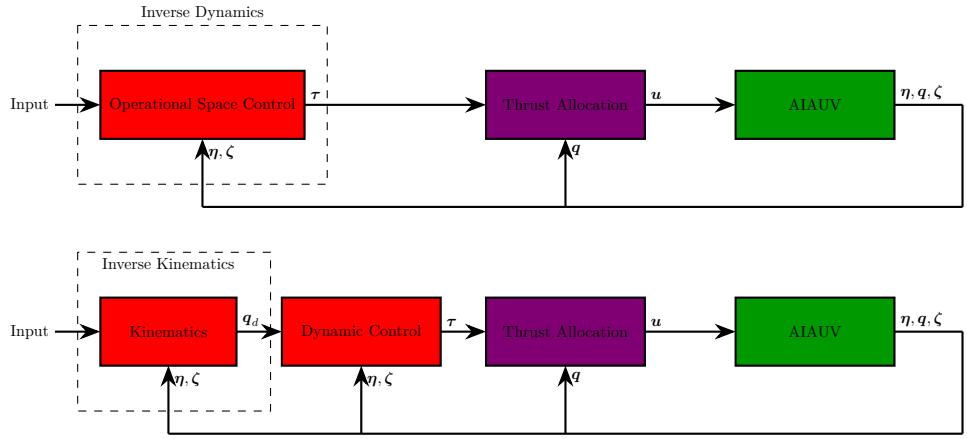

Figure 1.2: Inverse-kinematics vs. inverse-dynamics control scheme.

1.2.3 Actuator singularity avoidance

The term *singularity* is used in many different contexts, be it as the center of a black hole or the future super-intelligent artificial intelligence. In this thesis, the term denotes the configurations of the manipulator in which a singularity arises. A singularity is a configuration in which the manipulator looses one or more degrees of freedom, meaning it is unable to exert force in one or more directions. The AIAUV may be affected by kinematic and actuator based singularities. These singularities arise when the manipulator Jacobian and actuator configuration matrix are singular, i.e. they do not have full rank, respectively.

Kinematic singularity avoidance has been performed in several studies, where [9] and [24] used the manipulability index from [25] as a measure of the distance the system is from a singularity. In [9], the manipulability index was implemented as a high-priority set-based task and the method guarantees that the manipulator maintains a configuration with high joint dexterity. In addition, compatible lower priority equality tasks will still converge asymptotically. In [26], it was pointed out that the actuator configuration matrix may become singular as the desired control torque is distributed among the actuators. The task enforcing the singularity avoidance in [9] was implemented using the set-based task framework presented in [27].

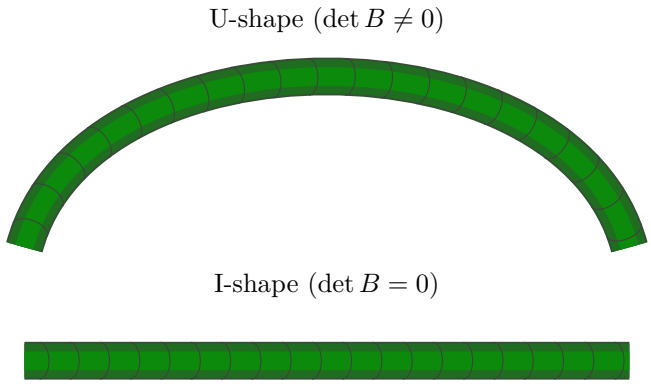

Figure 1.3: Example of singular configuration for the Eely robot used in simulations. When $\det B = 0$ the AIAUV is in a singular configuration.

1.3 Thesis goal and contributions

1.3.1 Goal

This thesis aims to apply the sliding mode control methods presented in [22] and [23] within the operational space formulation [2] to perform robust control on the AIAUV presented in [1]. The methods will then be compared to a PID-controller with compensation extended from the PD-controller in [28] and the results will be evaluated using several simulated scenarios. Furthermore, the actuator singularity avoidance scheme developed in the specialization project will be further analyzed through application of the advanced control methods.

1.3.2 Contributions

The main contribution of this thesis is the analysis of robust higher-order sliding mode controllers in the operational space applied to an AIAUV. This thesis applies the STA and GSTA controllers within the operational space formulation, which, to the author's knowledge, has not been done before.

Furthermore, the operational space formulation has not previously been used with the AIAUV manipulator framework. An important point for the use of the AIAUV is the redundant degrees of freedom, which through the operational space may be exploited by implementing multiple tasks in a prioritized manner.

As such, the operational space framework may be important for future work with the AIAUV platform. This thesis examines the performance of the operational space multiple task-priority framework for the AIAUV and shows that it is well suited for task-priority control on the platform.

The actuation index task, that was first derived in the specialization project, is presented in this thesis. The actuation index will serve as an important use case in part III of this thesis, where the robust control methods are demonstrated in simulations.

1.4 Outline

Part I — Background Theory

Part I provides the background material required to understand the work presented in parts II and III.

Chapter 2 — Snake Robot Model: This chapter presents the simulation model and conventions for the general AIAUV. In addition, the physical parameters of the Eely snake robot manipulator used in simulations in part III are given.

Chapter 3 — Operational Space Formulation: An introduction to the inverse dynamics operational space formulation is given in this chapter. This framework implements task priorities for the control system, enabling exploitation of the redundant degrees of freedom for the AIAUV.

Chapter 4 — Sliding Mode Control: Sliding mode control is a robust control method which can compensate for uncertainties in the model and in uncertain model parameters. This chapter gives an introduction to SMC on a basic level, before presenting two higher-order SMC used in this thesis.

Part II — Robust Control and Singularity Avoidance

Part II focuses on the implementation of the methods presented in part I. In addition, the actuation index examined in the specialization project is derived since it provides a use case for the simulation examples in part III of this thesis.

Chapter 5 — Actuator Singularity Avoidance: This chapter restates the derivations of the actuation index task for actuator singularity avoidance done in the specialization project.

Chapter 6 — Control Tasks: First, the three control tasks used for simulations in part III are derived. These are the end-effector position and orientation expressed with quaternions, the manipulator joint angles and the actuation index. A summary of the derivations of the actuation index in chapter 5 is given because the associated control task serves as a simulation example in this thesis.

Chapter 7 — Sliding Mode Control: The implementation of the sliding mode control methods described in chapter 4 is examined, and all necessary control gains used in simulations for the controllers are stated.

Chapter 8 — PID Control: The PID-controller torque equation is derived and the controller gains used for the PID-controller in all three tasks are presented.

Part III — Simulation

Chapter 9 — Setup: A brief description of the Matlab/Simulink simulator used for simulation is given before the reference trajectory generation is discussed.

Chapter 10 — Scenario I: Low-priority Task Convergence: The operational space formulation supports separating control tasks into different priorities, where the higher priority task might interfere with the convergence of the lower priority task. This scenario examines the convergence of the low-priority task whenever a higher priority task is active.

Chapter 11 — Scenario II: Configuration Maneuver: The manipulator requires a different set of configurations to perform a variety of tasks. The configuration is largely controlled by the angles of the joints, where a U-shape is often desired as the most dexterous configuration. This scenario examines the controller's ability to achieve different manipulator configurations.

Chapter 12 — Scenario III: Online Singularity Avoidance: The performance of the control methods are examined when used for online singularity avoidance with the actuation index task. Furthermore, the validity of the singularity avoidance method from the specialization project is discussed.

Chapter 13 — Analysis: A summary of the results from the simulations and some observations are made.

Chapter 14 — Conclusions and Future Work: Finally, the conclusion of the thesis is given before suggestions for future work is presented.

Part I

Background Theory

Chapter 2

Snake Robot Model

Remark: Some of the theory presented in this chapter is the same as was presented in the specialization project, but is included here for completeness. (Specifically, section 2.1, section 2.2.2(with some changes) and section 2.2.3.

This chapter presents the snake robot model derived in [29], along with background theory and notation required to understand the model equations. The notation is also highly relevant for the remainder of this thesis, where it is used in most controller and task derivations. Finally, the Eely, which is the real AIAUV the simulations are based on, is presented with physical parameters. An introduction to manipulator kinematics is outside the scope of this thesis, but the interested reader can refer to [30] for more background information.

2.1 The Jacobian

The forward kinematics of a manipulator describes the mapping from joint parameters to Cartesian positions and orientations. The relationship between joint velocities and velocities in Cartesian coordinates is then described by the

Jacobian of this mapping function. For a general function $\mathbf{y}$

$$\mathbf{y} = \begin{bmatrix} y_1 \\ y_2 \\ y_3 \\ \vdots \\ y_n \end{bmatrix} = \begin{bmatrix} f_1(\mathbf{x}) \\ f_2(\mathbf{x}) \\ f_3(\mathbf{x}) \\ \vdots \\ f_n(\mathbf{x}) \end{bmatrix} = \mathbf{f}(\mathbf{x}) \tag{2.1}$$

where $\mathbf{x} = \begin{bmatrix} x_1 & x_2 & x_3 & \cdots & x_n \end{bmatrix}$, the Jacobian is given as [30]

$$\mathbf{J}(\mathbf{x}) = \begin{bmatrix} \frac{\partial f_1}{\partial x_1} & \frac{\partial f_1}{\partial x_2} & \frac{\partial f_1}{\partial x_3} & \cdots & \frac{\partial f_1}{\partial x_n} \\ \frac{\partial f_2}{\partial x_1} & \frac{\partial f_2}{\partial x_2} & \frac{\partial f_2}{\partial x_3} & \cdots & \frac{\partial f_2}{\partial x_n} \\ \frac{\partial f_3}{\partial x_1} & \frac{\partial f_3}{\partial x_2} & \frac{\partial f_3}{\partial x_3} & \cdots & \frac{\partial f_3}{\partial x_n} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ \frac{\partial f_n}{\partial x_1} & \frac{\partial f_n}{\partial x_2} & \frac{\partial f_n}{\partial x_3} & \cdots & \frac{\partial f_n}{\partial x_n} \end{bmatrix} \tag{2.2}$$

In the special case $y = f_1(\mathbf{x})$ where y is scalar, the Jacobian is

$$\mathbf{J}(\mathbf{x}) = \begin{bmatrix} \frac{\partial f_1}{\partial x_1} & \frac{\partial f_1}{\partial x_2} & \frac{\partial f_1}{\partial x_3} & \cdots & \frac{\partial f_1}{\partial x_n} \end{bmatrix} \tag{2.3}$$

2.2 Snake robot model

This section begins by introducing the most relevant coordinate frame before a detailed explanation of notation used in the simulation model is given. Then, the model itself from [29] is presented. Finally, the equations for the Jacobians and derivatives in the model are given.

2.2.1 Coordinate frames

The AIAUV is a floating base manipulator, which means that the base is not fixed relative to the inertial frame. As a consequence, finding the thruster configuration matrix for the AIAUV is more complex since both the position and orientation of the thrusters relative to the base is dependent on the joint angles $\boldsymbol{q}$.

Figure 2.1 gives a visual representation of the frames defined for the AIAUV used in this thesis. The forward kinematics of the manipulator defines coordinate frames at every joint, but most parameters are given relative to inertial frame O_i, base frame O_b, or head frame O_h.

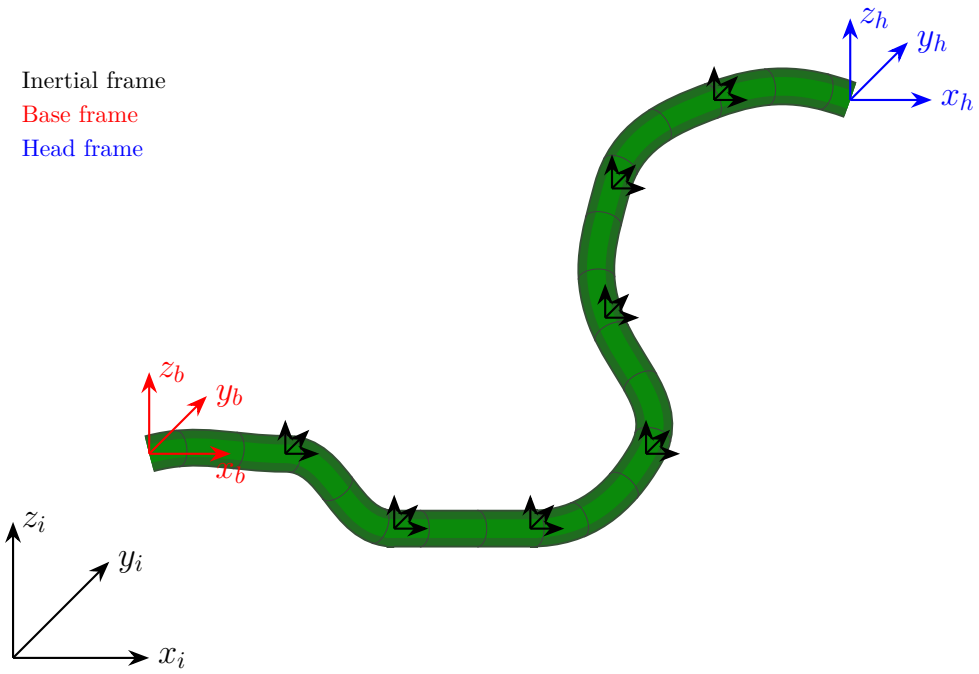

Figure 2.1: Coordinate frames for the AIAUV

2.2.2 Notation

The AIAUV is a serial-chain manipulator which consists of n rigid links connected by $n-1$ joints. Links are numbered $1 \cdots n$, while joints are numbered $1 \cdots n-1$. This means that link i is connected to link $i+1$ by joint i. Every joint angle is described in the joint vector

$$\boldsymbol{q} = \begin{bmatrix} q_1 & q_2 & \cdots & q_{n-1} \end{bmatrix}^T \in \mathbb{R}^{n-1} \tag{2.4}$$

The joint angles are visualized in fig. 2.2. The linear and angular velocities of the robot center of mass are combined into a vector known as the body twist vector.

$$\boldsymbol{\nu}^b_{b/i} = \begin{bmatrix} \left(\boldsymbol{v}^b_{b/i}\right)^T & \left(\boldsymbol{\omega}^b_{b/i}\right)^T \end{bmatrix}^T \in \mathbb{R}^6 \tag{2.5}$$

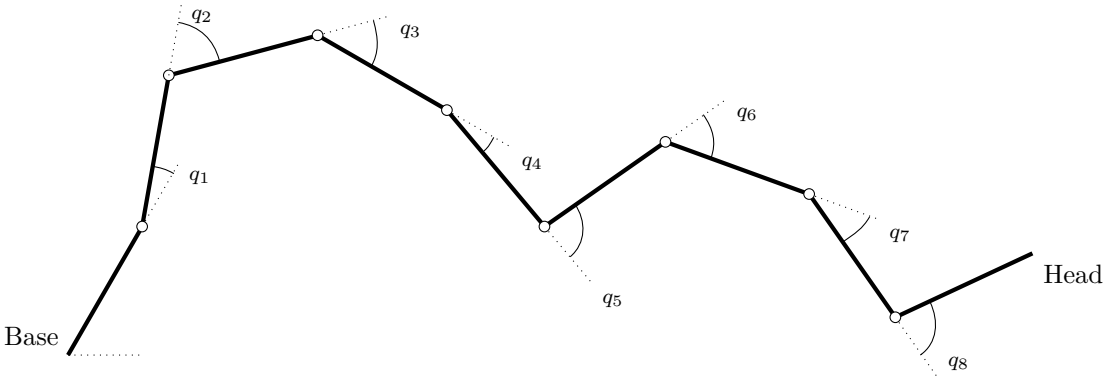

Figure 2.2: Snake robot joint angle configuration.

The body twists and joint velocities are then combined into a system velocity vector as

$$\boldsymbol{\zeta}_{b/i}^{b} = \begin{bmatrix} \left(\boldsymbol{\nu}_{b/i}^{b}\right)^T & \dot{\boldsymbol{q}}^T \end{bmatrix}^T \in \mathbb{R}^{n+5} \tag{2.6}$$

Position and orientation of the tail frame is described by $\boldsymbol{t}_{b/i}^{i} = \begin{bmatrix} x & y & z \end{bmatrix}^T$ and the quaternion $\boldsymbol{p}_{ib} = \begin{bmatrix} \eta_{ib} & \boldsymbol{\epsilon}_{ib} \end{bmatrix}^T$ through the vector

$$\boldsymbol{\eta}_{b/i}^{i} = \begin{bmatrix} \left(\mathbf{t}_{b/i}^{i}\right)^T & \boldsymbol{p}_{ib}^T \end{bmatrix}^T \in \mathbb{R}^7 \tag{2.7}$$

The vector in (2.7) is specified using quaternions, but sometimes specifying the orientation with Euler angles is more intuitive. Thus, the following notation will be used to express (2.7) with Euler angles.

$$^{e}\boldsymbol{\eta}_{b/i}^{i} = \begin{bmatrix} \left(^{e}\mathbf{t}_{b/i}^{i}\right)^T & ^{e}\boldsymbol{p}_{ib}^T \end{bmatrix}^T \in \mathbb{R}^6 \tag{2.8}$$

2.2.3 Simulation model

Below is the simulation model introduced in [29], where the joint and thruster torques have been combined through the actuator configuration matrix $\boldsymbol{B}(\boldsymbol{q})$.

$$\dot{\boldsymbol{\eta}}^i_{b/i} = \boldsymbol{T}(\boldsymbol{\eta}^i_{b/i})\boldsymbol{\nu}^b_{b/i} \tag{2.9a}$$

$$\boldsymbol{M}(\boldsymbol{q})\dot{\boldsymbol{\zeta}}^b_{b/i} + \boldsymbol{C}(\boldsymbol{q},\boldsymbol{\zeta}^b_{b/i})\boldsymbol{\zeta}^b_{b/i} + \boldsymbol{D}(\boldsymbol{q},\boldsymbol{\zeta}^b_{b/i})\boldsymbol{\zeta}^b_{b/i} + \boldsymbol{g}(\boldsymbol{\eta}^i_{b/i},\boldsymbol{q}) = \boldsymbol{B}(\boldsymbol{q})\boldsymbol{\tau} \tag{2.9b}$$

Where $\boldsymbol{M}(\boldsymbol{q})$ is the rigid-body mass matrix, $\boldsymbol{C}(\boldsymbol{q},\boldsymbol{\zeta}^b_{b/i})$ the rigid-body coriolis and centripetal forces, $\boldsymbol{D}(\boldsymbol{q},\boldsymbol{\zeta}^b_{b/i})$ the hydrodynamic damping matrix and $\boldsymbol{g}(\boldsymbol{\eta},\boldsymbol{q})$ is the hydrostatic wrench. The expression in (2.9a) is the forward kinematics of the snake robot. The mentioned matrices can be computed as [29]

$$\boldsymbol{M}(\boldsymbol{q}) = \sum_{i=1}^{n} \boldsymbol{J}_i(\boldsymbol{q})^T \boldsymbol{M}_i \boldsymbol{J}_i(\boldsymbol{q}) \tag{2.10a}$$

$$\boldsymbol{C}(\boldsymbol{q},\boldsymbol{\zeta}^b_{b/i}) = \sum_{i=1}^{n} \left[\boldsymbol{J}_i(\boldsymbol{q})^T \boldsymbol{M}_i \frac{d}{dt}\left(\boldsymbol{J}_i(\boldsymbol{q},\dot{\boldsymbol{q}})\right) - \boldsymbol{J}_i(\boldsymbol{q})^T \boldsymbol{W}_i(\boldsymbol{q},\boldsymbol{\zeta}^b_{b/i}) \boldsymbol{J}_i(\boldsymbol{q}) \right] \tag{2.10b}$$

$$\boldsymbol{D}(\boldsymbol{q},\boldsymbol{\zeta}^b_{b/i}) = \sum_{i=1}^{n} \boldsymbol{J}_i(\boldsymbol{q})^T \boldsymbol{D}_i(\boldsymbol{q},\boldsymbol{\zeta}^b_{b/i}) \boldsymbol{J}_i(\boldsymbol{q}) \tag{2.10c}$$

$$\boldsymbol{W}_i(\boldsymbol{q},\boldsymbol{\zeta}^b_{b/i}) = \begin{bmatrix} 0 & [\{\boldsymbol{M}_i\boldsymbol{\nu}_i\}_v \times] \\ [\{\boldsymbol{M}_i\boldsymbol{\nu}_i\}_v \times] & [\{\boldsymbol{M}_i\boldsymbol{\nu}_i\}_\omega \times] \end{bmatrix} \tag{2.10d}$$

At the core of the model, there are the Jacobians for each link, which can be found through a recursive relationship as

$$\boldsymbol{J}_1 = \begin{bmatrix} \boldsymbol{I} & \boldsymbol{0} & \cdots & \boldsymbol{0} \end{bmatrix} \tag{2.11a}$$

$$\boldsymbol{J}_{i+1} = Ad^{-1}\left(\boldsymbol{A}_i(q_i)\right)\boldsymbol{J}_i + \begin{bmatrix} 0 & \boldsymbol{0} & \cdots & \boldsymbol{0} & \boldsymbol{a}_i & \boldsymbol{0} & \cdots & \boldsymbol{0} \end{bmatrix} \tag{2.11b}$$

From differentiation of eqs. (2.11a) and (2.11b), the derivative of the Jacobian can be found as [29]

$$\frac{d}{dt}\left(\boldsymbol{J}_1\right) = \begin{bmatrix} 0 & \boldsymbol{0} & \cdots & \boldsymbol{0} \end{bmatrix} \tag{2.12a}$$

$$\frac{d}{dt}\left(\boldsymbol{J}_{i+1}\right) = ad(\boldsymbol{a}_i)\boldsymbol{J}_{i+1}\dot{q}_i + Ad^{-1}\left(\boldsymbol{A}_i(q_i)\right) \tag{2.12b}$$

2.2.4 Actuator allocation

The matrix describing the position and orientation of the actuators of the manipulator, known as the actuator configuration matrix $\boldsymbol{B}(\boldsymbol{q})$, is used to implement the actuator allocation scheme for the AIAUV. The actuator configuration matrix consists of two matrices $\boldsymbol{B}_{joint} \in \mathbb{R}^{n_q+6\times n_t}$ and $\boldsymbol{B}_{thrust} \in \mathbb{R}^{n_q\times n_q}$, which is a block-diagonal input map for the joint motor forces and the thruster configuration matrix, respectively. The thruster configuration matrix $\boldsymbol{B}_{thrust}$ can be found through the algorithm in [29] and has to be recomputed at every time-step, as it is configuration dependent. Together, they may be combined into the actuator configuration matrix as

$$\boldsymbol{B}(\boldsymbol{q}) = \begin{bmatrix} & \mathbf{0}_{6\times n_q} \\ \boldsymbol{B}_{thrust} & \\ & \boldsymbol{B}_{joint} \end{bmatrix} \tag{2.13}$$

The control torque $\boldsymbol{\tau}$ is then given by

$$\boldsymbol{\tau} = \boldsymbol{B}(\boldsymbol{q})\boldsymbol{u} \tag{2.14}$$

with $\boldsymbol{u}$ being the applied torque to the individual motors and thrusters. This relationship can then be inverted using the Moore-Penrose pseudoinverse in section 3.1, such that

$$\boldsymbol{u} = \boldsymbol{B}^T \left(\boldsymbol{B}\boldsymbol{B}^T\right)^{-1} \boldsymbol{\tau} \tag{2.15}$$

Furthermore, as was pointed out in [26], $\boldsymbol{B}(\boldsymbol{q})$ may become singular in this computation. Therefore, a damped pseudoinverse [31] is used in the allocation whenever the actuation index task is not used, as

$$\boldsymbol{u} = \boldsymbol{B}^T \left(\boldsymbol{B}\boldsymbol{B}^T + \lambda^2 \boldsymbol{I}_{(n+5)\times(n+5)}\right)^{-1} \boldsymbol{\tau} \tag{2.16}$$

where λ is chosen empirically. In this thesis it has been chosen as $\lambda = 0.0071$.

2.2.5 Model assumptions

During modeling of a real process like the movement of the AIAUV, certain simplifying assumptions are required to effectively represent the real behavior. This section will quickly present some of the assumptions made in [29].

First, it is assumed that the manipulator is a kinematic tree, in which there are no kinematic loops [29]. To ensure that the rigid body mass matrix $\boldsymbol{M}(\boldsymbol{q})$ and the hydrodynamic damping matrix $\boldsymbol{D}(\boldsymbol{q})$ is positive definite it is assumed that the system Jacobians $\boldsymbol{J}_i$ are full rank [29], which is both necessary and sufficient for positive definiteness. Furthermore, to reduce the complexity of computing $\boldsymbol{M}_i$ and $\boldsymbol{D}_i$ in (2.10), the links are assumed to be hydrodynamically decoupled [29], meaning links do not interact with one another hydrodynamically. Finally, to greatly reduce the number of parameters required to represent the drag forces on a link, it is assumed that there are no cross-terms present in the link drag wrenches presented in a frame where the coordinate planes are aligned with the three planes of symmetry for the link [29].

2.3 Eely AIAUV

The AIAUV considered in this thesis is the Eely snake robot depicted in figs. 2.1 and 2.3. The manipulator has $n = 9$ links and $n_q = n - 1$ revolute joints. The properties for each link are given in table 2.1.

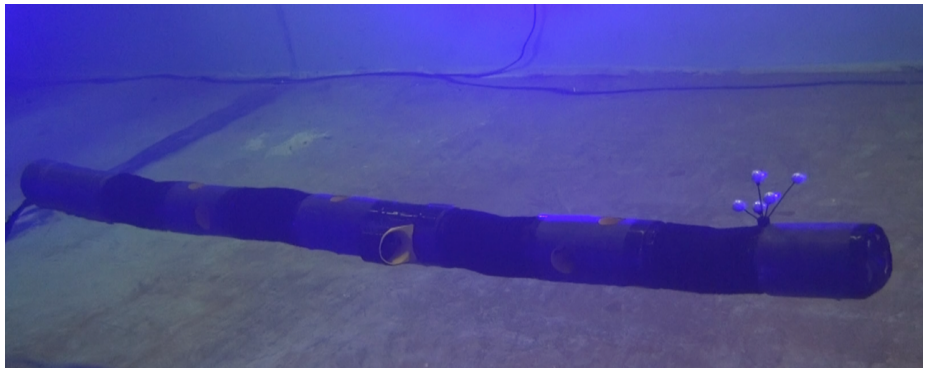

Figure 2.3: Eely AIAUV at NTNU Tyholt. Image provided by Anna M. Kohl.

Remark: It should be noted that every other link in the Eely are short links inside the Eely joints, which makes the robot movement appear three-dimensional.

The robot has a total of $n_t = 7$ thrusters distributed as indicated in table 2.1.

Link L_i	Length $[m]$	Radius $[m]$	Mass $[kg]$	Thrusters
L_1	0.62	0.09	15.771	
L_2	0.10	0.09	2.5447	
L_3	0.59	0.09	15.0137	2
L_4	0.10	0.09	2.5447	
L_5	0.80	0.09	20.3575	3
L_6	0.10	0.09	2.5447	
L_7	0.59	0.09	15.0137	2
L_8	0.10	0.09	2.5447	
L_9	0.37	0.09	9.4154	

Table 2.1: Eely link properties.

C_a	C_{d_x}	C_{d_ϕ}	C_{d_c}	C_{d_t}	α	β	γ
1	0.2	0.1	0.5	0.1	0.2	0.1	0.1

Table 2.2: Eely manipulator parameters.

This configuration of thrusters, combined with the dexterity provided by the $n-1$ revolute joints, means that the Eely AIAUV is underactuated in roll whenever all joints align in an I-shape. As a result, the manipulator is ideal for testing the configuration based singularity avoidance scheme presented in chapter 5.

Table 2.2 shows the physical parameters used in the simulation. Each parameter corresponds to the following:

- C_a: Added mass coefficient for the cross section.
- C_{d_x}: Nonlinear drag coefficient in surge.
- C_{d_ϕ}: Nonlinear drag coefficient in roll.
- C_{d_c}: Nonlinear crossflow drag coefficient.
- C_{d_l}: Linear cross-sectional drag coefficient.
- α: Added mass ratio in surge/heave for a link.
- β: Linear drag parameter in surge.
- γ: Linear drag parameter in roll.

Chapter 3

Operational Space Formulation

Remark: The theory presented in this chapter is the same as was presented in the specialization project, but is included here for completeness.

For robot manipulators, the position and orientation of the end-effector is usually the desired control objective. However, when a hyper-redundant manipulator is considered, additional tasks can be introduced to extend the control objective and utilize the redundant degrees of freedom. With multiple tasks, a framework can be introduced for hierarchical priorities among tasks, and the importance of different control objectives can be set. In the following section, the operational space dynamics formulation introduced by [2] will be presented. The formulation incorporates hierarchical priorities in tasks and enables torque assignment directly from the end-effector dynamics. The following theory focuses on the summary of the operational space formulation given in [28]. First, some basic mathematical concepts are introduced.

3.1 Moore-Penrose pseudoinverse

The Moore-Penrose pseudoinverse is defined as the matrix; $\boldsymbol{A}^+$, that satisfies the following conditions [32]

$$\boldsymbol{A}\boldsymbol{A}^+\boldsymbol{A} = \boldsymbol{A} \tag{3.1a}$$

$$\boldsymbol{A}^+\boldsymbol{A}\boldsymbol{A}^+ = \boldsymbol{A}^+ \tag{3.1b}$$

$$\boldsymbol{A}\boldsymbol{A}^+ \quad \text{is symmetric} \tag{3.1c}$$

$$\boldsymbol{A}^+\boldsymbol{A} \quad \text{is symmetric} \tag{3.1d}$$

The pseudoinverse does not require a matrix to be square, as is the requirement for the normal inverse. Depending on whether the matrix is broad or tall, the right and left pseudoinverse respectively, is defined as

$$\text{Right: } \boldsymbol{A}^+ = \boldsymbol{A}^T \left(\boldsymbol{A}\boldsymbol{A}^T\right)^{-1} \tag{3.2a}$$

$$\text{Left: } \boldsymbol{A}^+ = \left(\boldsymbol{A}^T\mathbf{A}\right)^{-1} \boldsymbol{A}^T \tag{3.2b}$$

Respectively, the right and left pseudoinverse satisfies the property

$$\text{Right: } \boldsymbol{A}\boldsymbol{A}^+ = \boldsymbol{I} \tag{3.3a}$$

$$\text{Left: } \boldsymbol{A}^+\boldsymbol{A} = \boldsymbol{I} \tag{3.3b}$$

3.2 Null-space projector

The Jacobian matrix of a redundant manipulator has more columns than rows, so a general matrix $\boldsymbol{J} \in \mathbb{R}^{m\times n}$ with $m < n$ will admit infinite solutions in $\boldsymbol{x}$ to the linear transformation given by

$$\boldsymbol{y} = \boldsymbol{J}\boldsymbol{x} \tag{3.4}$$

A general solution of this equation is given as

$$\boldsymbol{x} = \boldsymbol{J}^+\boldsymbol{y} + \underbrace{\left(\boldsymbol{I}_n - \boldsymbol{J}^+\boldsymbol{J}\right)}_{\boldsymbol{N}} \boldsymbol{x}_0 \tag{3.5}$$

Where $\boldsymbol{J}^+$ is the Moore-Penrose pseudoinverse described in section 3.1. The matrix $\boldsymbol{N}$ projects any vector into the null space of $\boldsymbol{J}$ and is known as the null-space projector associated with $\boldsymbol{J}$. By multiplying the null-space projector with a vector, the vector elements that affect the movements described by the Jacobian are removed.

3.3 Operational space dynamics

The equations below have been taken from [28]. However, the formulations have been adapted to fit the full simulation model for the AIAUV given in (2.9b). From (2.9b)

$$\boldsymbol{M}(\boldsymbol{q})\dot{\boldsymbol{\zeta}}^b_{b/i} + \boldsymbol{C}(\boldsymbol{q},\boldsymbol{\zeta}^b_{b/i})\boldsymbol{\zeta}^b_{b/i} + \boldsymbol{D}(\boldsymbol{q},\boldsymbol{\zeta}^b_{b/i})\boldsymbol{\zeta}^b_{b/i} + \boldsymbol{g}(\boldsymbol{q}) = \boldsymbol{B}(\boldsymbol{q})\boldsymbol{\tau} \tag{3.6}$$

To simplify the formulations to come, the matrix $\boldsymbol{n}(\boldsymbol{q},\boldsymbol{\zeta}^b_{b/i})$ is introduced as

$$\boldsymbol{n}(\boldsymbol{q},\boldsymbol{\zeta}^b_{b/i}) = \boldsymbol{C}(\boldsymbol{q},\boldsymbol{\zeta}^b_{b/i})\boldsymbol{\zeta}^b_{b/i} + \boldsymbol{D}(\boldsymbol{q},\boldsymbol{\zeta}^b_{b/i})\boldsymbol{\zeta}^b_{b/i} + \boldsymbol{g}(\boldsymbol{q}) \tag{3.7}$$

Now, the generic **task** x is introduced through the task function $\boldsymbol{\sigma}_x$ as

$$\boldsymbol{\sigma}_x = \boldsymbol{\sigma}(\boldsymbol{q}), \quad \boldsymbol{\sigma}_x \in \mathbb{R}^{m_x}, \quad m_x < n \tag{3.8}$$

$\boldsymbol{\sigma}_x$ has the following derivatives [28]

$$\dot{\boldsymbol{\sigma}}_x = \boldsymbol{J}_x(\boldsymbol{q})\boldsymbol{\zeta}^b_{b/i} \tag{3.9a}$$

$$\ddot{\boldsymbol{\sigma}}_x = \boldsymbol{J}_x(\boldsymbol{q})\dot{\boldsymbol{\zeta}}^b_{b/i} + \frac{d}{dt}\left(\boldsymbol{J}_x(\boldsymbol{q},\boldsymbol{\zeta}^b_{b/i})\right)\boldsymbol{\zeta}^b_{b/i} \tag{3.9b}$$

Remark: $\boldsymbol{C}(\boldsymbol{q},\boldsymbol{\zeta}^b_{b/i})$, $\boldsymbol{D}(\boldsymbol{q},\boldsymbol{\zeta}^b_{b/i})$, $\boldsymbol{c}_x(\boldsymbol{q},\boldsymbol{\zeta}^b_{b/i})$, $\boldsymbol{d}_x(\boldsymbol{q},\boldsymbol{\zeta}^b_{b/i})$ and $\frac{d}{dt}\left(\boldsymbol{J}_x(\boldsymbol{q},\boldsymbol{\zeta}^b_{b/i})\right)$ are the only matrices dependent on $\boldsymbol{\zeta}^b_{b/i}$ in the following derivations. All others depend on $\boldsymbol{q}$, thus both dependencies in $\boldsymbol{\zeta}^b_{b/i}$ and $\boldsymbol{q}$ have been omitted from the derivations to improve readability.

The matrix $\boldsymbol{J}_x \in \mathbb{R}^{m_x \times n}$ is the task specific Jacobian associated with task x and can be found as shown in (2.2). Using the task Jacobian, it holds [28]

$$\boldsymbol{\tau} = \boldsymbol{J}_x^T \boldsymbol{f}_x, \tag{3.10}$$

where $\boldsymbol{f}_x$ is the generalized force vector associated with task x. It can be thought of as the force contributing to the fulfillment of the task control objective. The accelerations in (3.6) can then be found as

$$\dot{\boldsymbol{\zeta}}^b_{b/i} = \boldsymbol{M}^{-1}\left[\boldsymbol{J}_x^T \boldsymbol{f}_x - \boldsymbol{n}\right] \tag{3.11}$$

Inserting (3.11) into (3.9b) yields

$$\ddot{\boldsymbol{\sigma}}_x = \boldsymbol{J}_x \boldsymbol{M}^{-1} \boldsymbol{J}_x^T \boldsymbol{f}_x - \boldsymbol{J}_x \boldsymbol{M}^{-1} \boldsymbol{n} + \frac{d}{dt} \left(\boldsymbol{J}_x \right) \boldsymbol{\zeta}_{b/i}^b \tag{3.12}$$

System matrices can be defined to rewrite the operational space dynamics in (3.12) more intuitively [28]. The matrices are

$$\boldsymbol{M}_x = \left(\boldsymbol{J}_x \boldsymbol{M}^{-1} \boldsymbol{J}_x^T \right)^{-1} \in \mathbb{R}^{m_x \times m_x} \tag{3.13a}$$

$$\boldsymbol{c}_x = \boldsymbol{M}_x \left[\boldsymbol{J}_x \boldsymbol{M}^{-1} \boldsymbol{C} \boldsymbol{\zeta}_{b/i}^b - \frac{d}{dt} \left(\boldsymbol{J}_x \right) \boldsymbol{\zeta}_{b/i}^b \right] \tag{3.13b}$$

$$\boldsymbol{d}_x = \boldsymbol{M}_x \boldsymbol{J}_x \boldsymbol{M}^{-1} \boldsymbol{D} \boldsymbol{\zeta}_{b/i}^b \tag{3.13c}$$

$$\boldsymbol{g}_x = \boldsymbol{M}_x \boldsymbol{J}_x \boldsymbol{M}^{-1} \boldsymbol{g} \tag{3.13d}$$

Which gives the following operational space dynamics

$$\boldsymbol{M}_x \ddot{\boldsymbol{\sigma}}_x + \boldsymbol{c}_x + \boldsymbol{d}_x + \boldsymbol{g}_x = \boldsymbol{f}_x \tag{3.14}$$

From equation (3.10) it would be desirable to find the generalized task force expressed through the torque. However, the task Jacobian $\boldsymbol{J}_x$ is not invertible. Define the weighted pseudoinverse of the task Jacobian as [28]

$$\bar{\boldsymbol{J}}_x = \boldsymbol{M}^{-1} \boldsymbol{J}_x^T \boldsymbol{M}_x \tag{3.15a}$$

$$\bar{\boldsymbol{J}}_x = \boldsymbol{M}^{-1} \boldsymbol{J}_x^T \left(\boldsymbol{J}_x \boldsymbol{M}^{-1} \boldsymbol{J}_x^T \right)^{-1} \in \mathbb{R}^{n \times m_x} \tag{3.15b}$$

Replacing the Jacobian in (3.10) by the weighted pseudoinverse, the equation can be inverted. Thus

$$\boldsymbol{f}_x = \bar{\boldsymbol{J}}_x^T \boldsymbol{\tau} \tag{3.16}$$

The task specific null-space projector is given by [28]

$$\boldsymbol{N}_x = \boldsymbol{I}_n - \boldsymbol{J}_x^T \bar{\boldsymbol{J}}_x^T \tag{3.17}$$

where the weighted pseudoinverse in (3.15b) has been used to compute the null-space projector in (3.5). This null-space projector, where $\boldsymbol{M}(\boldsymbol{q})$ is used as the weighting matrix, has been shown to be dynamically consistent [2], meaning that a subtask will never generate any accelerations in the operational space of any higher priority task in either static or dynamic conditions [33]. It decouples the operational space dynamics from the null space dynamics and minimizes the kinetic and acceleration energy of the system [18].

3.3.1 Multiple task control

For two tasks denoted by a and b, where a is the higher priority task and b has lower priority, the applied torque, $\boldsymbol{\tau}$, is [28]

$$\boldsymbol{\tau} = \boldsymbol{\tau}_a + \boldsymbol{N}_a \boldsymbol{\tau}_b \tag{3.18}$$

The null-space projector is given for a generic task in (3.17). For the highest priority task a the dynamics are given by

$$\boldsymbol{M}_a \ddot{\boldsymbol{\sigma}}_a + \boldsymbol{c}_a + \boldsymbol{d}_a + \boldsymbol{g}_a = \bar{\boldsymbol{J}}_a^T \boldsymbol{\tau}_a \tag{3.19}$$

For task b

$$\boldsymbol{M}_b \ddot{\boldsymbol{\sigma}}_b + \boldsymbol{c}_b + \boldsymbol{d}_b + \boldsymbol{g}_b = \bar{\boldsymbol{J}}_b^T \left(\boldsymbol{\tau}_a + \boldsymbol{N}_a \boldsymbol{\tau}_b \right) \tag{3.20}$$

The torque assignment for two tasks in a task-priority system is the simplest, but [34] shows how the framework can be adapted to suit n tasks. The control torque with a third task c unto task n is given by

$$\boldsymbol{\tau} = \boldsymbol{\tau}_a + \boldsymbol{N}_a \boldsymbol{\tau}_b + \boldsymbol{N}_{ab} \boldsymbol{\tau}_c + \cdots + \boldsymbol{N}_{a\ldots n} \boldsymbol{\tau}_n \tag{3.21}$$

where the combined null-space projector is

$$\boldsymbol{N}_{a\ldots i} = \boldsymbol{I} - \sum_{i=a}^{i-1} \boldsymbol{J}_i^T \bar{\boldsymbol{J}}_i^T \tag{3.22}$$

Chapter 4

Sliding Mode Control

This chapter introduces the concept of sliding mode control, where a short introduction is given. Then, the considerations when designing a sliding surface is discussed, before the super-twisting algorithm with adaptive gains [23] and the generalized super-twisting algorithm [21] is presented.

4.1 Introduction to sliding mode control

When modeling real world applications with mathematics there will always be uncertainty and errors. This means simulation results might behave perfectly, while new phenomena not captured by the modeling can arise during real world application. To deal with these uncertainties, the area of robust control methods have received increased attention.

One approach to robust control design is sliding mode control (SMC). In SMC, the goal is to design a feedback control law $\boldsymbol{\tau}$ which achieves asymptotic convergence for the chosen control objective in presence of unknown disturbances. The system is often written in error coordinates $\tilde{\boldsymbol{\xi}}$ where $\boldsymbol{x}_1$ and $\boldsymbol{x_2}$ is chosen as

$$\boldsymbol{x}_1 = \tilde{\boldsymbol{\xi}} \tag{4.1a}$$

$$\boldsymbol{x}_2 = \dot{\tilde{\boldsymbol{\xi}}} \tag{4.1b}$$

The error dynamics can then be seen as

$$\dot{\boldsymbol{x}}_1 = \boldsymbol{x}_2 \tag{4.2a}$$
$$\dot{\boldsymbol{x}}_2 = f\left(\boldsymbol{x}_1, \boldsymbol{x}_2, t\right) \tag{4.2b}$$

where t indicates it can be time dependent. Instead of directly designing a control law $\boldsymbol{\tau}$ to make $\boldsymbol{x}_1$ and $\boldsymbol{x}_2$ converge, SMC introduces a sliding surface which should be driven to zero in finite time by means of $\boldsymbol{\tau}$. The control design is split in two as

1. Design a sliding surface $\boldsymbol{s}(\boldsymbol{x}_1, \boldsymbol{x}_2)$.
2. Design a control law $\boldsymbol{\tau}$ which makes $\boldsymbol{s} \to \mathbf{0}$ in finite time.

The sliding surface should be designed such that when $\boldsymbol{\tau}$ drives $\boldsymbol{s} \to \mathbf{0}$, the error dynamics $\boldsymbol{x}_1$ and $\boldsymbol{x}_2$ asymptotically converge to zero. The control $\boldsymbol{\tau}$ that drives $\boldsymbol{x}_1$ and $\boldsymbol{x}_2$ to the sliding surface in finite time is known as a sliding mode controller.

Figure 4.1 demonstrates the two stages of a sliding mode controller. First, $\boldsymbol{x}_1$ and $\boldsymbol{x}_2$ is driven to the sliding surface during what is called the reaching phase. Once on the surface, the controller ensures $\boldsymbol{x}_1$ and $\boldsymbol{x}_2$ remain on the surface in the presence of unknown disturbances and model uncertainties and that $\boldsymbol{x}_1$ and $\boldsymbol{x}_2$ is asymptotically driven to zero, in what is known as the sliding phase.

For the sliding surface to converge to zero in finite time, a sign function is used in the controller $\boldsymbol{\tau}$. This means SMC is a nonlinear controller with high-frequency switching. The switching behavior causes chattering since the control input is not continuous. Chattering can be prohibited through higher-order sliding mode controllers, some of which are mentioned later in this chapter.

A simple and intuitive example of first-order SMC is given in the first chapter of [17].

4.2 Sliding surface

The first step when implementing an SMC is to design a sliding surface $\boldsymbol{s}_x$. The surface should be such that the error variables are asymptotically driven to zero when $\boldsymbol{s}_x = 0$. In addition, the control input should appear in the first derivative of $\boldsymbol{s}_x$, i.e. the system has a relative degree of 1. The sliding surface used in this

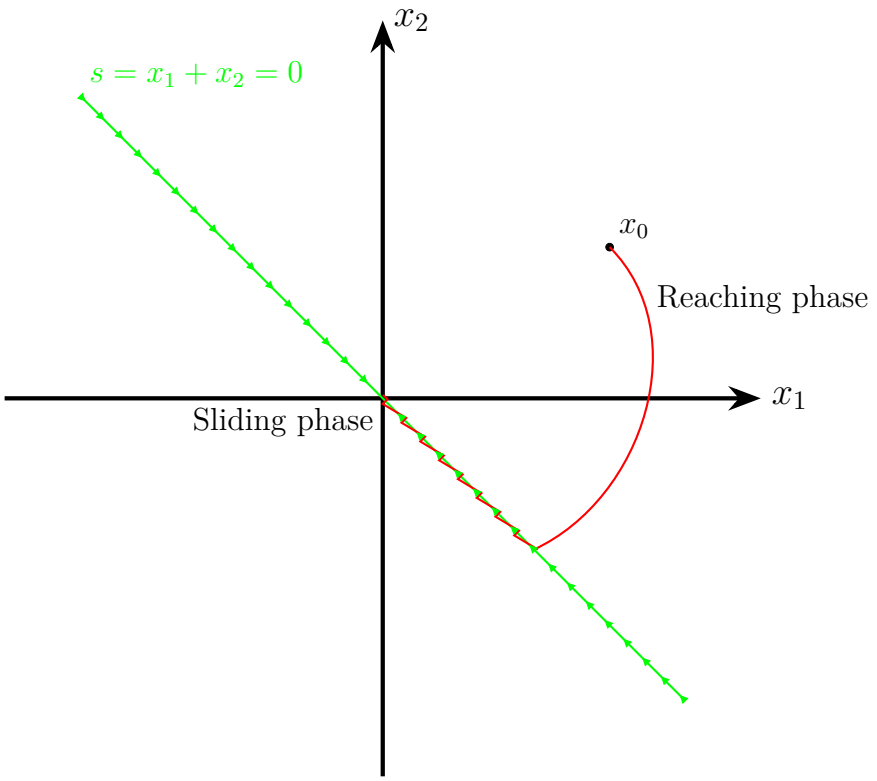

Figure 4.1: Reaching and sliding phase represented in red and green, respectively.

project is chosen as

$$\boldsymbol{s}_x = \boldsymbol{x}_1 + \boldsymbol{x}_2 \in \mathbb{R}^{m_x} \tag{4.3}$$

where $\boldsymbol{x}_1$ is the coordinate error and $\boldsymbol{x}_2$ is the coordinate velocity error. Provided the system error dynamics are in the form described by eqs. (4.1) and (4.2), setting $\boldsymbol{s}_x = 0$ and using that $\dot{\boldsymbol{x}}_\mathbf{1} = \boldsymbol{x}_2$ yields

$$\boldsymbol{x}_1 + \boldsymbol{x}_2 = 0 \tag{4.4a}$$

$$\dot{\boldsymbol{x}}_1 = -\boldsymbol{x}_1 \tag{4.4b}$$

This means that $\boldsymbol{x}_1$ will globally exponentially converge to zero [11] provided that $\boldsymbol{s}_x = 0$.

4.3 Super-twisting algorithm with adaptive gains

An SMC algorithm previously used for AIAUVs, and proven to be better than a standard PD controller in [20], is the super-twisting algorithm with adaptive gains (STA). The super-twisting algorithm attenuates chattering and only requires than an upper bound exists on disturbances and parameter uncertainties, but the bound must not be known. Proposed in [23], the super-twisting algorithm with adaptive gains is

$$\boldsymbol{u}_{STA} = -\boldsymbol{\alpha}|\boldsymbol{s}|^{\frac{1}{2}} \circ sgn(\boldsymbol{s}) + \boldsymbol{v} \tag{4.5a}$$

$$\dot{\boldsymbol{v}} = -\boldsymbol{\beta}_{STA} \circ sgn(\boldsymbol{s}) \tag{4.5b}$$

where $\boldsymbol{s}$ is the sliding surface described in (4.3). Here $\circ$ is the Hadamard product, or element-wise vector multiplication, where

$$(\boldsymbol{a} \circ \boldsymbol{b})_i = \boldsymbol{a}_i \boldsymbol{b}_i \tag{4.6}$$

The adaptive gain $\boldsymbol{\alpha}$ is chosen as

$$\dot{\boldsymbol{\alpha}} = \begin{cases} \boldsymbol{\omega}_1 \circ \sqrt{\frac{\boldsymbol{\gamma}_1}{2}} & \text{if } \boldsymbol{s} \neq 0 \\ 0 & \text{if } \boldsymbol{s} = 0 \end{cases} \tag{4.7}$$

and the adaptive gain $\boldsymbol{\beta}_{STA}$ is chosen through several specified gains and the adaptive gain $\boldsymbol{\alpha}$ as

$$\boldsymbol{\beta}_{STA} = 2\boldsymbol{\epsilon}_1 \circ \boldsymbol{\alpha} + \boldsymbol{\lambda}_1 + 4\boldsymbol{\epsilon}_1 \circ \boldsymbol{\epsilon}_1 \tag{4.8}$$

The remaining gains $\boldsymbol{\epsilon}_1 \in \mathbb{R}^{m_x}, \boldsymbol{\omega}_1 \in \mathbb{R}^{m_x}, \boldsymbol{\lambda}_1 \in \mathbb{R}^{m_x}, \boldsymbol{\gamma}_1 \in \mathbb{R}^{m_x}$ are chosen prior to the simulation. The terms $\boldsymbol{\alpha}$ and $\boldsymbol{\beta}_{STA}$ are adaptive gains which change according to the system during runtime. When implementing the controller a boundary is set for the sliding surface, such that (4.7) becomes

$$\dot{\boldsymbol{\alpha}} = \begin{cases} \boldsymbol{\omega}_1 \circ \sqrt{\frac{\boldsymbol{\gamma}_1}{2}} & \text{if } |\boldsymbol{s}| > \alpha_m \\ 0 & \text{if } |\boldsymbol{s}| \leq \alpha_m \end{cases} \tag{4.9}$$

where the constant α_m is chosen empirically.

4.3.1 Assumptions

There are some underlying assumptions required for the STA algorithm to function. First, assume that a sliding surface $\boldsymbol{s}_x$ has been chosen such that the

compensated dynamics while the system is in sliding mode ($\boldsymbol{s}_x = 0$) is as desired [23]. Furthermore, the system's input-output dynamics, from control input to sliding surface, should be of relative degree one, and the internal dynamics should be stable [23]. The input-output dynamics are of the form

$$\dot{\boldsymbol{s}}_x = \boldsymbol{\phi}(\boldsymbol{x}, t) + \boldsymbol{b}(\boldsymbol{x}, t)\boldsymbol{u} \tag{4.10}$$

where $\boldsymbol{b}(\boldsymbol{x}, t)$ is known and not equal to zero and the perturbation function $\boldsymbol{\phi}(\boldsymbol{x}, t)$ is bounded, although the exact bound is unknown [23]. Then the super-twisting algorithm will drive the sliding variable $\boldsymbol{s}_x$ and $\dot{\boldsymbol{s}}_x$ to zero in finite time [23].

4.4 Generalized super-twisting algorithm

This section describes the base equations, taken from [11], of the generalized super-twisting algorithm (GSTA) [21]. The GSTA is an extension to the STA presented in section 4.3 where an extra linear term is added in addition to two growing terms. The super-twisting algorithm handles the chattering problem by hiding it behind an integrator as shown in (4.11b). The generalized super-twisting algorithm is able to handle time and state-dependent perturbations which through state dependent derivative would drive the perturbations to grow exponentially in time [21]. The controller equations are

$$\boldsymbol{u}_{GSTA} = -\boldsymbol{k}_1 \circ \boldsymbol{\phi}_1(\boldsymbol{s}) + \boldsymbol{z} \tag{4.11a}$$

$$\dot{\boldsymbol{z}} = -\boldsymbol{k}_2 \circ \boldsymbol{\phi}_2(\boldsymbol{s}) \tag{4.11b}$$

$$\boldsymbol{\phi}_1(\boldsymbol{s}) = \lceil \boldsymbol{s} \rfloor^{\frac{1}{2}} + \boldsymbol{\beta}_{GSTA} \circ \boldsymbol{s} \tag{4.11c}$$

$$\boldsymbol{\phi}_2(\boldsymbol{s}) = \frac{1}{2}\lceil \boldsymbol{s} \rfloor^0 + \frac{3}{2}\boldsymbol{\beta}_{GSTA} \circ \lceil \boldsymbol{s} \rfloor^{\frac{3}{2}} + \boldsymbol{\beta}_{GSTA}^2 \circ \boldsymbol{s} \tag{4.11d}$$

where $\boldsymbol{s}$ is the sliding surface described in (4.3). Here $\lceil \boldsymbol{a} \rfloor^b = |\boldsymbol{a}|^b \circ sgn(\boldsymbol{a})$ with $|\boldsymbol{a}|$ being the element-wise absolute value and $\circ$ is the Hadamard product in (4.6). The control gains to be chosen are $\boldsymbol{\beta}_{GSTA} \in \mathbb{R}^{m_x}, \boldsymbol{k}_1 \in \mathbb{R}^{m_x}, \boldsymbol{k}_2 \in \mathbb{R}^{m_x}$.

The linear term in (4.11c) means three degrees of freedom are obtained through the controller gains design; $\boldsymbol{\beta}_{GSTA}, \boldsymbol{k}_1, \boldsymbol{k}_2$. In addition, the growing terms in (4.11d) combats the effects of state-dependent perturbations, making the algorithm more robust.

4.4.1 Assumptions

Given a dynamic system

$$\dot{x} = \boldsymbol{\gamma}(\boldsymbol{x}, t)\boldsymbol{u} + \boldsymbol{\phi}(\boldsymbol{x}, t) \tag{4.12}$$

where $\boldsymbol{\gamma}$ and $\boldsymbol{\phi}$ are uncertain function which depend on state $\boldsymbol{x}$ and time t. Assume $\boldsymbol{\gamma}$ and $\boldsymbol{\phi}$ are Lipschitz continuous with respect to t and $\mathbb{C}^1$ with respect to $\boldsymbol{x}$ [22]. Furthermore, $\boldsymbol{\gamma}$ is assumed to be bounded above and below by positive constants [22]. Provided the perturbations $\boldsymbol{\phi}$ are bounded for the vanishing term and the total time-derivative is bounded as described in [22], then the states $\boldsymbol{x}_1$ and $\boldsymbol{x}_2$ converge to zero $\boldsymbol{z}$ converges globally and in finite time [22].

Part II

Robust Control and Singularity Avoidance

Chapter 5

Actuator Singularity Avoidance

Remark: This chapter is a product of the specialization project and is included her for completeness, as the actuation index task is an important use case for the simulation examples in part III.

To perform singularity avoidance the actuation index is introduced. It is inspired by the manipulability index [25] used for kinematic singularity avoidance in [9], which was verified through a simulation study using an AIAUV. The actuation index is introduced as

$$\sigma = \det\left(\mathbf{B}(\mathbf{q})\mathbf{B}(\mathbf{q})^T\right) \tag{5.1}$$

Where $\mathbf{B}$ is the actuator configuration matrix. An actuator singularity occurs when $\sigma = 0$. The actuation index limit is then enforced as a set-based high-priority task.

In [9], the control problem was divided into inverse kinematics and a dynamic controller. Using the operational space formulation, these are merged into the inverse dynamics such that the control torques are assigned directly based on the dynamics of the end-effector. For multiple tasks, the torque assignment is given by eqs. (3.18) and (8.3). To calculate the matrices in the operational space

dynamics in (3.13) the task specific Jacobian, $\boldsymbol{J}_\sigma$, and its derivative, $\dot{\boldsymbol{J}}_\sigma$, is required. The following sections will derive formulas to be used for computation of these Jacobians.

5.1 Task Jacobian

The task Jacobian can be calculated according to (2.3), where σ is a scalar function as it is given by the determinant. Inserting σ gives the task Jacobian

$$\boldsymbol{J}_\sigma = \begin{bmatrix} \boldsymbol{0}_{1\times 6} & \frac{\partial \sigma}{\partial q_1} & \frac{\partial \sigma}{\partial q_2} & \frac{\partial \sigma}{\partial q_3} & \cdots & \frac{\partial \sigma}{\partial q_{n_q}} \end{bmatrix} \tag{5.2}$$

The first six elements correspond to the base of the robot, where the position of the robot does not matter, only the joint angles, and thus they are zero for the task Jacobian. The remaining elements can be found by differentiating the actuation index with respect to the different joint coordinates.

5.2 Actuation index derivative

To find the task Jacobian, the partial derivative of σ with respect to an arbitrary joint parameter q_i is computed. The partial derivative of the determinant function in σ is given by (A.5e), where $\boldsymbol{Y} = \boldsymbol{B}(\boldsymbol{q})\boldsymbol{B}(\boldsymbol{q})^T$. Inserting $\boldsymbol{Y}$ into (A.5e), and omitting dependencies for readability, yields

$$\frac{\partial \sigma}{\partial q_i} = \det\left(\boldsymbol{B}\boldsymbol{B}^T\right) \operatorname{Tr}\left(\left(\boldsymbol{B}\boldsymbol{B}^T\right)^{-1} \frac{\partial\left(\boldsymbol{B}\boldsymbol{B}^T\right)}{\partial q_i}\right) \tag{5.3a}$$

$$= \sigma \operatorname{Tr}\left(\left(\boldsymbol{B}\boldsymbol{B}^T\right)^{-1} \frac{\partial\left(\boldsymbol{B}\boldsymbol{B}^T\right)}{\partial q_i}\right) \tag{5.3b}$$

Using (A.4a) on the trace in (5.3b) yields

$$\operatorname{Tr}\left(\left(\boldsymbol{B}\boldsymbol{B}^T\right)^{-1} \frac{\partial\left(\boldsymbol{B}\boldsymbol{B}^T\right)}{\partial q_i}\right) = \operatorname{Tr}\left(\frac{\partial\left(\boldsymbol{B}\boldsymbol{B}^T\right)}{\partial q_i}\left(\boldsymbol{B}\boldsymbol{B}^T\right)^{-1}\right) \tag{5.4}$$

Combining (5.4) with (A.5a) gives

$$\mathcal{Q} = \mathrm{Tr}\left(\frac{\partial\left(\boldsymbol{B}\boldsymbol{B}^T\right)}{\partial q_i}\left(\boldsymbol{B}\boldsymbol{B}^T\right)^{-1}\right) = \mathrm{Tr}\left(\left(\frac{\partial \boldsymbol{B}}{\partial q_i}\boldsymbol{B}^T + \boldsymbol{B}\frac{\partial \boldsymbol{B}^T}{\partial q_i}\right)\left(\boldsymbol{B}\boldsymbol{B}^T\right)^{-1}\right) \tag{5.5}$$

Equation (A.5b) allows rewriting (5.5) into

$$\mathcal{Q} = \mathrm{Tr}\left(\frac{\partial \boldsymbol{B}}{\partial q_i}\boldsymbol{B}^T\left(\boldsymbol{B}\boldsymbol{B}^T\right)^{-1} + \boldsymbol{B}\left(\frac{\partial \boldsymbol{B}}{\partial q_i}\right)^T\left(\boldsymbol{B}\boldsymbol{B}^T\right)^{-1}\right) \tag{5.6}$$

Through eqs. (A.4b) and (5.6), $\mathcal{Q}$ is

$$\mathcal{Q} = \mathrm{Tr}\left(\frac{\partial \boldsymbol{B}}{\partial q_i}\boldsymbol{B}^T\left(\boldsymbol{B}\boldsymbol{B}^T\right)^{-1}\right) + \mathrm{Tr}\left(\boldsymbol{B}\left(\frac{\partial \boldsymbol{B}}{\partial q_i}\right)^T\left(\boldsymbol{B}\boldsymbol{B}^T\right)^{-1}\right) \tag{5.7}$$

The Moore-Penrose pseudoinverse is defined in (3.2) and for the broad matrix $\boldsymbol{B}$ it is

$$\boldsymbol{B}^+ = \boldsymbol{B}^T\left(\boldsymbol{B}\boldsymbol{B}^T\right)^{-1} \tag{5.8}$$

This means

$$\mathcal{Q} = \underbrace{\mathrm{Tr}\left(\frac{\partial \boldsymbol{B}}{\partial q_i}\boldsymbol{B}^+\right)}_{\mathcal{Q}_1} + \underbrace{\mathrm{Tr}\left(\boldsymbol{B}\left(\frac{\partial \boldsymbol{B}}{\partial q_i}\right)^T\left(\boldsymbol{B}\boldsymbol{B}^T\right)^{-1}\right)}_{\mathcal{Q}_2} \tag{5.9}$$

Examining the second trace it can be rewritten through (A.4c) as

$$\mathcal{Q}_2 = \mathrm{Tr}\left(\left(\boldsymbol{B}\boldsymbol{B}^T\right)^{-1}\boldsymbol{B}\left(\frac{\partial \boldsymbol{B}}{\partial q_i}\right)^T\right) \tag{5.10}$$

Equation (A.4d) allows taking the transpose of the interior of the trace as

$$\mathcal{Q}_2 = \mathrm{Tr}\left(\left[\left(\boldsymbol{B}\boldsymbol{B}^T\right)^{-1}\boldsymbol{B}\left(\frac{\partial \boldsymbol{B}}{\partial q_i}\right)^T\right]^T\right) \tag{5.11}$$

From (A.2a), $\mathcal{Q}_2$ can be written as

$$\mathcal{Q}_2 = \mathrm{Tr}\left(\left[\left(\frac{\partial \boldsymbol{B}}{\partial q_i}\right)^T\right]^T \boldsymbol{B}^T \left[\left(\boldsymbol{B}\boldsymbol{B}^T\right)^{-1}\right]^T\right) \tag{5.12}$$

With eqs. (A.2b) and (A.2c) and the symmetry property in (A.2d), it holds

$$\mathcal{Q}_2 = \mathrm{Tr}\left(\frac{\partial \boldsymbol{B}}{\partial q_i}\boldsymbol{B}^T\left(\boldsymbol{B}\boldsymbol{B}^T\right)^{-1}\right) \tag{5.13a}$$

$$= \mathrm{Tr}\left(\frac{\partial \boldsymbol{B}}{\partial q_i}\boldsymbol{B}^+\right) \tag{5.13b}$$

Replacing (5.13b) into (5.9) yields the derivative of the task function σ.

$$\frac{\partial \sigma}{\partial q_i} = 2\sigma\,\mathrm{Tr}\left(\frac{\partial \boldsymbol{B}}{\partial q_i}\boldsymbol{B}^+\right) \tag{5.14}$$

The expression in (5.14) can then be used to compute the task Jacobian in (5.2).

5.3 Task Jacobian derivative

The result in section 5.2 can further be used to find the derivative of the task Jacobian. Differentiating (5.2) with respect to time gives

$$\frac{d}{dt}\left(\boldsymbol{J}_\sigma\right) = \begin{bmatrix} \frac{\partial}{\partial \boldsymbol{q}}\left(\mathbf{0}_{1\times 6}\right)\dot{\boldsymbol{q}} & \frac{\partial}{\partial \boldsymbol{q}}\left(\frac{\partial \sigma}{\partial q_1}\right)\dot{\boldsymbol{q}} & \frac{\partial}{\partial \boldsymbol{q}}\left(\frac{\partial \sigma}{\partial q_2}\right)\dot{\boldsymbol{q}} & \cdots & \frac{\partial}{\partial \boldsymbol{q}}\left(\frac{\partial \sigma}{\partial q_{n_q}}\right)\dot{\boldsymbol{q}} \end{bmatrix} \tag{5.15a}$$

$$= \begin{bmatrix} \mathbf{0}_{6\times 1} \\ \frac{\partial^2\sigma}{\partial q_1^2}\dot{q}_1 + \frac{\partial^2\sigma}{\partial q_1 \partial q_2}\dot{q}_2 + \frac{\partial^2\sigma}{\partial q_1 \partial q_3}\dot{q}_3 + \cdots + \frac{\partial^2\sigma}{\partial q_1 \partial q_{n_q}}\dot{q}_{n_q} \\ \frac{\partial^2\sigma}{\partial q_2 \partial q_1}\dot{q}_1 + \frac{\partial^2\sigma}{\partial q_2^2}\dot{q}_2 + \frac{\partial^2\sigma}{\partial q_2 \partial q_3}\dot{q}_3 + \cdots + \frac{\partial^2\sigma}{\partial q_2 \partial q_{n_q}}\dot{q}_{n_q} \\ \frac{\partial^2\sigma}{\partial q_3 \partial q_1}\dot{q}_1 + \frac{\partial^2\sigma}{\partial q_3 \partial q_2}\dot{q}_2 + \frac{\partial^2\sigma}{\partial q_3^2}\dot{q}_3 + \cdots + \frac{\partial^2\sigma}{\partial q_3 \partial q_{n_q}}\dot{q}_{n_q} \\ \vdots \\ \frac{\partial^2\sigma}{\partial q_{n_q} \partial q_1}\dot{q}_1 + \frac{\partial^2\sigma}{\partial q_{n_q} \partial q_2}\dot{q}_2 + \frac{\partial^2\sigma}{\partial q_{n_q} \partial q_3}\dot{q}_3 + \cdots + \frac{\partial^2\sigma}{\partial q_{n_q}^2}\dot{q}_{n_q} \end{bmatrix}^T \tag{5.15b}$$

$$= \begin{bmatrix} \mathbf{0}_{6\times 1} \\ \dot{q}_1 \\ \dot{q}_2 \\ \dot{q}_3 \\ \vdots \\ \dot{q}_{n_q} \end{bmatrix}^T \begin{bmatrix} \mathbf{0}_{6\times n_q} \\ \frac{\partial^2\sigma}{\partial q_1^2} & \frac{\partial^2\sigma}{\partial q_2 \partial q_1} & \frac{\partial^2\sigma}{\partial q_3 \partial q_1} & \cdots & \frac{\partial^2\sigma}{\partial q_{n_q} \partial q_1} \\ \frac{\partial^2\sigma}{\partial q_1 \partial q_2} & \frac{\partial^2\sigma}{\partial q_2^2} & \frac{\partial^2\sigma}{\partial q_3 \partial q_2} & \cdots & \frac{\partial^2\sigma}{\partial q_{n_q} \partial q_2} \\ \frac{\partial^2\sigma}{\partial q_1 \partial q_3} & \frac{\partial^2\sigma}{\partial q_2 \partial q_3} & \frac{\partial^2\sigma}{\partial q_3^2} & \cdots & \frac{\partial^2\sigma}{\partial q_{n_q} \partial q_3} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ \frac{\partial^2\sigma}{\partial q_1 \partial q_{n_q}} & \frac{\partial^2\sigma}{\partial q_2 \partial q_{n_q}} & \frac{\partial^2\sigma}{\partial q_3 \partial q_{n_q}} & \cdots & \frac{\partial^2\sigma}{\partial q_{n_q}^2} \end{bmatrix} \tag{5.15c}$$

Thus it comes down to finding the matrix elements of (5.15c). A general rule for the derivatives can be found by differentiating (5.14) with respect to q_j as

$$\sigma''_{ij} = \frac{\partial}{\partial q_j}\left(\frac{\partial \sigma}{\partial q_i}\right) = \frac{\partial}{\partial q_j}\left(2\sigma \operatorname{Tr}\left[\frac{\partial \boldsymbol{B}}{\partial q_i}\boldsymbol{B}^+\right]\right) \quad i = 1...n_q, j = 1...n_q \tag{5.16}$$

Where σ''_{ij} is the cross partial of σ first differentiated with respect to q_i, then q_j.

Remark: The determinant is a continuous function, meaning σ is continuous. This means the non-zero elements of the matrix in (5.15c) is symmetric and it holds

$$\sigma''_{ij} = \sigma''_{ji} \tag{5.17}$$

This fact saves computation time during simulation as the symmetry prevents repeated calculations of the same element.

Using the product rule for matrix differentiation in (A.5a) gives

$$\sigma''_{ij} = \underbrace{2\frac{\partial \sigma}{\partial q_j}\,\mathrm{Tr}\left(\frac{\partial \boldsymbol{B}}{\partial q_i}\boldsymbol{B}^{+}\right)}_{\alpha_1} + \underbrace{2\sigma\frac{\partial}{\partial q_j}\left(\mathrm{Tr}\left[\frac{\partial \boldsymbol{B}}{\partial q_i}\boldsymbol{B}^{+}\right]\right)}_{\alpha_2} \tag{5.18}$$

For α_1

- $\frac{\partial \sigma}{\partial q_j}$ is found in section 5.1 through (5.14) and corresponds to the non-zero columns of the task Jacobian.
- $\frac{\partial \boldsymbol{B}}{\partial q_i}$ is found symbolically as described in section 5.4.
- $\boldsymbol{B}^{+}$ is found through (5.8).

Thus α_1 is known. Equation (A.5c) allows rewriting α_2 as

$$\alpha_2 = 2\sigma\,\mathrm{Tr}\left(\frac{\partial}{\partial q_j}\left[\frac{\partial \boldsymbol{B}}{\partial q_i}\boldsymbol{B}^{+}\right]\right) \tag{5.19}$$

Applying (A.5a) yields

$$\alpha_2 = 2\sigma\,\mathrm{Tr}\left(\underbrace{\frac{\partial^2 \boldsymbol{B}}{\partial q_j \partial q_i}\boldsymbol{B}^{+}}_{\beta_1} + \underbrace{\frac{\partial \boldsymbol{B}}{\partial q_i}\frac{\partial \boldsymbol{B}^{+}}{\partial q_j}}_{\beta_2}\right) \tag{5.20}$$

For β_1 the only unknown is $\frac{\partial}{\partial q_j}\left[\frac{\partial \boldsymbol{B}}{\partial q_i}\right]$ which can be found symbolically as described in section 5.4. For β_2 the unknown is $\frac{\partial \boldsymbol{B}^{+}}{\partial q_j}$ which can be found through (5.8) as

$$\frac{\partial \boldsymbol{B}^{+}}{\partial q_j} = \frac{\partial\left(\boldsymbol{B}^{T}\left(\boldsymbol{B}\boldsymbol{B}^{T}\right)^{-1}\right)}{\partial q_j} \tag{5.21}$$

Once more applying (A.5a) gives

$$\frac{\partial \boldsymbol{B}^{+}}{\partial q_j} = \underbrace{\left(\frac{\partial \boldsymbol{B}}{\partial q_j}\right)^{T}\left(\boldsymbol{B}\boldsymbol{B}^{T}\right)^{-1}}_{\gamma_1} + \underbrace{\boldsymbol{B}^{T}\frac{\partial}{\partial q_j}\left(\boldsymbol{B}\boldsymbol{B}^{T}\right)^{-1}}_{\gamma_2} \tag{5.22}$$

All the elements of γ_1 is already known. For γ_2, the derivative of an inverse matrix is shown in (A.5d), which leads to

$$\gamma_2 = \boldsymbol{B}^{T}\left(-\left(\boldsymbol{B}\boldsymbol{B}^{T}\right)^{-1}\frac{\partial\left(\boldsymbol{B}\boldsymbol{B}^{T}\right)}{\partial q_j}\left(\boldsymbol{B}\boldsymbol{B}^{T}\right)^{-1}\right) \tag{5.23}$$

Where

$$\frac{\partial\left(\boldsymbol{B}\boldsymbol{B}^{T}\right)}{\partial q_j} = \frac{\partial \boldsymbol{B}}{\partial q_j}\boldsymbol{B}^{T} + \boldsymbol{B}\left(\frac{\partial \boldsymbol{B}}{\partial q_j}\right)^{T} \tag{5.24}$$

At this point every element is known and the equations can be retraced to the full expression. Inserting (5.24) into (5.23) yields

$$\gamma_2 = -\boldsymbol{B}^{+}\left(\frac{\partial \boldsymbol{B}}{\partial q_j}\boldsymbol{B}^{T} + \boldsymbol{B}\left(\frac{\partial \boldsymbol{B}}{\partial q_j}\right)^{T}\right)\left(\boldsymbol{B}\boldsymbol{B}^{T}\right)^{-1} \tag{5.25}$$

Inserting γ_2 in (5.22) results in

$$\frac{\partial \boldsymbol{B}^{+}}{\partial q_j} = \left(\frac{\partial \boldsymbol{B}}{\partial q_j}\right)^{T}\left(\boldsymbol{B}\boldsymbol{B}^{T}\right)^{-1} - \boldsymbol{B}^{+}\left(\frac{\partial \boldsymbol{B}}{\partial q_j}\boldsymbol{B}^{T} + \boldsymbol{B}\left(\frac{\partial \boldsymbol{B}}{\partial q_j}\right)^{T}\right)\left(\boldsymbol{B}\boldsymbol{B}^{T}\right)^{-1} \tag{5.26a}$$

$$= \left(\left(\frac{\partial \boldsymbol{B}}{\partial q_j}\right)^{T} - \boldsymbol{B}^{+}\left[\frac{\partial \boldsymbol{B}}{\partial q_j}\boldsymbol{B}^{T} + \boldsymbol{B}\left(\frac{\partial \boldsymbol{B}}{\partial q_j}\right)^{T}\right]\right)\left(\boldsymbol{B}\boldsymbol{B}^{T}\right)^{-1} \tag{5.26b}$$

Replacing (5.26b) in (5.20) yields

$$\alpha_2 = 2\sigma\,\mathrm{Tr}\left(\frac{\partial^2 \boldsymbol{B}}{\partial q_j \partial q_i}\boldsymbol{B}^{+} + \frac{\partial \boldsymbol{B}}{\partial q_i}\left\{\left(\frac{\partial \boldsymbol{B}}{\partial q_j}\right)^{T}\right.\right. \tag{5.27}$$
$$\left.\left. -\boldsymbol{B}^{+}\left[\frac{\partial \boldsymbol{B}}{\partial q_j}\boldsymbol{B}^{T} + \boldsymbol{B}\left(\frac{\partial \boldsymbol{B}}{\partial q_j}\right)^{T}\right]\right\}\left(\boldsymbol{B}\boldsymbol{B}^{T}\right)^{-1}\right)$$

Finally, inserting (5.27) into (5.18) gives the total expression

$$\sigma''_{ij} = 2\frac{\partial \sigma}{\partial q_j} \operatorname{Tr}\left(\frac{\partial \boldsymbol{B}}{\partial q_i}\boldsymbol{B}^{+}\right) + 2\sigma \operatorname{Tr}\left(\frac{\partial^2 \boldsymbol{B}}{\partial q_j \partial q_i}\boldsymbol{B}^{+} + \frac{\partial \boldsymbol{B}}{\partial q_i}\left\{\left(\frac{\partial \boldsymbol{B}}{\partial q_j}\right)^{T}\right.\right. \\ \left.\left. -\boldsymbol{B}^{+}\left[\frac{\partial \boldsymbol{B}}{\partial q_j}\boldsymbol{B}^{T} + \boldsymbol{B}\left(\frac{\partial \boldsymbol{B}}{\partial q_j}\right)^{T}\right]\right\}\left(\boldsymbol{B}\boldsymbol{B}^{T}\right)^{-1}\right) \tag{5.28}$$

This expression can be used to compute all missing elements of the matrix in (5.15c) and thus the task Jacobian derivative.

5.4 Symbolic computations

The actuator configuration matrix, $\boldsymbol{B}(\boldsymbol{q})$, can be computed according to the algorithm given in [29]. However, this computation provides a numeric matrix which is not well suited for computation of the partial derivatives of $\boldsymbol{B}(\boldsymbol{q})$. Thus, $\boldsymbol{B}(\boldsymbol{q})$ is instead found as a symbolic matrix, which can then be used for further differentiation to find $\frac{\partial \boldsymbol{B}}{\partial q_i}$ and $\frac{\partial}{\partial q_j}\left(\frac{\partial \boldsymbol{B}}{\partial q_i}\right)$.

As matrix B depends on the individual configuration of the robot, it is not possible to derive a general analytical expression for the partial derivatives. The implementation will therefore make use of the Matlab symbolic toolbox. The symbolic computations are tailored to the specific AIAUV configuration, and must be computed in advance of the simulation due to the complexity of the computations.

Preparation

Before simulating the AIAUV, the actuator configuration matrix and its derivatives must be computed specifically for the chosen manipulator configuration. This depends on thruster position, orientation and the number of joints for the snake robot. The matrices are then computed according to algorithm 1, where the final products are symbolic functions dB.m and dBB.m. The arguments passed to these functions are the current joint coordinates, such that the derivatives can be computed at every time-step during the simulation.

Algorithm 1 Symbolic computations

Initialize Snake Model
Define symbols for $\boldsymbol{q}$
Compute symbolic $\boldsymbol{J}$
Compute symbolic $\boldsymbol{B}$

$\partial \boldsymbol{B} \leftarrow [size(\boldsymbol{B}), n_q]$
$\partial\partial \boldsymbol{B} \leftarrow [size(\boldsymbol{B}), n_q, n_q]$

for i = 1:n **do**
  $\partial \boldsymbol{B}(:,:,i) \leftarrow diff(\boldsymbol{B}, q_i)$
end for
for i = 1:n **do**
  for j = 1:n **do**
    $\partial\partial \boldsymbol{B}(:,:,i,j) \leftarrow diff(\partial \boldsymbol{B}(:,:,i), q_j)$
  end for
end for

Generate symbolic function for $\partial \boldsymbol{B}$ and $\partial\partial \boldsymbol{B}$

5.5 Set-based tasks for singularity avoidance

The control torques for two equality based tasks in the operational space control framework is given in (3.18). However, enforcing an equality constraint on all control tasks is not always required or necessary. For the actuation index the goal is to ensure that the control task is able to stay within some feasible set, and to perform no direct action while the task is adhering to this set constraint. This is known as a set-based task, where the control input is dependent on whether the system is within the feasible set. A few techniques of implementing set-based tasks have been introduced, and this section will outline the framework introduced in [27].

$$D = \{\sigma \in \mathbb{R} \quad | \quad \sigma_{min} \leq \sigma \leq \sigma_{max}\} \tag{5.29}$$

Provided that $\sigma \in D$, the set-based task is considered inactive and should have no effect on the system. Whenever $\sigma \notin D$ the task should be frozen. This is implemented by considering the set-based task as an equality task which is switched on and off as the system moves in and out of D. Determining if the

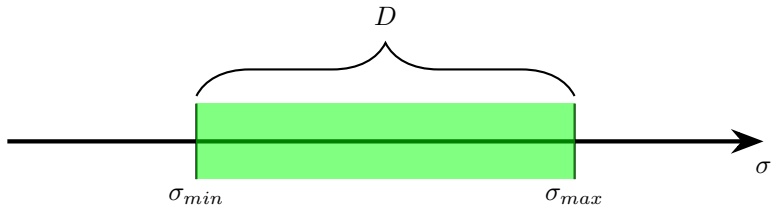

Figure 5.1: Feasible area for set D (Figure adapted from [27]).

task should be frozen or not is done by the extended tangent cone function in (5.30). The function returns true or false as illustrated in fig. 5.2.

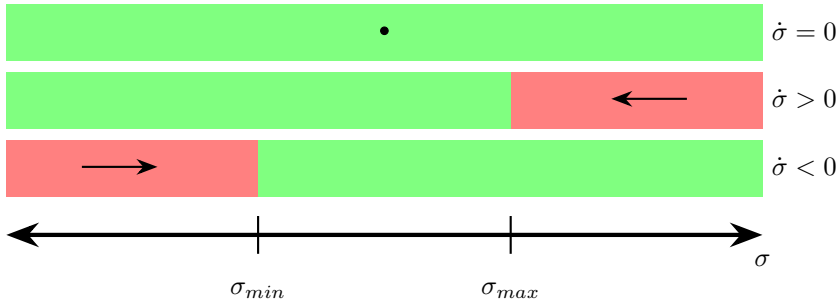

Figure 5.2: Visualization of the function in_T_RC (Figure adapted from [27]).

$$T_{\mathbb{R},D}(\sigma) = \begin{cases} [0,\infty), & \sigma \leq \sigma_{min} \\ \mathbb{R}, & \sigma \in (\sigma_{min}, \sigma_{max}) \\ (-\infty, 0], & \sigma \geq \sigma_{max} \end{cases} \tag{5.30}$$

For $\dot{\sigma} \in T_{\mathbb{R},D} \forall t$ it holds that $\sigma \in \mathbb{R}$ will either remain constant or converge to D. The extended tangent cone is implemented according to algorithm 2 as shown in [27].

Algorithm 2 The Boolean function in_T_RC [27]

if $\sigma_{min} < \sigma < \sigma_{max}$ **then**
  return True
else if $\sigma \leq \sigma_{min}$ and $\dot{\sigma} \geq 0$ **then**
  return True
else if $\sigma \leq \sigma_{min}$ and $\dot{\sigma} < 0$ **then**
  return False
else if $\sigma \geq \sigma_{max}$ and $\dot{\sigma} \leq 0$ **then**
  return True
else
  return True
end if

Chapter 6

Control Tasks

The operational space formulation enables the use of multiple tasks in a hierarchical control structure for the AIAUV. Thus, the redundant degrees of freedom may be exploited to simultaneously achieve multiple control objectives. In this thesis, three different control objectives are used for simulations within the operational space; namely the end-effector position and orientation; joint angle configuration; and the actuation index task introduced in chapter 5. This chapter derives the Jacobians and coordinate transformations necessary to implement these tasks for simulation.

6.1 End-effector position and orientation

The AIAUV aims to use tools to perform maintenance and repair operations where the end-effector control is required to be precise and quick. As a result, control of the end-effector position and orientation is an important problem for the AIAUV.

Position is easily expressed in x, y, z-coordinates in the inertial frame. However, orientation and rotations may be expressed in a number of ways, but in manipulator robotics Euler angles and quaternions are often used. The Euler angle representation consists of three parameters while the quaternion representation has four. However, the Euler angle representation has a singularity which the quaternions avoid through the extra parameter. As a result, the quaternion

representation will be used in the simulations conducted for this thesis. However, this requires some adaptation of the simulator and tailoring to work with the operational space formulation. This section will present these alterations and shows how the end-effector task is implemented within the operational space formulation.

6.1.1 End-effector task Jacobian

From the snake robot model, recursive relationships for the body Jacobians are given in eqs. (2.11) and (2.12). These Jacobians map the linear and angular body and joint velocities $\boldsymbol{\zeta}^i_{b/i}$ to linear and angular velocities $\boldsymbol{\nu}^i_{n_i/i}$ in the respective frames of the manipulator joints as

$$\boldsymbol{\nu}^i_{n_i/i} = \boldsymbol{J}_{n_i}\boldsymbol{\zeta}^i_{b/i} \tag{6.1}$$

By denoting the Jacobian at the end of the recursive relationship as $\boldsymbol{J}_h$, which maps the body and joint velocities of the base frame to linear and angular velocities of the head frame, the following relationship is given

$$\boldsymbol{\nu}^i_{h/i} = \begin{bmatrix} \boldsymbol{v}^h_{h/i} \\ \boldsymbol{\omega}^h_{h/i} \end{bmatrix} = \boldsymbol{J}_h\boldsymbol{\zeta}^i_{b/i} \tag{6.2}$$

From [35], the angular velocity transform using the quaternion representation is

$$\dot{\boldsymbol{\eta}}^i_{h/i} = \begin{bmatrix} \boldsymbol{v}^i_{h/i} \\ \dot{\boldsymbol{p}}_{ih} \end{bmatrix} = \underbrace{\begin{bmatrix} \boldsymbol{R}^i_h(\boldsymbol{p}_{ih}) & \boldsymbol{0}_{3\times 3} \\ \boldsymbol{0}_{4\times 3} & \boldsymbol{T}_p(\boldsymbol{p}_{ih}) \end{bmatrix}}_{\boldsymbol{J_p(p_{ih})}} \begin{bmatrix} \boldsymbol{v}^h_{h/i} \\ \boldsymbol{\omega}^h_{h/i} \end{bmatrix} \tag{6.3}$$

where

$$\boldsymbol{R}^i_h(\boldsymbol{p}_{ih}) = \begin{bmatrix} 1-2\left(\boldsymbol{\epsilon}_2^2+\boldsymbol{\epsilon}_3^2\right) & 2\left(\boldsymbol{\epsilon}_1\boldsymbol{\epsilon}_2-\boldsymbol{\epsilon}_3\eta\right) & 2\left(\boldsymbol{\epsilon}_1\boldsymbol{\epsilon}_3+\boldsymbol{\epsilon}_2\eta\right) \\ 2\left(\boldsymbol{\epsilon}_1\boldsymbol{\epsilon}_2+\boldsymbol{\epsilon}_3\eta\right) & 1-2\left(\boldsymbol{\epsilon}_1^2+\boldsymbol{\epsilon}_3^2\right) & 2\left(\boldsymbol{\epsilon}_2\boldsymbol{\epsilon}_3-\boldsymbol{\epsilon}_1\eta\right) \\ 2\left(\boldsymbol{\epsilon}_1\boldsymbol{\epsilon}_3-\boldsymbol{\epsilon}_2\eta\right) & 2\left(\boldsymbol{\epsilon}_2\boldsymbol{\epsilon}_3+\boldsymbol{\epsilon}_1\eta\right) & 1-2\left(\boldsymbol{\epsilon}_1^2+\boldsymbol{\epsilon}_2^2\right) \end{bmatrix} \tag{6.4}$$

and

$$\boldsymbol{T}_p(\boldsymbol{p}_{ih}) = \frac{1}{2}\begin{bmatrix} -\boldsymbol{\epsilon}^T \\ \eta\boldsymbol{I}_{3\times 3} + \boldsymbol{S}(\boldsymbol{\epsilon}) \end{bmatrix} \tag{6.5}$$

where $\boldsymbol{S}(\cdot)$ denotes the skew-symmetric matrix given by

$$\boldsymbol{S}(\boldsymbol{x}) = \begin{bmatrix} 0 & -x_3 & x_2 \\ x_3 & 0 & -x_1 \\ -x_2 & x_1 & 0 \end{bmatrix} \tag{6.6}$$

By combining eqs. (6.2) and (6.3), the Jacobian $\boldsymbol{J}_{ph} \in \mathbb{R}^{7\times(n+5)}$ can be found. It transforms linear and angular velocities $\boldsymbol{\nu}^b_{b/i}$ and joint angles $\boldsymbol{q}$ to linear velocities $\boldsymbol{v}^i_{h/i}$ and quaternion velocity $\boldsymbol{p}_{ih}$.

$$\dot{\boldsymbol{\eta}}^i_{h/i} = \underbrace{\boldsymbol{J}_p(\boldsymbol{p_{ih}})\boldsymbol{J}_h}_{\boldsymbol{J}_{ph}(\boldsymbol{p_{ih}})} \boldsymbol{\zeta}^b_{b/i} \tag{6.7}$$

In addition to being important for finding $\dot{\boldsymbol{\eta}}^i_{h/i}$ when computing the error coordinates, $\boldsymbol{J}_{ph}$ is also the task specific Jacobian for the head frame used in the operational space formulation. The Jacobian is used to compute the operational space dynamics, the task specific torque and it is vital for the controller.

6.1.2 Jacobian derivative

The operational space formulation requires the derivative of the task specific Jacobian to compute the operational space dynamics, as specified in (3.13b). This can be found by applying the product rule (A.5a) to the following

$$\boldsymbol{J}_{ph}(\boldsymbol{p}) = \boldsymbol{J}_p(\boldsymbol{p})\boldsymbol{J}_h \tag{6.8a}$$

$$\frac{d}{dt}\left(\boldsymbol{J}_{ph}(\boldsymbol{p})\right) = \frac{d}{dt}\left(\boldsymbol{J}_p(\boldsymbol{p})\right)\boldsymbol{J}_h + \boldsymbol{J}_p(\boldsymbol{p})\frac{d}{dt}\left(\boldsymbol{J}_h\right) \tag{6.8b}$$

where $\frac{d}{dt}\left(\boldsymbol{J}_h\right)$ is the final element of the recursion in (2.12b). As for $\frac{d}{dt}\left(\boldsymbol{J}_p(\boldsymbol{p})\right)$, it must be found by direct differentiation of elements in (6.3). From [35], the derivative of a rotation matrix is

$$\frac{d}{dt}\left(\boldsymbol{R}^i_h\right) = \boldsymbol{R}^i_h \boldsymbol{S}\left(\boldsymbol{\omega}^h_{i/h}\right) \tag{6.9}$$

The second element can be differentiated directly in (6.5), which becomes

$$\frac{d}{dt}\left(\boldsymbol{T}_p(\dot{\boldsymbol{p}}_{ih})\right) = \frac{1}{2}\begin{bmatrix} -\dot{\boldsymbol{\epsilon}}^T \\ \dot{\eta}\boldsymbol{I}_{3\times3} + \boldsymbol{S}(\dot{\boldsymbol{\epsilon}}) \end{bmatrix} \tag{6.10}$$

Combining eqs. (6.9) and (6.10), $\frac{d}{dt}\left(\boldsymbol{J}_p\right)$ becomes

$$\frac{d}{dt}\left(\boldsymbol{J}_p(\boldsymbol{p}_{ih})\right) = \begin{bmatrix} \boldsymbol{R}_h^i(\boldsymbol{p}_{ih})\boldsymbol{S}\left(\boldsymbol{\omega}_{i/h}^h\right) & \boldsymbol{0}_{3\times 3} \\ \boldsymbol{0}_{4\times 3} & \frac{d}{dt}\left(\boldsymbol{T}_p(\dot{\boldsymbol{p}}_{ih})\right) \end{bmatrix} \tag{6.11}$$

The derivative of $\boldsymbol{J}_{ph}(\boldsymbol{p})$ can then be found from (6.8b).

6.1.3 Control Jacobian

The unit quaternion is designed such that $\|\boldsymbol{p}\| = 1$. As a result, whenever the vector part of the quaternion $\boldsymbol{\epsilon} \to \boldsymbol{0}$, the scalar factor $\eta \to \pm 1$. Thus the scalar factor does not have to be controlled, and the control algorithms described later will only control the vector part of the quaternion. This also means that the Jacobians derived in section 6.1.1 and section 6.1.2 have the wrong dimensions. The solution is simply to remove the row of the Jacobians corresponding to the scalar factor η, but the formulas are shown below.

$$\boldsymbol{J}_p(\boldsymbol{p}_{ih}) = \begin{bmatrix} \boldsymbol{R}_h^i(\boldsymbol{p}_{ih}) & \boldsymbol{0}_{3\times 3} \\ \boldsymbol{0}_{3\times 3} & \frac{1}{2}\left(\eta_{ih}\boldsymbol{I}_{3\times 3} + \boldsymbol{S}(\boldsymbol{\epsilon}_{ih})\right) \end{bmatrix} \tag{6.12}$$

And its derivative

$$\frac{d}{dt}\left(\boldsymbol{J}_p(\boldsymbol{p}_{ih})\right) = \begin{bmatrix} \boldsymbol{R}_h^i(\boldsymbol{p}_{ih})\boldsymbol{S}\left(\boldsymbol{\omega}_{i/h}^h\right) & \boldsymbol{0}_{3\times 3} \\ \boldsymbol{0}_{3\times 3} & \frac{1}{2}\left(\dot{\eta}_{ih}\boldsymbol{I}_{3\times 3} + \boldsymbol{S}(\dot{\boldsymbol{\epsilon}}_{ih})\right) \end{bmatrix} \tag{6.13}$$

The other Jacobians are still the same, but $\boldsymbol{J}_{ph}(\boldsymbol{p}_{ih})$ and $\frac{d}{dt}\left(\boldsymbol{J}_{ph}(\boldsymbol{p}_{ih})\right)$ are altered by the change in the Jacobians above. This change corresponds to removing the row corresponding to the scalar factor, and the resulting dimension is $\boldsymbol{J}_{ph}(\boldsymbol{p}_{ih}) \in \mathbb{R}^{6\times(n+5)}$.

6.1.4 Quaternion error coordinates

In order to perform control, the error coordinates are required. The goal is for $\tilde{\boldsymbol{\eta}}_{h/i}^i \to 0$. The position error can be easily found by subtracting the desired position as

$$\tilde{\boldsymbol{t}}_{h/i}^i = \boldsymbol{t}_{h/i}^i - \boldsymbol{t}_d \tag{6.14}$$

However, the quaternion error behaves slightly differently. From [31] and [36], the error quaternion given a desired orientation $\boldsymbol{p}_d$ is

$$\tilde{\boldsymbol{p}}_{ih} = \boldsymbol{p}_d^{-1} \otimes \boldsymbol{p}_{ih} \tag{6.15a}$$

$$= \begin{bmatrix} \eta_d & \boldsymbol{\epsilon}_d^T \\ -\boldsymbol{\epsilon}_d & \eta_d \boldsymbol{I} - \boldsymbol{S}(\boldsymbol{\epsilon}_d) \end{bmatrix} \begin{bmatrix} \eta \\ \boldsymbol{\epsilon} \end{bmatrix} \tag{6.15b}$$

and the quaternion derivative error is

$$\dot{\tilde{\boldsymbol{p}}}_{ih} = \frac{1}{2} \boldsymbol{T}_p(\tilde{\boldsymbol{p}}_{ih}) \tilde{\boldsymbol{\omega}}_{h/i}^h \tag{6.16}$$

Here, $\boldsymbol{T}_p$ is given in (6.5) and the angular velocity error is $\tilde{\boldsymbol{\omega}}_{h/i}^h = \boldsymbol{\omega}_{h/i}^h - \boldsymbol{\omega}_d$.

As with the task specific control Jacobian in section 6.1.3, the end-effector control vector must also be changed to support the removal of the scalar factor. Thus, when computing control torques in the simulations, the vector used is

$$\boldsymbol{\eta}_{h/i}^i = \begin{bmatrix} x & y & z & \epsilon_1 & \epsilon_2 & \epsilon_3 \end{bmatrix}^T \tag{6.17}$$

such that the scalar factor η does not affect the control torque computation.

6.1.5 End-effector operational space control

Combining the task specific Jacobian (6.7) and its derivative (6.8b) with the alterations described in section 6.1.3, the operational space matrices (3.13) for the end-effector task become:

$$\boldsymbol{M}_e = \left(\boldsymbol{J}_{ph} \boldsymbol{M}^{-1} \boldsymbol{J}_{ph}^T \right)^{-1} \in \mathbb{R}^{m_e \times m_e} \tag{6.18a}$$

$$\boldsymbol{c}_e = \boldsymbol{M}_e \left[\boldsymbol{J}_{ph} \boldsymbol{M}^{-1} \boldsymbol{C} \boldsymbol{\zeta}_{b/i}^b - \frac{d}{dt} \left(\boldsymbol{J}_{ph} \right) \boldsymbol{\zeta}_{b/i}^b \right] \tag{6.18b}$$

$$\boldsymbol{d}_e = \boldsymbol{M}_e \boldsymbol{J}_{ph} \boldsymbol{M}^{-1} \boldsymbol{D} \boldsymbol{\zeta}_{b/i}^b \tag{6.18c}$$

$$\boldsymbol{g}_e = \boldsymbol{M}_e \boldsymbol{J}_{ph} \boldsymbol{M}^{-1} \boldsymbol{g} \tag{6.18d}$$

6.2 Joint angles

To perform end-effector tasks in tight spaces, the AIAUV is required to control the occupancy of its entire body. The manipulator consists of 1D revolute

joints, meaning the position and orientation of the manipulator links can be decided based on the end-effector position and orientation and the angles of the revolute joints. The joint angles can be controlled as a separate task within the operational space framework.

6.2.1 Joint angle task Jacobian

By examining the system velocity vector $\boldsymbol{\zeta}_{b/i}^{b}$ in (2.6), the task specific Jacobian can easily be found as

$$\dot{\boldsymbol{q}} = \boldsymbol{J}_q \boldsymbol{\zeta}_{b/i}^{b} \tag{6.19a}$$

$$\boldsymbol{J}_q = \begin{bmatrix} \boldsymbol{0}_{(n-1)\times 6} & \boldsymbol{I}_{(n-1)\times(n-1)} \end{bmatrix} \tag{6.19b}$$

where $\boldsymbol{I}$ is the identity matrix. Since the Jacobian is a constant matrix, the derivative is

$$\frac{d}{dt}\left(\boldsymbol{J}_q\right) = \boldsymbol{0}_{(n-1)\times(n+5)} \tag{6.20}$$

6.2.2 Joint angle operational space control

Combining the task specific Jacobian (6.19b) and its derivative (6.20), the operational space matrices (3.13) for the joint angles task become:

$$\boldsymbol{M}_q = \left(\boldsymbol{J}_q \boldsymbol{M}^{-1} \boldsymbol{J}_q^T\right)^{-1} \in \mathbb{R}^{m_q \times m_q} \tag{6.21a}$$

$$\boldsymbol{c}_q = \boldsymbol{M}_q \boldsymbol{J}_q \boldsymbol{M}^{-1} \boldsymbol{C} \boldsymbol{\zeta}_{b/i}^{b} \tag{6.21b}$$

$$\boldsymbol{d}_q = \boldsymbol{M}_q \boldsymbol{J}_q \boldsymbol{M}^{-1} \boldsymbol{D} \boldsymbol{\zeta}_{b/i}^{b} \tag{6.21c}$$

$$\boldsymbol{g}_q = \boldsymbol{M}_q \boldsymbol{J}_q \boldsymbol{M}^{-1} \boldsymbol{g} \tag{6.21d}$$

6.3 Actuation index

The configuration of the AIAUV used in the simulations for this thesis has an actuator singularity in I-shape. As an attempt to avoid actuator singularities, a measure of distance to these singularities is introduced known as the actuation index.

$$\sigma = \det\left(\boldsymbol{B}\boldsymbol{B}^T\right) \tag{6.22}$$

The actuation index is used as the third task for the controller testing. Due to the complexity of the derivations, the actuation index task is separated into a chapter of its own in chapter 5. The main results achieved in chapter 5 is a result of the specialization project and is summarized in this section to pinpoint the most important pieces of chapter 5 for the control task.

6.3.1 Actuation index task Jacobian

The task Jacobian can be found as (5.2)

$$\boldsymbol{J}_\sigma = \begin{bmatrix} \mathbf{0}_{1\times 6} & \frac{\partial \sigma}{\partial q_1} & \frac{\partial \sigma}{\partial q_2} & \frac{\partial \sigma}{\partial q_3} & \cdots & \frac{\partial \sigma}{\partial q_{n_q}} \end{bmatrix} \tag{6.23}$$

where each individual element is given by (5.14) as

$$\frac{\partial \sigma}{\partial q_i} = 2\sigma \operatorname{Tr}\left(\frac{\partial \boldsymbol{B}}{\partial q_i} \boldsymbol{B}^+ \right) \tag{6.24}$$

6.3.2 Jacobian derivative

The task Jacobian derivative can be found through the product in (5.15c)

$$\frac{d}{dt}\left(\boldsymbol{J}_\sigma\right) = \begin{bmatrix} \mathbf{0}_{6\times 1} \\ \dot{q}_1 \\ \dot{q}_2 \\ \dot{q}_3 \\ \vdots \\ \dot{q}_{n_q} \end{bmatrix}^T \begin{bmatrix} & \mathbf{0}_{6\times n_q} & \\ \frac{\partial^2 \sigma}{\partial q_1^2} & \frac{\partial^2 \sigma}{\partial q_2 \partial q_1} & \frac{\partial^2 \sigma}{\partial q_3 \partial q_1} & \cdots & \frac{\partial^2 \sigma}{\partial q_{n_q} \partial q_1} \\ \frac{\partial^2 \sigma}{\partial q_1 \partial q_2} & \frac{\partial^2 \sigma}{\partial q_2^2} & \frac{\partial^2 \sigma}{\partial q_3 \partial q_2} & \cdots & \frac{\partial^2 \sigma}{\partial q_{n_q} \partial q_2} \\ \frac{\partial^2 \sigma}{\partial q_1 \partial q_3} & \frac{\partial^2 \sigma}{\partial q_2 \partial q_3} & \frac{\partial^2 \sigma}{\partial q_3^2} & \cdots & \frac{\partial^2 \sigma}{\partial q_{n_q} \partial q_3} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ \frac{\partial^2 \sigma}{\partial q_1 \partial q_{n_q}} & \frac{\partial^2 \sigma}{\partial q_2 \partial q_{n_q}} & \frac{\partial^2 \sigma}{\partial q_3 \partial q_{n_q}} & \cdots & \frac{\partial^2 \sigma}{\partial q_{n_q}^2} \end{bmatrix} \tag{6.25}$$

where the elements of the right-hand matrix is (5.28)

$$\begin{aligned} \sigma''_{ij} = 2\frac{\partial \sigma}{\partial q_j} \operatorname{Tr}\left(\frac{\partial \boldsymbol{B}}{\partial q_i} \boldsymbol{B}^+ \right) + 2\sigma \operatorname{Tr}\Bigg(& \frac{\partial^2 \boldsymbol{B}}{\partial q_j \partial q_i} \boldsymbol{B}^+ + \frac{\partial \boldsymbol{B}}{\partial q_i} \Bigg\{ \left(\frac{\partial \boldsymbol{B}}{\partial q_j} \right)^T \\ & - \boldsymbol{B}^+ \left[\frac{\partial \boldsymbol{B}}{\partial q_j} \boldsymbol{B}^T + \boldsymbol{B} \left(\frac{\partial \boldsymbol{B}}{\partial q_j} \right)^T \right] \Bigg\} \left(\boldsymbol{B}\boldsymbol{B}^T \right)^{-1} \Bigg) \end{aligned} \tag{6.26}$$

The derivatives of the actuator configuration matrix $\boldsymbol{B}$ is found through symbolic computations as described in section 5.4.

6.3.3 Actuation index operational space control

Combining the task specific Jacobian (5.2) and its derivative (5.15c), the operational space matrices (3.13) for the actuation index task become:

$$\boldsymbol{M}_\sigma = \left(\boldsymbol{J}_\sigma \boldsymbol{M}^{-1} \boldsymbol{J}_\sigma^T\right)^{-1} \in \mathbb{R}^{m_\sigma \times m_\sigma} \tag{6.27a}$$

$$\boldsymbol{c}_\sigma = \boldsymbol{M}_\sigma \left[\boldsymbol{J}_\sigma \boldsymbol{M}^{-1} \boldsymbol{C} \boldsymbol{\zeta}_{b/i}^b - \frac{d}{dt}\left(\boldsymbol{J}_\sigma\right) \boldsymbol{\zeta}_{b/i}^b\right] \tag{6.27b}$$

$$\boldsymbol{d}_\sigma = \boldsymbol{M}_\sigma \boldsymbol{J}_\sigma \boldsymbol{M}^{-1} \boldsymbol{D} \boldsymbol{\zeta}_{b/i}^b \tag{6.27c}$$

$$\boldsymbol{g}_\sigma = \boldsymbol{M}_\sigma \boldsymbol{J}_\sigma \boldsymbol{M}^{-1} \boldsymbol{g} \tag{6.27d}$$

Chapter 7

Sliding Mode Control

The following sections describe the sliding mode controllers implementation, where control gains chosen through tuning is presented for every control task. In addition, the full control torque assignment is shown for every task.

7.1 Super-twisting algorithm with adaptive gains

To control the end-effector, actuation index and joint angles as described in chapter 6, a super-twisting algorithm with adaptive gains (STA) controller is implemented for every task according to eqs. (4.5), (4.8) and (4.9).

The parameters $\boldsymbol{\alpha}$ is chosen as in (4.9), and initialized to zero. The parameter $\boldsymbol{\beta}_{STA}$ is defined by (4.8), and therefore does not have to be initialized. The control gains outlined in eqs. (4.5a), (4.5b), (4.8) and (4.9) are chosen for every control task separately, meaning three controllers are designed.

7.1.1 End-effector

The controller gains for the end-effector STA controller are chosen as

$$\boldsymbol{\alpha}_{m_e} = \begin{bmatrix} 0.005 & 0.005 & 0.005 & 0.005 & 0.005 & 0.005 \end{bmatrix}^T \tag{7.1a}$$

$$\boldsymbol{\epsilon}_{1_e} = \begin{bmatrix} 0.0001 & 0.0001 & 0.0001 & 0.0001 & 0.0001 & 0.0001 \end{bmatrix}^T \tag{7.1b}$$

$$\boldsymbol{\omega}_{1_e} = \begin{bmatrix} 8 & 8 & 8 & 8 & 8 & 8 \end{bmatrix}^T \tag{7.1c}$$

$$\boldsymbol{\lambda}_{1_e} = \begin{bmatrix} 0.1 & 0.1 & 0.1 & 0.1 & 0.1 & 0.1 \end{bmatrix}^T \tag{7.1d}$$

$$\boldsymbol{\gamma}_{1_e} = \begin{bmatrix} 1 & 1 & 1 & 1 & 1 & 1 \end{bmatrix}^T \tag{7.1e}$$

The end-effector torque assignment, where $\boldsymbol{J}_{ph}$ and $\tilde{\boldsymbol{\eta}}^i_{i/h}$ is without the scalar factor as described in sections 6.1.3 and 6.1.4, is given by eqs. (3.10), (4.3), (4.5), (4.8) and (4.9) as

$$\boldsymbol{\tau}_e = \boldsymbol{J}_{ph}^T \boldsymbol{f}_e \tag{7.2a}$$

$$\boldsymbol{s}_e = \tilde{\boldsymbol{\eta}}^i_{i/h} + \dot{\tilde{\boldsymbol{\eta}}}^i_{i/h} \tag{7.2b}$$

$$\boldsymbol{f}_e = -\boldsymbol{\alpha}_e |\boldsymbol{s}_e|^{\frac{1}{2}} \circ sgn(\boldsymbol{s}_e) + \boldsymbol{v}_e \tag{7.2c}$$

$$\dot{\boldsymbol{v}}_e = -\boldsymbol{\beta}_{STA_e} \circ sgn(\boldsymbol{s}_e) \tag{7.2d}$$

$$\dot{\boldsymbol{\alpha}}_e = \begin{cases} \boldsymbol{\omega}_{1_e} \circ \sqrt{\frac{\boldsymbol{\gamma}_{1_e}}{2}} & \text{if } |\boldsymbol{s}_e| > \alpha_{m_e} \\ 0 & \text{if } |\boldsymbol{s}_e| \leq \alpha_{m_e} \end{cases} \tag{7.2e}$$

$$\boldsymbol{\beta}_{STA_e} = 2\boldsymbol{\epsilon}_{1_e} \circ \boldsymbol{\alpha}_e + \boldsymbol{\lambda}_{1_e} + 4\boldsymbol{\epsilon}_{1_e} \circ \boldsymbol{\epsilon}_{1_e} \tag{7.2f}$$

where the task specific Jacobian is given in (6.7).

7.1.2 Joint angles

The controller gains for the joint angles STA controller are chosen as

$$\boldsymbol{\alpha}_{m_q} = \begin{bmatrix} 0.005 & 0.005 & 0.005 & 0.005 & 0.005 & 0.005 & 0.005 & 0.005 \end{bmatrix}^T \tag{7.3a}$$

$$\boldsymbol{\epsilon}_{1_q} = \begin{bmatrix} 0.0001 & 0.0001 & 0.0001 & 0.0001 & 0.0001 & 0.0001 & 0.0001 & 0.0001 \end{bmatrix}^T \tag{7.3b}$$

$$\boldsymbol{\omega}_{1_q} = \begin{bmatrix} 3.5 & 3.5 & 3.5 & 3.5 & 3.5 & 3.5 & 3.5 & 3.5 \end{bmatrix}^T \tag{7.3c}$$

$$\boldsymbol{\lambda}_{1_q} = \begin{bmatrix} 0.1 & 0.1 & 0.1 & 0.1 & 0.1 & 0.1 & 0.1 & 0.1 \end{bmatrix}^T \tag{7.3d}$$

$$\boldsymbol{\gamma}_{1_q} = \begin{bmatrix} 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \end{bmatrix}^T \tag{7.3e}$$

where the torque assignment for the joint angles controller is given by eqs. (3.10), (4.3), (4.5), (4.8) and (4.9) as

$$\boldsymbol{\tau}_q = \boldsymbol{J}_q^T \boldsymbol{f}_q \tag{7.4a}$$
$$\boldsymbol{s}_q = \tilde{\boldsymbol{q}} + \dot{\tilde{\boldsymbol{q}}} \tag{7.4b}$$
$$\boldsymbol{f}_q = -\boldsymbol{\alpha}_q |\boldsymbol{s}_q|^{\frac{1}{2}} \circ sgn(\boldsymbol{s}_q) + \boldsymbol{v}_q \tag{7.4c}$$
$$\dot{\boldsymbol{v}}_q = -\boldsymbol{\beta}_{STA_q} \circ sgn(\boldsymbol{s}_q) \tag{7.4d}$$
$$\dot{\boldsymbol{\alpha}}_q = \begin{cases} \boldsymbol{\omega}_{1_q} \circ \sqrt{\frac{\gamma_{1_q}}{2}} & \text{if } |\boldsymbol{s}_q| > \alpha_{m_q} \\ 0 & \text{if } |\boldsymbol{s}_q| \leq \alpha_{m_q} \end{cases} \tag{7.4e}$$
$$\boldsymbol{\beta}_{STA_q} = 2\boldsymbol{\epsilon}_{1_q} \circ \boldsymbol{\alpha}_q + \boldsymbol{\lambda}_{1_q} + 4\boldsymbol{\epsilon}_{1_q} \circ \boldsymbol{\epsilon}_{1_q} \tag{7.4f}$$

where the task specific Jacobian is given in (6.19b).

7.1.3 Actuation index

The controller gains for the actuation index STA controller are chosen as

$$\boldsymbol{\alpha}_{m_\sigma} = 0.005 \tag{7.5a}$$
$$\boldsymbol{\epsilon}_{1_\sigma} = 0.0001 \tag{7.5b}$$
$$\boldsymbol{\omega}_{1_\sigma} = 20 \tag{7.5c}$$
$$\boldsymbol{\lambda}_{1_\sigma} = 2 \tag{7.5d}$$
$$\boldsymbol{\gamma}_{1_\sigma} = 2 \tag{7.5e}$$

where the torque assignment for the actuation index controller is given by eqs. (3.10), (4.3), (4.5), (4.8) and (4.9) as

$$\boldsymbol{\tau}_\sigma = \boldsymbol{J}_\sigma^T \boldsymbol{f}_\sigma \tag{7.6a}$$
$$\boldsymbol{s}_\sigma = \tilde{\boldsymbol{\sigma}} + \dot{\tilde{\boldsymbol{\sigma}}} \tag{7.6b}$$
$$\boldsymbol{f}_\sigma = -\boldsymbol{\alpha}_\sigma |\boldsymbol{s}_\sigma|^{\frac{1}{2}} \circ sgn(\boldsymbol{s}_\sigma) + \boldsymbol{v}_\sigma \tag{7.6c}$$
$$\dot{\boldsymbol{v}}_\sigma = -\boldsymbol{\beta}_{STA_\sigma} \circ sgn(\boldsymbol{s}_\sigma) \tag{7.6d}$$
$$\dot{\boldsymbol{\alpha}}_\sigma = \begin{cases} \boldsymbol{\omega}_{1_\sigma} \circ \sqrt{\frac{\gamma_{1_\sigma}}{2}} & \text{if } |\boldsymbol{s}_\sigma| > \alpha_{m_\sigma} \\ 0 & \text{if } |\boldsymbol{s}_\sigma| \leq \alpha_{m_\sigma} \end{cases} \tag{7.6e}$$
$$\boldsymbol{\beta}_{STA_\sigma} = 2\boldsymbol{\epsilon}_{1_\sigma} \circ \boldsymbol{\alpha}_\sigma + \boldsymbol{\lambda}_{1_\sigma} + 4\boldsymbol{\epsilon}_{1_\sigma} \circ \boldsymbol{\epsilon}_{1_\sigma} \tag{7.6f}$$

where the task specific Jacobian is given in (6.23).

7.2 Generalized super-twisting algorithm

To control the end-effector, actuation index and joint angles as described in chapter 6, a generalized super-twisting algorithm (GSTA) controller is implemented for every task according to eqs. (4.11a) to (4.11d).

7.2.1 End-effector

The control gains for the end-effector GSTA controller are chosen as

$$\boldsymbol{k}_{1_e} = \begin{bmatrix} 7 & 7 & 7 & 7 & 7 & 7 \end{bmatrix}^T \tag{7.7a}$$

$$\boldsymbol{k}_{2_e} = \begin{bmatrix} 0.06 & 0.06 & 0.06 & 0.06 & 0.06 & 0.06 \end{bmatrix}^T \tag{7.7b}$$

$$\boldsymbol{\beta}_{GSTA_e} = \begin{bmatrix} 15 & 15 & 15 & 15 & 15 & 15 \end{bmatrix}^T \tag{7.7c}$$

From eqs. (3.10), (4.3) and (4.11), the controller torque for the end-effector using a GSTA controller becomes

$$\boldsymbol{\tau}_e = \boldsymbol{J}_{ph}^T \boldsymbol{f}_e \tag{7.8a}$$

$$\boldsymbol{s}_e = \tilde{\boldsymbol{\eta}}_{i/h}^i + \dot{\tilde{\boldsymbol{\eta}}}_{i/h}^i \tag{7.8b}$$

$$\boldsymbol{f}_e = -\boldsymbol{k}_{1_e} \circ \boldsymbol{\phi}_{1_e}(\boldsymbol{s}_e) + \boldsymbol{z}_e \tag{7.8c}$$

$$\dot{\boldsymbol{z}}_e = -\boldsymbol{k}_{2_e} \circ \boldsymbol{\phi}_{2_e}(\boldsymbol{s}_e) \tag{7.8d}$$

$$\boldsymbol{\phi}_{1_e}(\boldsymbol{s}_e) = \lceil \boldsymbol{s}_e \rfloor^{\frac{1}{2}} + \boldsymbol{\beta}_{GSTA_e} \circ \boldsymbol{s}_e \tag{7.8e}$$

$$\boldsymbol{\phi}_{1_e}(\boldsymbol{s}_e) = \frac{1}{2} \lceil \boldsymbol{s}_e \rfloor^0 + \frac{3}{2} \boldsymbol{\beta}_{GSTA_e} \circ \lceil \boldsymbol{s}_e \rfloor^{\frac{3}{2}} + \boldsymbol{\beta}_{GSTA_e}^2 \circ \boldsymbol{s}_e \tag{7.8f}$$

where the task specific Jacobian is given in (6.7).

7.2.2 Joint angles

The control gains for the joint angle GSTA controller are chosen as

$$\boldsymbol{k}_{1_q} = \begin{bmatrix} 0.5 & 0.5 & 0.5 & 0.5 & 0.5 & 0.5 \end{bmatrix}^T \tag{7.9a}$$

$$\boldsymbol{k}_{2_q} = \begin{bmatrix} 2 & 2 & 2 & 2 & 2 & 2 \end{bmatrix}^T \tag{7.9b}$$

$$\boldsymbol{\beta}_{GSTA_q} = \begin{bmatrix} 15 & 15 & 15 & 15 & 15 & 15 \end{bmatrix}^T \tag{7.9c}$$

From eqs. (3.10), (4.3) and (4.11), the controller torque for the joint angles using a GSTA controller becomes

$$\boldsymbol{\tau}_q = \boldsymbol{J}_{ph}^T \boldsymbol{f}_q \tag{7.10a}$$
$$\boldsymbol{s}_q = \tilde{\boldsymbol{q}} + \dot{\tilde{\boldsymbol{q}}} \tag{7.10b}$$
$$\boldsymbol{f}_q = -\boldsymbol{k}_{1_q} \circ \boldsymbol{\phi}_{1_q}(\boldsymbol{s}_q) + \boldsymbol{z}_q \tag{7.10c}$$
$$\dot{\boldsymbol{z}}_q = -\boldsymbol{k}_{2_q} \circ \boldsymbol{\phi}_{2_q}(\boldsymbol{s}_q) \tag{7.10d}$$
$$\boldsymbol{\phi}_{1_q}(\boldsymbol{s}_q) = \lceil \boldsymbol{s}_q \rfloor^{\frac{1}{2}} + \boldsymbol{\beta}_{GSTA_q} \circ \boldsymbol{s}_q \tag{7.10e}$$
$$\boldsymbol{\phi}_{1_q}(\boldsymbol{s}_q) = \frac{1}{2}\lceil \boldsymbol{s}_q \rfloor^0 + \frac{3}{2}\boldsymbol{\beta}_{GSTA_q} \circ \lceil \boldsymbol{s}_q \rfloor^{\frac{3}{2}} + \boldsymbol{\beta}_{GSTA_q}^2 \circ \boldsymbol{s}_q \tag{7.10f}$$

where the task specific Jacobian is given in (6.19b).

7.2.3 Actuation index

The control gains for the actuation index GSTA controller are chosen as

$$\boldsymbol{k}_{1_\sigma} = 5 \tag{7.11a}$$
$$\boldsymbol{k}_{2_\sigma} = 0.2 \tag{7.11b}$$
$$\boldsymbol{\beta}_{GSTA_\sigma} = 20 \tag{7.11c}$$

From eqs. (3.10), (4.3) and (4.11), the controller torque for the actuation index using a GSTA controller becomes

$$\boldsymbol{\tau}_\sigma = \boldsymbol{J}_{ph}^T \boldsymbol{f}_\sigma \tag{7.12a}$$
$$\boldsymbol{s}_\sigma = \tilde{\boldsymbol{\sigma}} + \dot{\tilde{\boldsymbol{\sigma}}} \tag{7.12b}$$
$$\boldsymbol{f}_\sigma = -\boldsymbol{k}_{1_\sigma} \circ \boldsymbol{\phi}_{1_\sigma}(\boldsymbol{s}_\sigma) + \boldsymbol{z}_\sigma \tag{7.12c}$$
$$\dot{\boldsymbol{z}}_\sigma = -\boldsymbol{k}_{2_\sigma} \circ \boldsymbol{\phi}_{2_\sigma}(\boldsymbol{s}_\sigma) \tag{7.12d}$$
$$\boldsymbol{\phi}_{1_\sigma}(\boldsymbol{s}_\sigma) = \lceil \boldsymbol{s}_\sigma \rfloor^{\frac{1}{2}} + \boldsymbol{\beta}_{GSTA_\sigma} \circ \boldsymbol{s}_\sigma \tag{7.12e}$$
$$\boldsymbol{\phi}_{1_\sigma}(\boldsymbol{s}_\sigma) = \frac{1}{2}\lceil \boldsymbol{s}_\sigma \rfloor^0 + \frac{3}{2}\boldsymbol{\beta}_{GSTA_\sigma} \circ \lceil \boldsymbol{s}_\sigma \rfloor^{\frac{3}{2}} + \boldsymbol{\beta}_{GSTA_\sigma}^2 \circ \boldsymbol{s}_\sigma \tag{7.12f}$$

where the task specific Jacobian is given in (6.23).

Chapter 8

PID Control

As a reference to the more advanced nonlinear methods examined, a simple PID-controller is used. The PID-controller is extended from the feedback linearizing PD-controller used with the operational space formulation in [28]. The controller includes dissipative terms required for asymptotic stability as described in [2]. At the end of the chapter, the PID-controller torque for each task, together with control gains, are given.

8.1 Controller synthesis

By choosing $\boldsymbol{f}_x$ as [28]

$$\boldsymbol{f}_x = \boldsymbol{M}_x \left(\ddot{\boldsymbol{\sigma}}_{x,d} - \boldsymbol{K}_i \int \tilde{\boldsymbol{\sigma}}_x dt - \boldsymbol{K}_d \dot{\tilde{\boldsymbol{\sigma}}}_x - \boldsymbol{K}_p \tilde{\boldsymbol{\sigma}}_x \right) + \boldsymbol{c}_x + \boldsymbol{d}_x + \boldsymbol{g}_x \tag{8.1}$$

the operational space dynamics in (3.14) can be converted into a second-order error dynamics system, where the error is given by $\tilde{\boldsymbol{\sigma}}_x = \boldsymbol{\sigma}_x - \boldsymbol{\sigma}_{x,d}$. By inserting (8.1) into (3.14) and using that $\boldsymbol{K}_i > 0$, $\boldsymbol{K}_d > 0$ and $\boldsymbol{K}_p > 0$, the second-order error dynamics can be written as

$$\ddot{\tilde{\boldsymbol{\sigma}}}_x + \boldsymbol{K}_i \int \tilde{\boldsymbol{\sigma}}_x dt + \boldsymbol{K}_d \dot{\tilde{\boldsymbol{\sigma}}}_x + \boldsymbol{K}_p \tilde{\boldsymbol{\sigma}}_x = 0 \tag{8.2}$$

Using (3.10), the controller can be expressed on torque level as

$$\begin{aligned}\boldsymbol{\tau}_x = \boldsymbol{J}_x^T \boldsymbol{M}_x \left(\ddot{\boldsymbol{\sigma}}_{x,d} + \boldsymbol{K}_i \int \tilde{\boldsymbol{\sigma}}_x dt + \boldsymbol{K}_d \dot{\tilde{\boldsymbol{\sigma}}}_x + \boldsymbol{K}_p \tilde{\boldsymbol{\sigma}}_x + F_{rs,x} \right) \\ + \Gamma_x + \boldsymbol{c}'_x + \boldsymbol{d}'_x + \boldsymbol{g}'_x \end{aligned} \tag{8.3}$$

where [28] has defined

$$\boldsymbol{c}'_x = \boldsymbol{J}_x^T \boldsymbol{c}_x \tag{8.4a}$$
$$\boldsymbol{d}'_x = \boldsymbol{J}_x^T \boldsymbol{d}_x \tag{8.4b}$$
$$\boldsymbol{g}'_x = \boldsymbol{J}_x^T \boldsymbol{g}_x \tag{8.4c}$$

and the dissipative terms $F_{rs,x}$ and Γ_x have been added to ensure asymptotic stability as described by Khatib in [2]. $K_{dis,x} > 0$

$$F_{rs,x} = K_{dis,x} \dot{x} \tag{8.5a}$$
$$\Gamma_x = -K_{dis,x} \boldsymbol{\zeta}^b_{b/i} \tag{8.5b}$$

Remark: It should be noted that the PID-controller designed in this section includes compensation terms, while the sliding mode controllers in chapter 7 does not have the luxury of this model knowledge.

8.2 End-effector

The end-effector control gains for the PID-controller are chosen as

$$\boldsymbol{K}_{p_e} = 400 \boldsymbol{I}_{6\times 6} \tag{8.6a}$$
$$\boldsymbol{K}_{d_e} = 20 \boldsymbol{I}_{6\times 6} \tag{8.6b}$$
$$\boldsymbol{K}_{i_e} = 30 \boldsymbol{I}_{6\times 6} \tag{8.6c}$$
$$K_{dis,e} = 4 \tag{8.6d}$$

and the control torque is

$$\begin{aligned}\boldsymbol{\tau}_e = \boldsymbol{J}_{ph}^T \boldsymbol{M}_e \left(\ddot{\boldsymbol{\sigma}}_{e,d} + \boldsymbol{K}_{i_e} \int \tilde{\boldsymbol{\eta}}^i_{i/h} dt + \boldsymbol{K}_{d_e} \dot{\tilde{\boldsymbol{\eta}}}^i_{i/h} + \boldsymbol{K}_{p_e} \tilde{\boldsymbol{\eta}}^i_{i/h} + F_{rs,e} \right) \\ + \Gamma_e + \boldsymbol{c}'_e + \boldsymbol{d}'_e + \boldsymbol{g}'_e \end{aligned} \tag{8.7}$$

where the task specific Jacobian is given in (6.7).

8.3 Joint angles

The joint angles control gains for the PID controller are chosen as

$$\boldsymbol{K}_{p_q} = 25\boldsymbol{I}_{8\times 8} \tag{8.8a}$$
$$\boldsymbol{K}_{d_q} = 2.5\boldsymbol{I}_{8\times 8} \tag{8.8b}$$
$$\boldsymbol{K}_{i_q} = 0.5\boldsymbol{I}_{8\times 8} \tag{8.8c}$$
$$K_{dis,q} = 4 \tag{8.8d}$$

and the control torque is

$$\begin{aligned}\boldsymbol{\tau}_q = \boldsymbol{J}_q^T \boldsymbol{M}_q \left(\ddot{\boldsymbol{q}}_d + \boldsymbol{K}_{i_q} \int \tilde{\boldsymbol{\sigma}}_q dt + \boldsymbol{K}_{d_q} \dot{\tilde{\boldsymbol{q}}} + \boldsymbol{K}_{p_q} \tilde{\boldsymbol{q}} + F_{rs,q} \right) \\ + \Gamma_q + \boldsymbol{c}_q' + \boldsymbol{d}_q' + \boldsymbol{g}_q'\end{aligned} \tag{8.9}$$

where the task specific Jacobian is given in (6.19b).

8.4 Actuation index

The actuation index control gains for the PID-controller are chosen as

$$\boldsymbol{K}_{\boldsymbol{p_\sigma}} = 30 \tag{8.10a}$$
$$\boldsymbol{K}_{\boldsymbol{d_\sigma}} = 2 \tag{8.10b}$$
$$\boldsymbol{K}_{\boldsymbol{i_\sigma}} = 2 \tag{8.10c}$$
$$K_{dis,\sigma} = 4 \tag{8.10d}$$

and the control torque is

$$\begin{aligned}\boldsymbol{\tau}_\sigma = \boldsymbol{J}_\sigma^T \boldsymbol{M}_\sigma \left(\ddot{\boldsymbol{\sigma}}_d + \boldsymbol{K}_{i_\sigma} \int \tilde{\boldsymbol{\sigma}}_\sigma dt + \boldsymbol{K}_{d_\sigma} \dot{\tilde{\boldsymbol{\sigma}}} + \boldsymbol{K}_{p_\sigma} \tilde{\boldsymbol{\sigma}} + F_{rs,\sigma} \right) \\ + \Gamma_\sigma + \boldsymbol{c}_\sigma' + \boldsymbol{d}_\sigma' + \boldsymbol{g}_\sigma'\end{aligned} \tag{8.11}$$

where the task specific Jacobian is given in (6.23).

Part III

Simulation

Chapter 9

Setup

This chapter quickly introduces the simulator used, and mentions some of the implementation related details. The reference trajectory generator is displayed and the performance metrics used to evaluate the simulations are presented. Finally, some of the uncertainties the controllers must deal with is discussed.

9.1 Simulator description

The simulator used to implement the systems presented in part I and part II, which is based on the Eely AIAUV in section 2.3, is implemented in Matlab Simulink. Simulink provides a fixed-step solver (Bogacki-Shampine [37]) which is used to simulate the dynamic model. All simulations are performed with a step size of $h = 0.002$.

9.2 End-effector definition

For simplicity, the end-effector control objective used in the following simulations coincides with the snake head frame. The head frame is located at the center of the last link in the manipulator chain. If the end-effector control is to be used for a tool outside of the manipulators body, a linear translation is required. Any built in rotations of the tool compared to the head frame must also be taken into consideration.

9.3 Reference trajectory generation

When transitioning from a set-point reference to another, problems may arise due to instantaneous change in reference. This leads to large derivatives, and the torque assignment can become very high. To generate a reference trajectory to feed into the controller, a third-order reference model [35] described in (9.1a) and (9.1b) is used.

$$\ddot{\boldsymbol{\eta}}_d + 2\lambda\omega\dot{\boldsymbol{\eta}}_d + \delta|\dot{\boldsymbol{\eta}}_d|\dot{\boldsymbol{\eta}}_d + \omega^2\boldsymbol{\eta}_d = \omega^2\boldsymbol{\nu} \tag{9.1a}$$

$$\dot{\boldsymbol{\nu}} = \omega\left(\boldsymbol{r} - \boldsymbol{\nu}\right) \tag{9.1b}$$

Here $\boldsymbol{\eta}_d$ is the desired trajectory, $\dot{\boldsymbol{\eta}}_d$ the desired trajectory velocity and $\ddot{\boldsymbol{\eta}}_d$ the desired trajectory acceleration. The set-point reference is represented through $\boldsymbol{r}$, while ω, λ and δ are chosen parameters describing the behavior of the filter. For the simulations, the reference generator parameters has been chosen individually for the different tasks as

$$\omega_{position} = 0.4 \tag{9.2a}$$

$$\omega_{orientation} = 0.15 \tag{9.2b}$$

$$\omega_{joint} = 0.5 \tag{9.2c}$$

$$\omega_{actuation} = 0.75 \tag{9.2d}$$

$$\lambda = 1 \tag{9.2e}$$

$$\delta = 1 \tag{9.2f}$$

where different ω is used for the generation of trajectories for the Euclidean coordinates and the orientation. The generated trajectory for a step signal in end-effector position is shown in fig. 9.1.

Actuation index reference trajectory

The actuation index task described in section 6.3 and chapter 5 uses a switching scheme from [27] as described in section 5.5. The switching behavior presents a problem when generating a reference trajectory for the actuation index. The reference is set at the minimum allowed value for σ whenever mode 2 is activated, i.e. whenever the actuation index limit is violated, but the index has no reference up until this point as it is allowed to evolve freely when not violating the constraints. Ideally, the reference would follow the index as it evolves freely whenever the task is not activated. However, as the reference is fed through the filter in eqs. (9.1a) and (9.1b), this reference is delayed. The delay in the

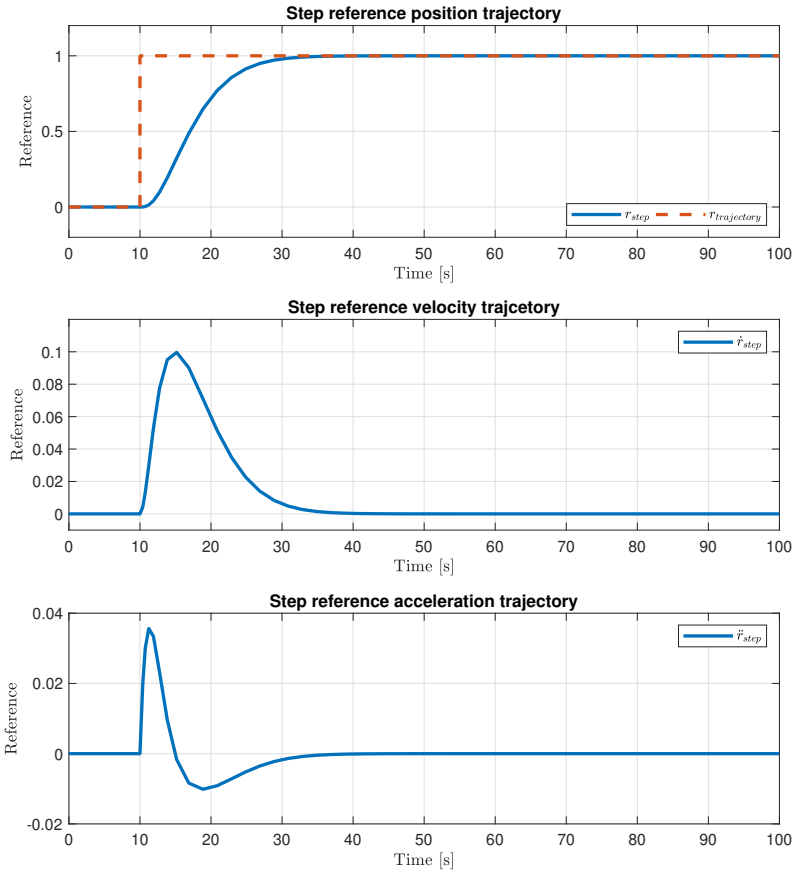

Figure 9.1: Reference trajectory generated for step reference.

filter proved manageable in the simulations, and the reference was set to follow σ whenever in mode 1.

9.4 Performance metrics

In addition to the visual representation of the results shown through plots of system variables, other performance metrics are used to see the detailed behavior. For scenario I.I and I.II, the Absolute Maximum Error (AME), during the transient and after the system has gone stationary, is used. The absolute maximum error is given by

$$AME = \max\left(|\tilde{\boldsymbol{x}}|\right) \tag{9.3}$$

where $\tilde{\boldsymbol{x}}$ is a vector of task objective behavior, for example the vector of all pitch angles θ during the simulation.

In scenario II and III there are multiple transients and settling phases, meaning the absolute maximum error is a poor choice to examine the oscillations between reference steps. Therefore, the Root-Mean-Square Error (RMSE), which is a measure of how the variable has deviated from its reference during the simulation, is used for these scenarios. The RMSE is computed as

$$RMSE = \sqrt{\frac{\sum_{t=0}^{T}\left(\tilde{\boldsymbol{x}}^2\right)}{T}} \tag{9.4}$$

where T is the total simulation time. It should be noted that RMSE is biased towards outliers in the sense that one controller may perform worse in the transient, which leads to a worse RMSE score, while performing as good or even better in the stationary phase. As such, the RMSE is a tool for examining the simulations, but it should also be viewed in relation to the plots provided and the AME performance in scenario I.I and I.II. Also, the RMSE is not used to measure the performance of the actuation index due to the reference delay discussed in section 9.3.

9.5 Simulation uncertainties

In the following simulations, uncertainties are represented in the sliding mode controllers by the lack of model knowledge and feedback. The PID-controller uses exact feedback compensation, as these are simulations without uncertainties in the matrices, and is therefore dependent on model knowledge. This is a result of the PID being very difficult to tune without compensation, and as such a comparison between the SMCs and the PID with compensation was more fruitful.

Further examination of the effects of external disturbances have been included as future work.

Chapter 10

Scenario I: Low-priority task convergence

10.1 Scenario description

The first simulation scenario examines the convergence of lower priority tasks while enforcing higher priority tasks. In this project the task combinations considered are

	Higher priority	Lower priority
Singularity avoidance	Actuation index	End-effector
Configuration control	Joint angles	End-effector

The actuation index and joint angles tasks corresponds to physical limitations in the configuration of the manipulator, and are therefore given the highest priority.

10.1.1 Scenario I.I: Singularity avoidance

The singularity avoidance simulation will enforce a constant actuation index reference while simultaneously moving the end-effector to the desired position and orientation. Contrary to the intended application of the actuation index,

where the task is only activated when required, this will allow an examination of the convergence of the end-effector in the worst case environment.

The initial conditions for the actuation index and end-effector, which is specified in Euler angles for simplicity, is

$$\sigma_0 = 0.2854 \tag{10.1a}$$

$${}^e\boldsymbol{\eta}^i_{h/i} = \begin{bmatrix} 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix}^T \tag{10.1b}$$

where σ_0 is the computed actuation index in U-shape, which the manipulator is initiated in as shown in fig. 10.1. The initial position of the end-effector and actuation index in (10.1) is a result of the initial conditions for the base and joints, which are

$${}^e\boldsymbol{\eta}^i_{b/i} = \begin{bmatrix} -0.8376 & 2.3514 & 0 & 0 & 0 & -\frac{4\pi}{3} \end{bmatrix}^T \tag{10.2a}$$

$$\boldsymbol{q}_0 = \begin{bmatrix} \frac{\pi}{6} & 0 & \frac{\pi}{6} & 0 & \frac{\pi}{6} & 0 & \frac{\pi}{6} & 0 \end{bmatrix}^T \tag{10.2b}$$

As was mentioned in section 2.3 the actuator configuration matrix may become singular whenever the manipulator is in I-shape, which is why this configuration is avoided as an initial condition when the actuation index task is applied. The desired actuation index and end-effector position and orientation is set to

$$\sigma_d = 0.10 \quad t \geq 10 \tag{10.3a}$$

$${}^e\boldsymbol{\eta}_d = \begin{bmatrix} 2.5 & -1.5 & -1 & 20 & 30 & 50 \end{bmatrix}^T \quad t \geq 10 \tag{10.3b}$$

where the end-effector references can be seen in fig. 10.2.

Control torque

The control torque within the operational space formulation for scenario I.I is given by (3.18) as

$$\boldsymbol{\tau} = \boldsymbol{\tau}_\sigma + \boldsymbol{N}_\sigma \boldsymbol{\tau}_e \tag{10.4}$$

where the torques $\boldsymbol{\tau}_\sigma$ and $\boldsymbol{\tau}_e$ are given by eqs. (8.7) and (8.11) for the PID-controller; eqs. (7.2) and (7.6) for the STA controller; and eqs. (7.8) and (7.12) for the GSTA controller.

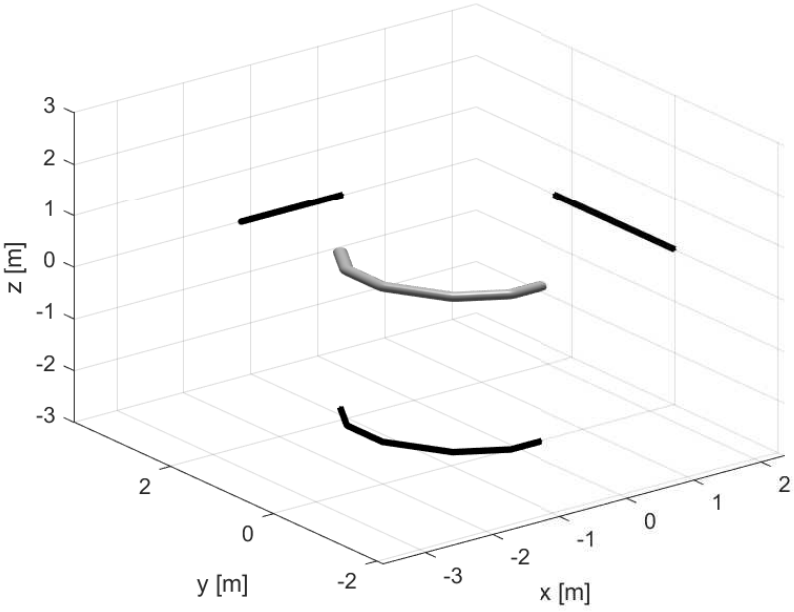

Figure 10.1: Snake robot U-shape initial position and orientation.

10.1.2 Scenario I.II: Configuration control

To look at the convergence of the end-effector when subject to a higher-priority joint angle task, a set-point reference is set for the end-effector while the joints should enforce a U-shape. The initial conditions for end-effector position and orientation is set to

$$ {}^{e}\boldsymbol{\eta}^{i}_{h/i} = \begin{bmatrix} 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix}^{T} \tag{10.5} $$

$$ \tag{10.6} $$

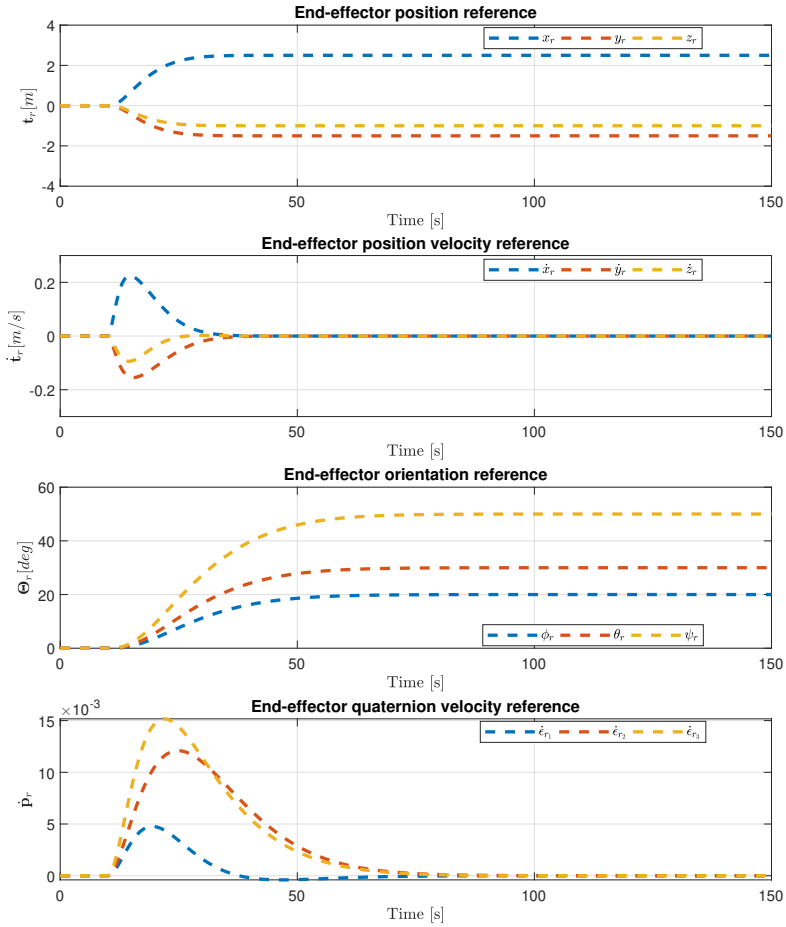

Figure 10.2: Scenario I.I & I.II: End-effector reference trajectories.

which is a result of the manipulator base and joint angle initial conditions, which are

$$\boldsymbol{q}_0 = \begin{bmatrix} 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix}^T \tag{10.7a}$$

$${}^e\boldsymbol{\eta}^i_{b/i} = \begin{bmatrix} -3 & 0 & 0 & 1 & 0 & 0 & 0 \end{bmatrix}^T \tag{10.7b}$$

The desired joint angles and end-effector position and orientation are set to

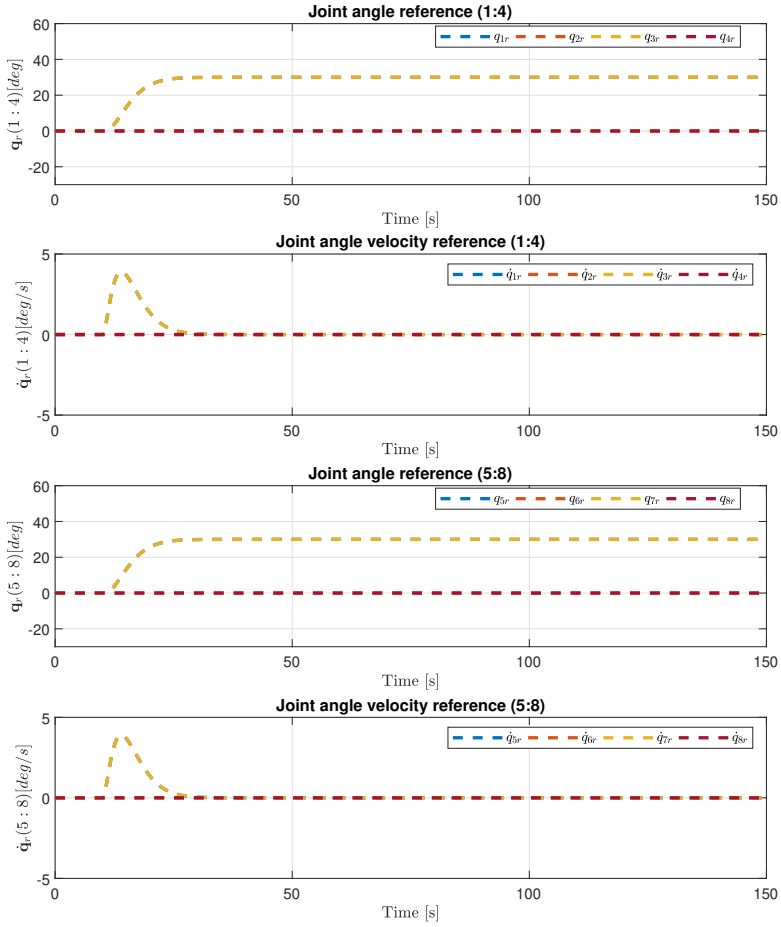

Figure 10.3: Scenario I.II: Joint angle reference trajectories.

$$\boldsymbol{q}_d = \begin{bmatrix} \frac{\pi}{6} & 0 & \frac{\pi}{6} & 0 & \frac{\pi}{6} & 0 & \frac{\pi}{6} & 0 \end{bmatrix}^T \quad t \geq 10 \tag{10.8a}$$

$${}^e\boldsymbol{\eta}_d = \begin{bmatrix} 2.5 & -1.5 & -1 & 20 & 30 & 50 \end{bmatrix}^T \quad t \geq 10 \tag{10.8b}$$

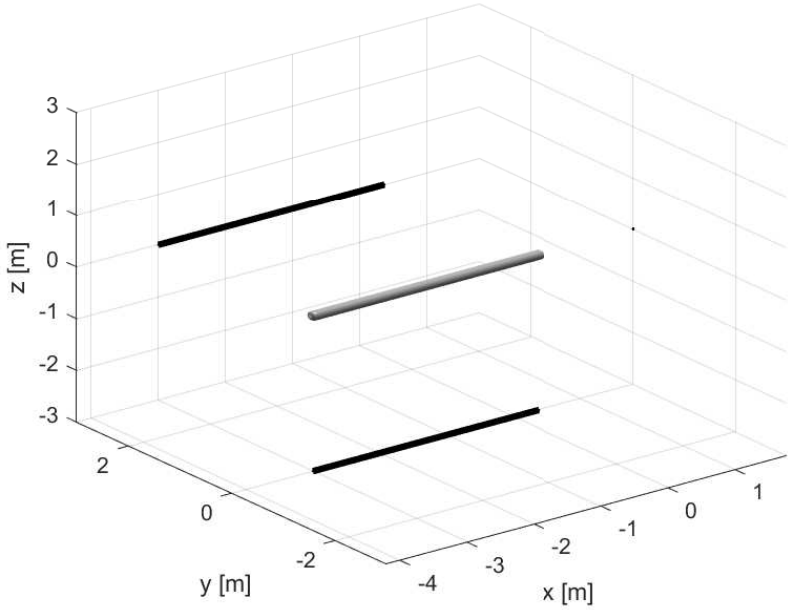

Figure 10.4: Snake robot I-shape initial position and orientation.

where the end-effector references can be seen in fig. 10.2, as it is the same for both scenario I.I and I.II, and the joint angle references in fig. 10.3.

Control torque

The control torque within the operational space formulation for scenario I.II is given by (3.18) as

$$\boldsymbol{\tau} = \boldsymbol{\tau}_q + \boldsymbol{N}_\sigma \boldsymbol{\tau}_e \tag{10.9}$$

where the torques $\boldsymbol{\tau}_q$ and $\boldsymbol{\tau}_e$ are given by eqs. (8.7) and (8.9) for the PID-controller; eqs. (7.2) and (7.4) for the STA controller; and eqs. (7.8) and (7.10) for the GSTA controller.

10.2 Scenario I.I: Singularity avoidance

In sections 10.2.1 to 10.2.3 the simulation results for the PID-controller, STA with adaptive gains and the GSTA are shown for scenario I.I, respectively. Figures 10.5, 10.7 and 10.9 contains four subplots which display the actuation index, derivative, error and the mode during simulation for the PID, STA and GSTA, respectively. Furthermore, figs. 10.6, 10.8 and 10.10 contains four subplots which display the end-effector position error, velocity error, orientation error and quaternion velocity error for the PID, STA and GSTA, respectively.

10.2.1 PID-control

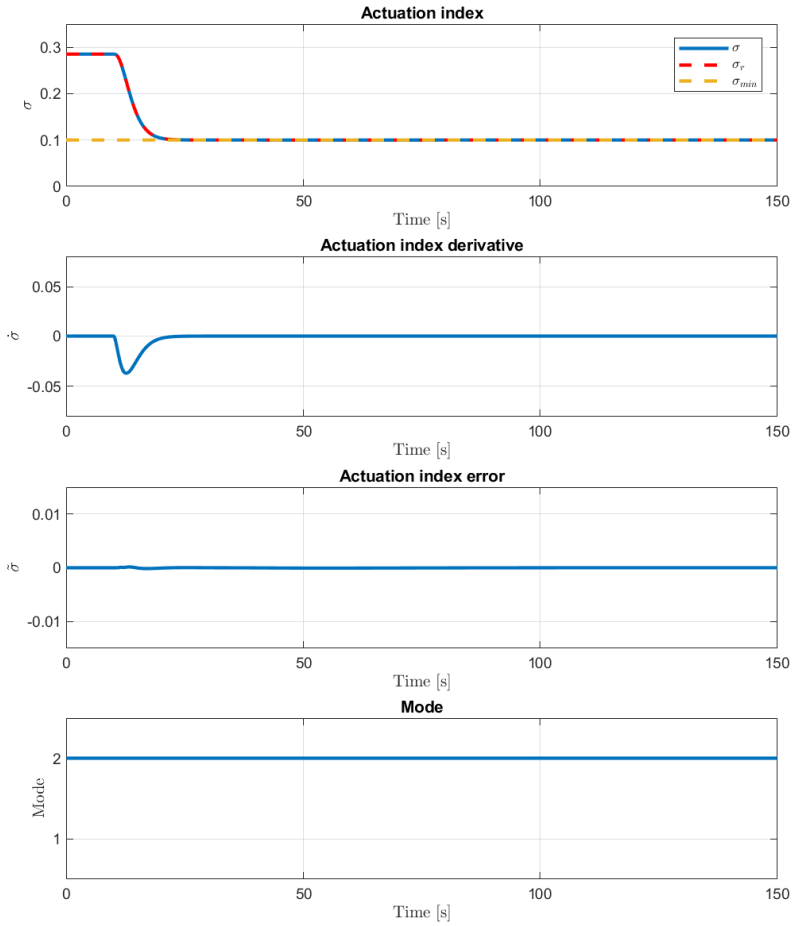

Figure 10.5: Scenario I.I: Actuation index and mode with PID controller.

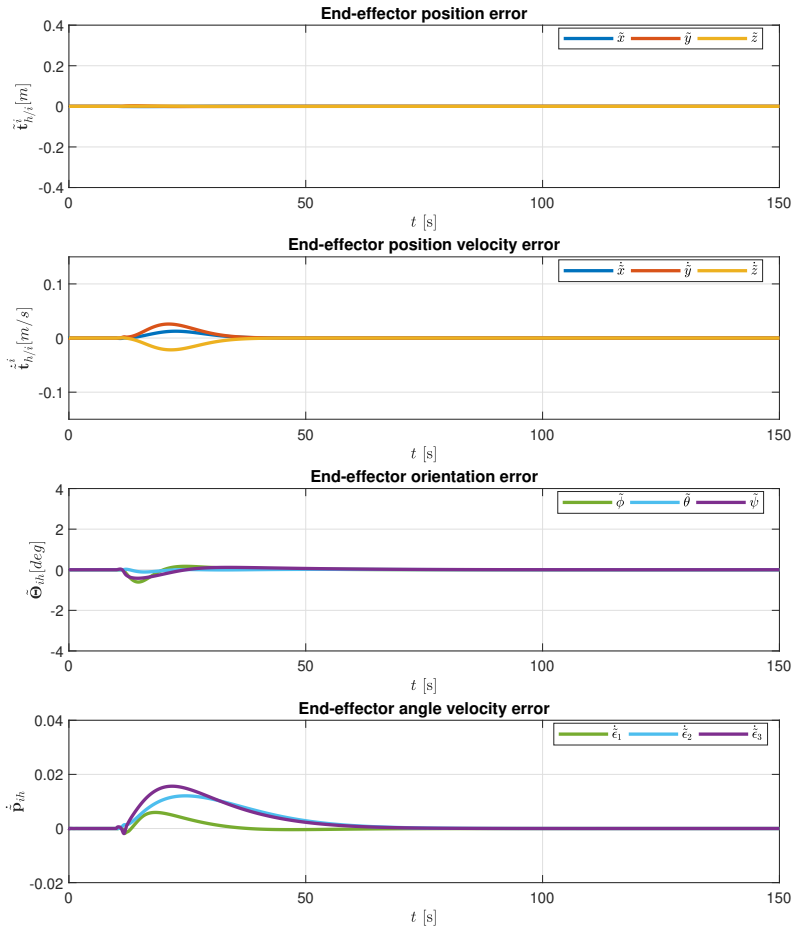

Figure 10.6: Scenario I.I: End-effector position and orientation error with PID controller.

10.2.2 Super-twisting with adaptive gains

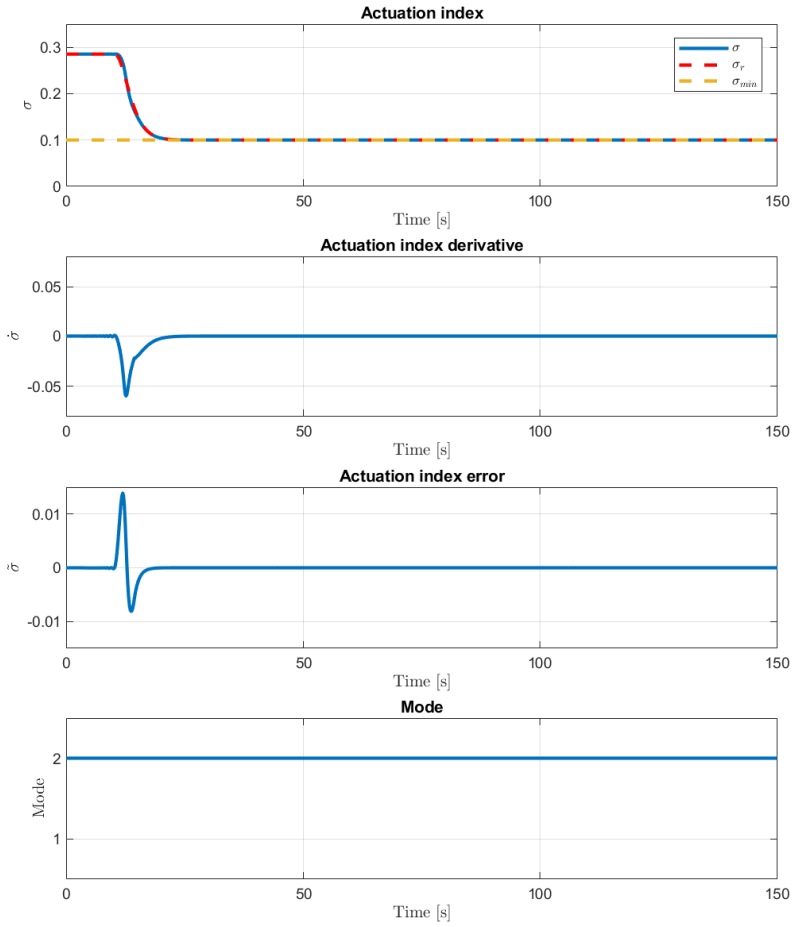

Figure 10.7: Scenario I.I: Actuation index and mode with STA controller.

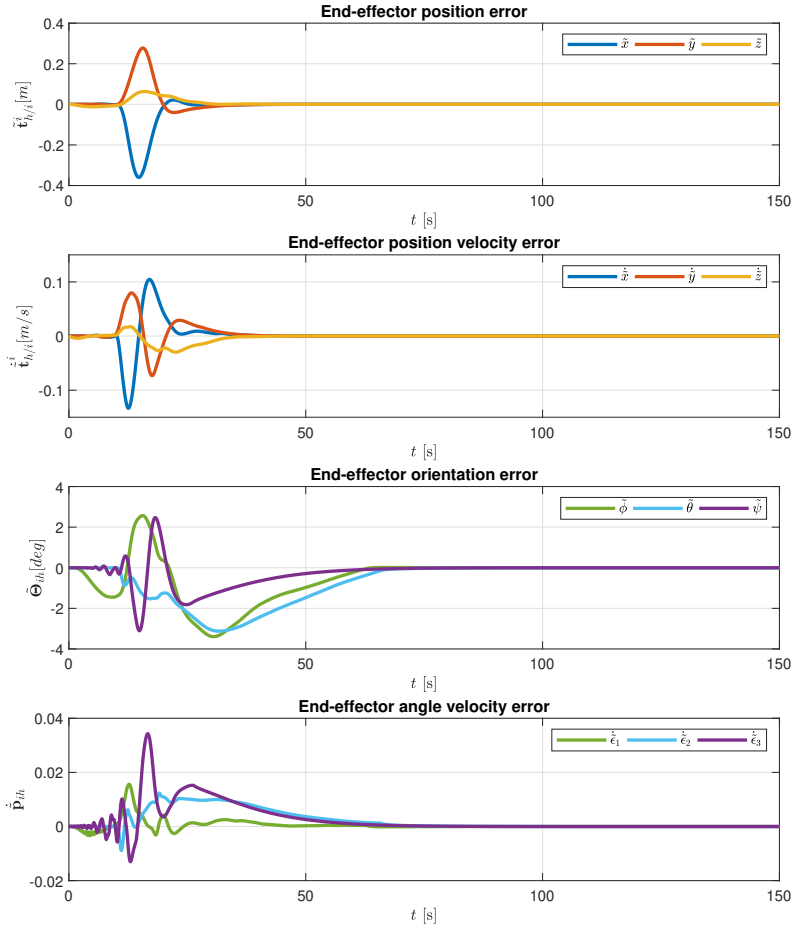

Figure 10.8: Scenario I.I: End-effector position and orientation error with STA controller.

10.2.3 Generalized super-twisting algorithm

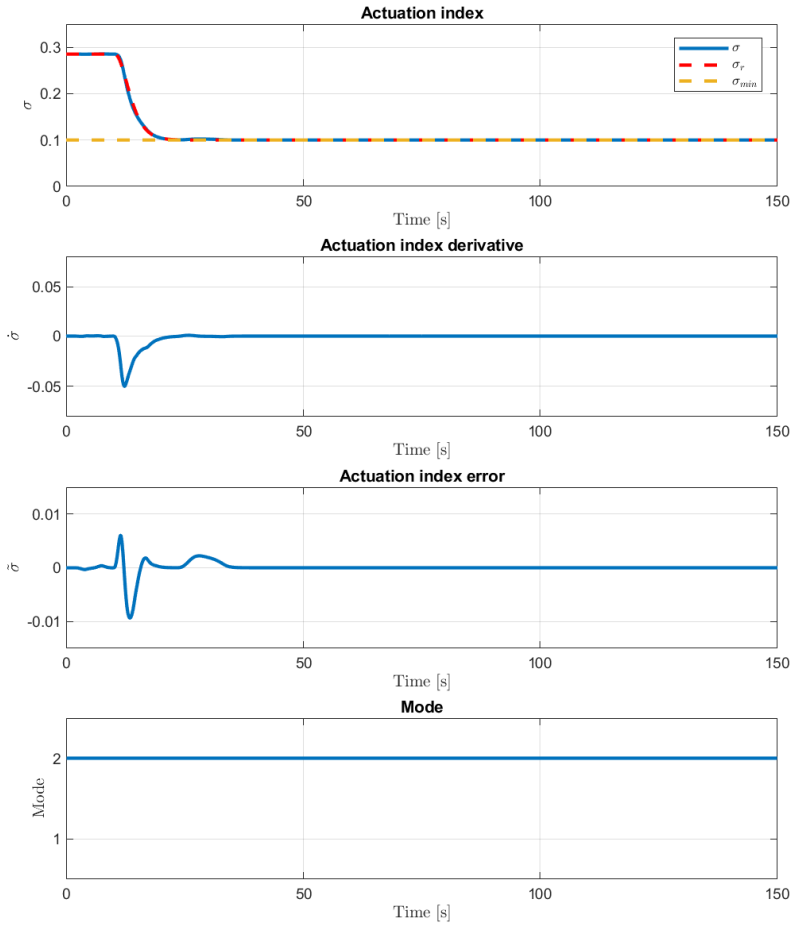

Figure 10.9: Scenario I.I: Actuation index and mode with GSTA controller.

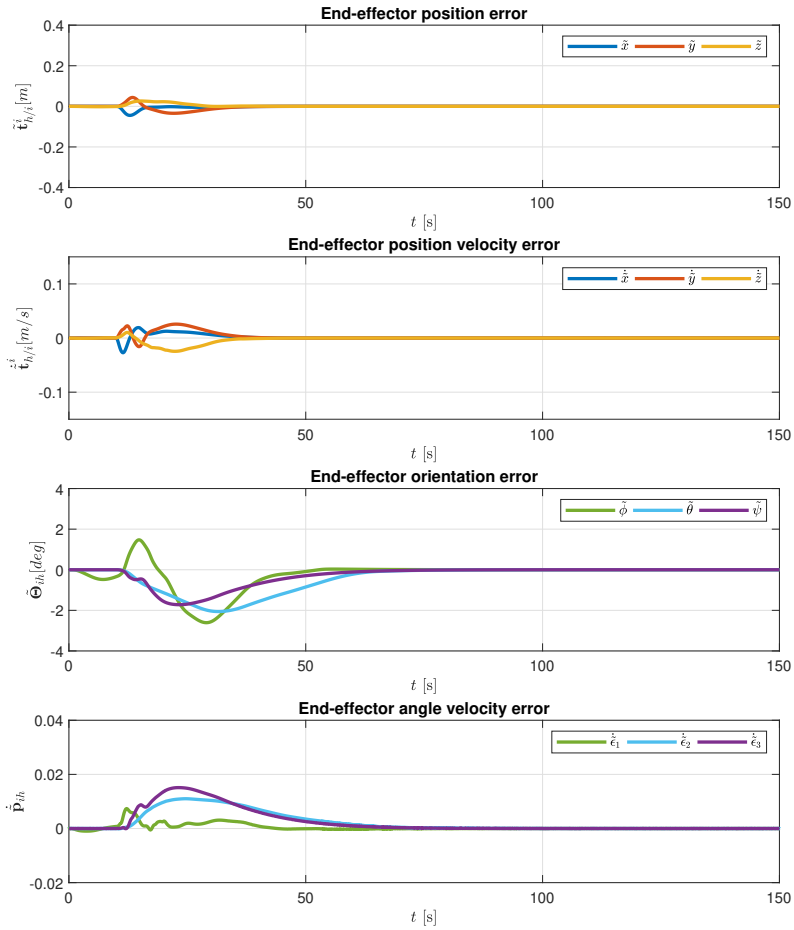

Figure 10.10: Scenario I.I: End-effector position and orientation error with GSTA controller.

10.2.4 Discussion

Since the actuation index is continually enforced as the highest priority task it does not suffer from interference by other tasks. Thus the expectation is that the actuation index will behave well at all times without any fluctuations. As seen in figs. 10.5, 10.7 and 10.9, all controllers are able to enforce the actuation index task according to the provided reference. The transient errors are relatively small, as seen in table 10.1, where the PID-controller performs slightly better than the sliding mode controllers. The STA has a slightly higher maximum transition error than GSTA, but this is not unexpected as the adaptive terms in STA must converge at the beginning. Furthermore, all controllers provide a very small error after having settled, where the sliding mode controllers perform slightly better than the PID.

The main objective of this scenario is to examine how the end-effector position and orientation converges while being projected through the null-space of the actuation index task. Looking at the end-effector errors shown in figs. 10.6, 10.8 and 10.10, and the absolute maximum errors in table 10.1, all controllers enforce the end-effector task while the actuation index is active. During the transition, the PID-controller is able to maintain the end-effector position and orientation better than the sliding mode controllers. Both the STA and GSTA controllers have a slight oscillation in figs. 10.8 and 10.10 during the transition, where the STA controller deviates from the desired path by almost 40 cm while the adaptive gains adjust to the situation. GSTA deviates by a maximum of 5 cm and the PID-controller performs on millimeter level. The same tendency can be seen for the Euler angles representing the orientation of the end-effector.

Once settled, both sliding mode controllers perform on a similar level as seen in table 10.1, while the PID-controller has a slightly lower tracking precision. However, the precision is measured in micrometers and below, so the stationary precision is high for all controllers.

To summarize, all controllers perform the task well and is able to make the lower-priority end-effector task converge with high precision. The STA controller shows some sensitivity during the transient due to the adaptive controller terms spending some time to converge.

	Errors					
	GSTA		STA		PID	
	Transition	Settled	Transition	Settled	Transition	Settled
σ	0.0094	7.6759e-9	0.0139	6.9516e-8	1.5972e-4	1.1811e-5
x [m]	0.0446	1.0211e-7	0.3600	4.2504e-7	0.0010	1.3559e-6
y [m]	0.0438	2.6160e-8	0.2775	4.2557e-7	0.0026	3.3617e-6
z [m]	0.0263	5.1361e-7	0.0631	6.2901e-8	0.0010	2.1129e-6
ϕ [deg]	2.6077	3.5727e-5	3.3946	2.0232e-5	0.6059	1.3476e-3
θ [deg]	2.0581	1.3427e-4	3.1218	4.9043e-5	0.1113	2.9823e-4
ψ [deg]	1.7215	7.2335e-5	3.1085	4.1936e-5	0.4188	5.0994e-4

Table 10.1: Absolute maximum errors for scenario I.I

10.3 Scenario I.II: Configuration control

In sections 10.3.1 to 10.3.3 the simulation results for the PID-controller, STA with adaptive gains and the GSTA are shown for scenario I.II, respectively. Figures 10.11, 10.13 and 10.15 contains four subplots which display the joint angle errors $\tilde{q}_1, \tilde{q}_2, \tilde{q}_3, \tilde{q}_4$, their derivatives, the joint angle errors $\tilde{q}_5, \tilde{q}_6, \tilde{q}_7, \tilde{q}_8$ and their derivatives for the PID, STA and GSTA, respectively. Furthermore, figs. 10.12, 10.14 and 10.16 contains four subplots which display the end-effector position error, velocity error, orientation error and quaternion velocity error for the PID, STA and GSTA, respectively.

10.3.1 PID-control

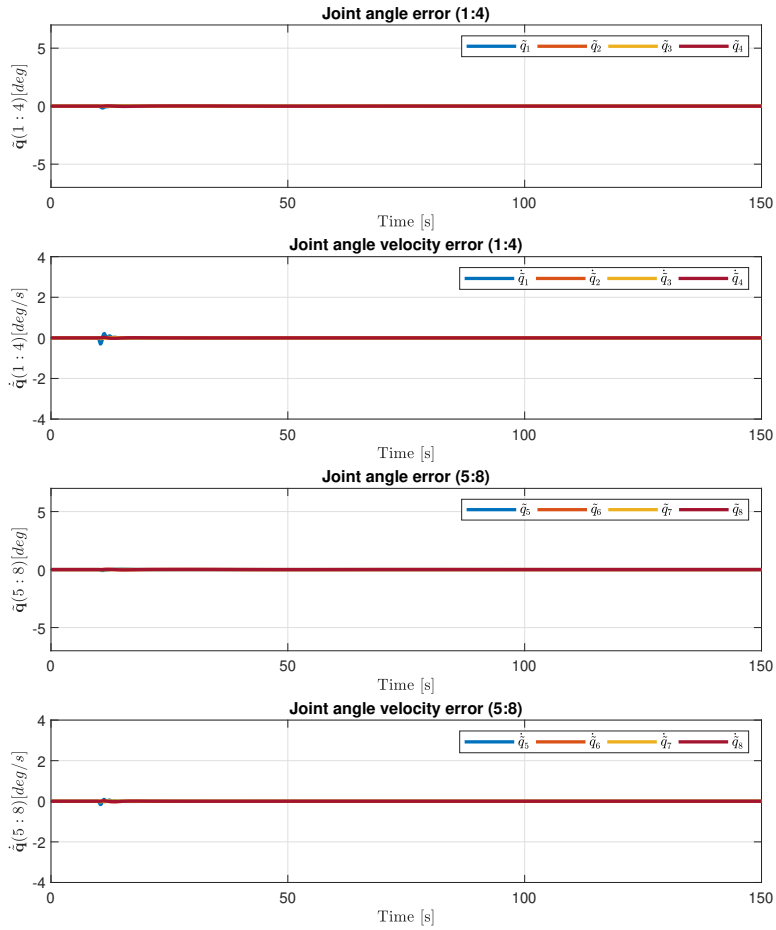

Figure 10.11: Scenario I.II: Joint angle and velocity errors with PID controller.

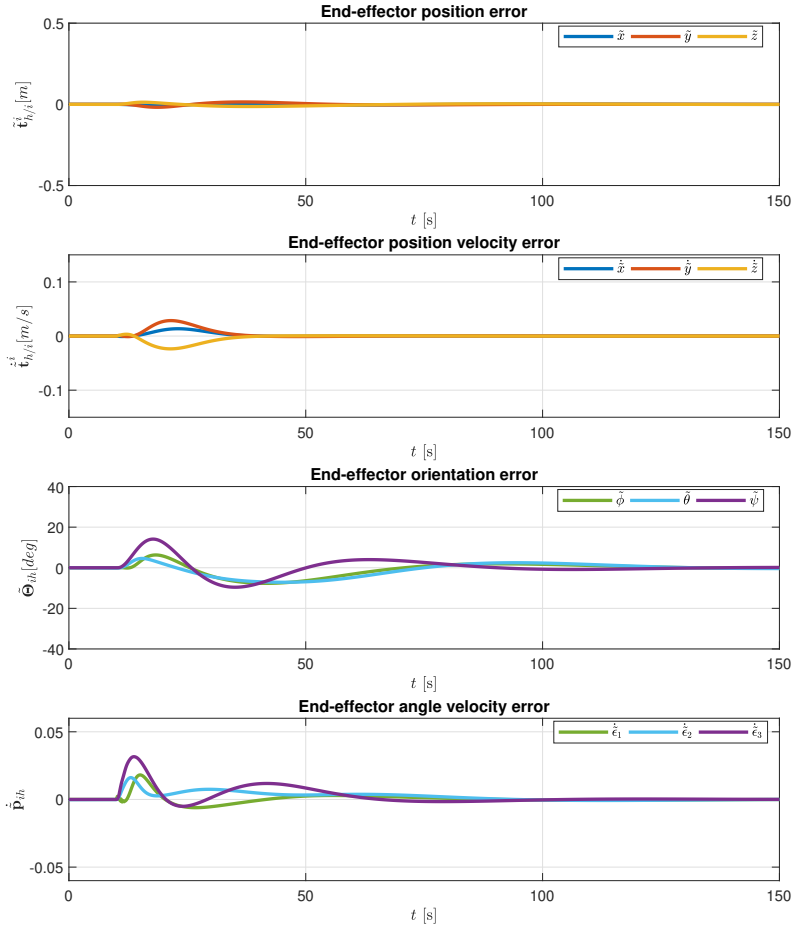

Figure 10.12: Scenario I.II: End-effector position and orientation error with PID controller.

10.3.2 Super-twisting with adaptive gains

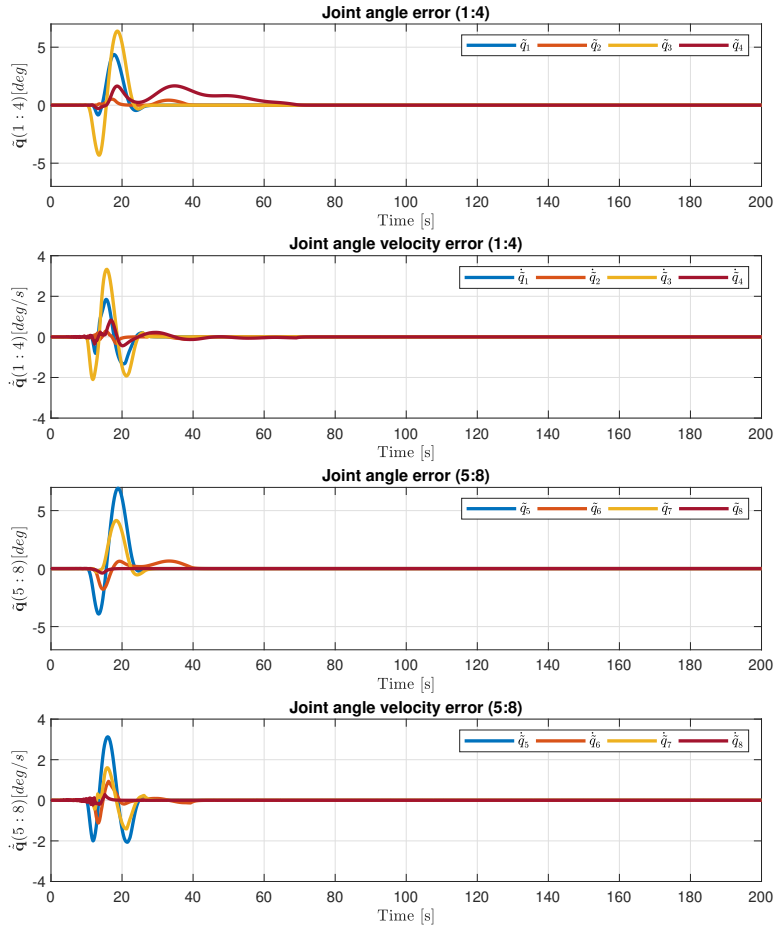

Figure 10.13: Scenario I.II: Joint angle and velocity errors with STA controller.

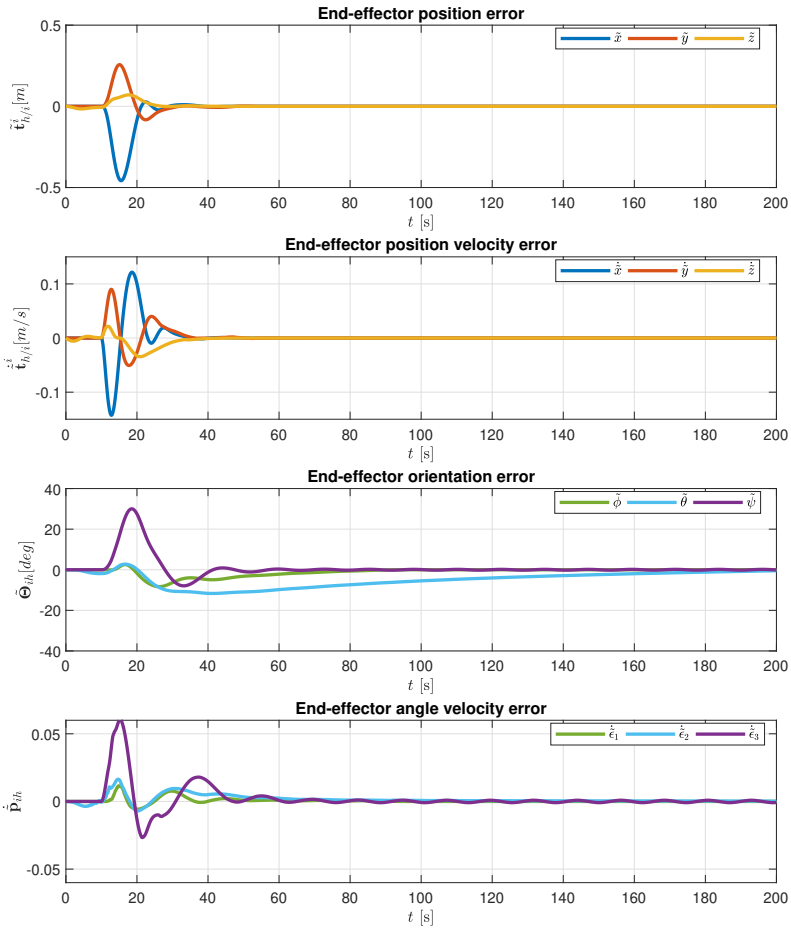

Figure 10.14: Scenario I.II: End-effector position and orientation error with STA controller.

10.3.3 Generalized super-twisting algorithm

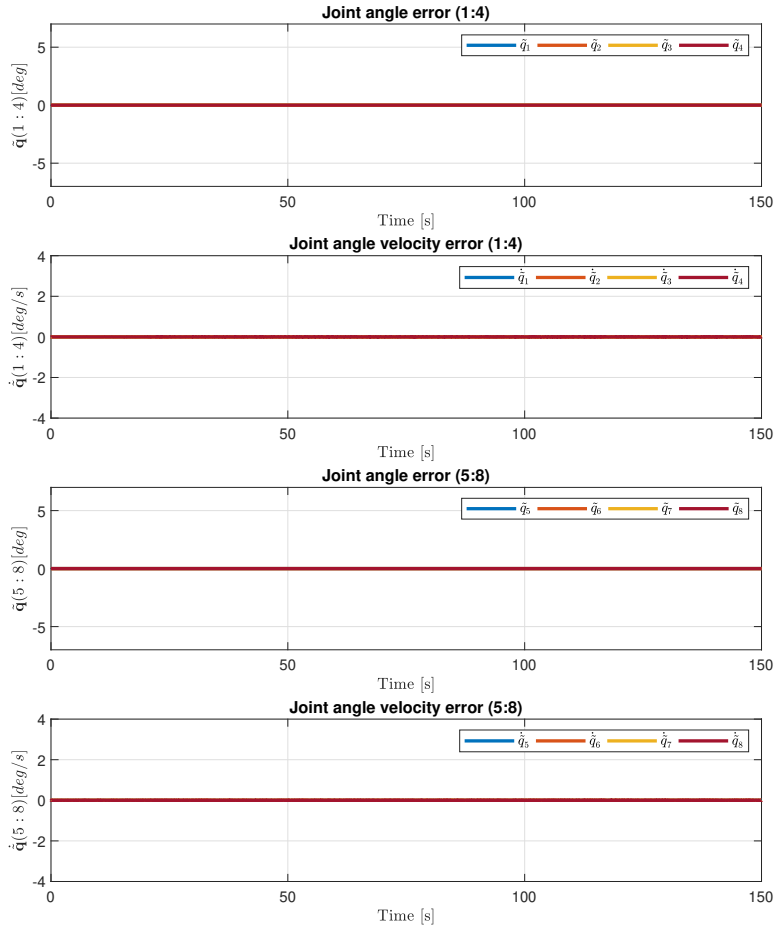

Figure 10.15: Scenario I.II: Joint angle and velocity errors with GSTA controller.

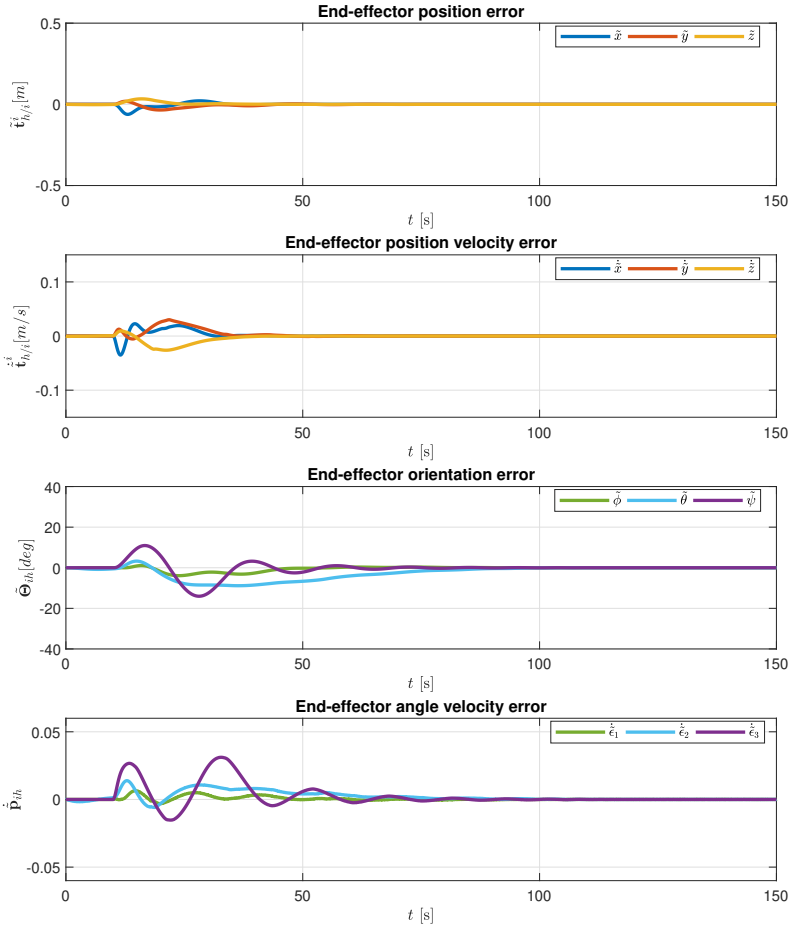

Figure 10.16: Scenario I.II: End-effector position and orientation error with GSTA controller.

10.3.4 Discussion

This scenario enforces the joint angles, being a physical limitation, as the higher priority task. From figs. 10.11, 10.13 and 10.15 and table 10.2 both GSTA and PID maintain the joint angles well during the transient. From table 10.2, keeping in mind that the angle error is measured in degrees, GSTA and PID has a very low transient error, where GSTA outperforms the PID-controller. As was mentioned in section 10.2.4, the STA controller has to adjust its adaptive gains during a transient and this shines through in this scenario, where the joint angle tracking during the transient is outperformed by the other controllers. However, once the system settles the stationary error is on par with both the PID and GSTA controllers.

The behavior of the end-effector error can be seen in figs. 10.12, 10.14 and 10.16 and table 10.2. It is much the same as observed in section 10.2, where the STA has the worst performance in the transition phase, while the GSTA is able to keep up with, and even outperform in the stationary phase, the PID-controller within this task priority layout. In addition, after the system has settled, the GSTA controller performs the best in all categories. The STA controller takes a long time to converge in orientation, where the simulation had to be run an extra 50 seconds to illustrate the convergence of $\tilde{\boldsymbol{\theta}}$. The STA controller is stronger when it comes to position error than the PID, although it also struggles with the orientation, especially the pitch angle θ. It should be noted that the PID is still accurate to a tenth of a millimeter, which would work in most physical applications.

	Errors					
	GSTA		STA		PID	
	Transition	Settled	Transition	Settled	Transition	Settled
x [m]	0.0618	3.3520e-8	0.4579	5.9860e-7	0.0073	8.2085e-4
y [m]	0.0348	2.5043e-8	0.2570	1.5866e-6	0.0180	1.9700e-4
z [m]	0.0337	2.0343e-8	0.0704	1.8874e-7	0.0134	8.6080e-4
ϕ [deg]	3.9024	3.8022e-5	8.3311	4.8257e-3	7.6797	5.3094e-1
θ [deg]	8.8466	1.2359e-4	11.677	2.8952e-0	7.1503	7.6475e-1
ψ [deg]	14.029	7.7387e-5	30.018	1.9390e-1	14.111	5.2219e-1
q_1 [deg]	4.4232e-5	9.1773e-6	4.3644	1.9356e-6	1.5564e-1	1.0062e-4
q_2 [deg]	1.1828e-4	2.8059e-5	0.5189	9.4819e-6	1.8464e-2	6.3600e-4
q_3 [deg]	5.6187e-5	7.1659e-6	6.3868	6.4003e-6	2.4874e-2	1.7634e-5
q_4 [deg]	1.2549e-4	1.3728e-5	1.6655	1.1577e-5	2.1954e-2	7.8014e-4
q_5 [deg]	4.8339e-5	8.1258e-6	6.9292	8.5401e-6	8.1554e-2	2.5631e-5
q_6 [deg]	1.1092e-4	1.6207e-5	1.7819	1.6963e-5	2.1650e-2	6.7972e-4
q_7 [deg]	4.9469e-5	1.0558e-5	4.1466	1.4608e-6	2.4079e-2	9.1118e-6
q_8 [deg]	5.4739e-5	1.3672e-5	0.3836	2.9918e-6	2.0155e-2	5.8660e-4

Table 10.2: Absolute maximum errors for scenario I.II

Chapter 11

Scenario II: Position and Configuration

11.1 Scenario description

The goal of scenario II is to examine how the controllers compare when controlling the joint angle as a primary task ahead of the end-effector position and orientation. This scenario extends on the simulations in section 10.3 and incorporates a more complex movement, where the manipulator starts with a transport task where the end-effector position is changed while maintaining the orientation and I-shape. Eventually the manipulator changes from I-shape to U-shape, where the dexterity is higher, before a final shift for the head angle, end-effector position and a large change in orientation is performed. This final movement could correspond to navigating in a tight area in a subsea environment.

	Higher priority	Lower priority
Position and configuration	Joint angles	End-effector

Furthermore, the initial conditions are set to the I-shape as depicted in fig. 10.1,

where

$$ {}^{e}\boldsymbol{\eta}^{i}_{h/i} = \begin{bmatrix} 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix}^{T} \tag{11.1a} $$

$$ \boldsymbol{q}_0 = \begin{bmatrix} 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix}^{T} \tag{11.1b} $$

$$ {}^{e}\boldsymbol{\eta}^{i}_{b/i} = \begin{bmatrix} -3 & 0 & 0 & 0 & 0 & 0 \end{bmatrix}^{T} \tag{11.1c} $$

The desired joint angles, where the time of change in set point is given by t, are set to

$$ \boldsymbol{q}_d = \begin{bmatrix} \frac{\pi}{6} & 0 & \frac{\pi}{6} & 0 & \frac{\pi}{6} & 0 & \frac{\pi}{6} & 0 \end{bmatrix}^{T} \quad 60 \leq t < 120 \tag{11.2a} $$

$$ \boldsymbol{q}_d = \begin{bmatrix} \frac{\pi}{6} & 0 & \frac{\pi}{6} & 0 & \frac{\pi}{6} & 0 & 0 & 0 \end{bmatrix}^{T} \quad t \geq 120 \tag{11.2b} $$

and can be seen in fig. 11.1. The desired end-effector position is given by

$$ {}^{e}\boldsymbol{\eta}_d = \begin{bmatrix} 3 & 4 & -2 & 0 & 0 & 0 \end{bmatrix}^{T} \quad t \geq 10 \tag{11.3a} $$

$$ {}^{e}\boldsymbol{\eta}_d = \begin{bmatrix} 5 & 4 & -1 & 70 & 10 & 70 \end{bmatrix}^{T} \quad t \geq 120 \tag{11.3b} $$

and can be seen in fig. 11.2.

Control torque

The control torque within the operational space formulation for scenario II is given by (3.18) as

$$ \boldsymbol{\tau} = \boldsymbol{\tau}_q + \boldsymbol{N}_\sigma \boldsymbol{\tau}_e \tag{11.4} $$

where the torques $\boldsymbol{\tau}_q$ and $\boldsymbol{\tau}_e$ are given by eqs. (8.7) and (8.9) for the PID-controller; eqs. (7.2) and (7.4) for the STA controller; and eqs. (7.8) and (7.10) for the GSTA controller.

In sections 11.2 to 11.4 the simulation results for the PID-controller, STA with adaptive gains and the GSTA are shown for scenario II, respectively. Figures 11.3, 11.5 and 11.7 contains four subplots which display the joint angle errors $\tilde{q}_1, \tilde{q}_2, \tilde{q}_3, \tilde{q}_4$, their derivatives, the joint angle errors $\tilde{q}_5, \tilde{q}_6, \tilde{q}_7, \tilde{q}_8$ and their derivatives for the PID, STA and GSTA, respectively. Furthermore, figs. 11.4, 11.6 and 11.8 contains four subplots which display the end-effector position error, velocity error, orientation error and quaternion velocity error for the PID, STA and GSTA, respectively.

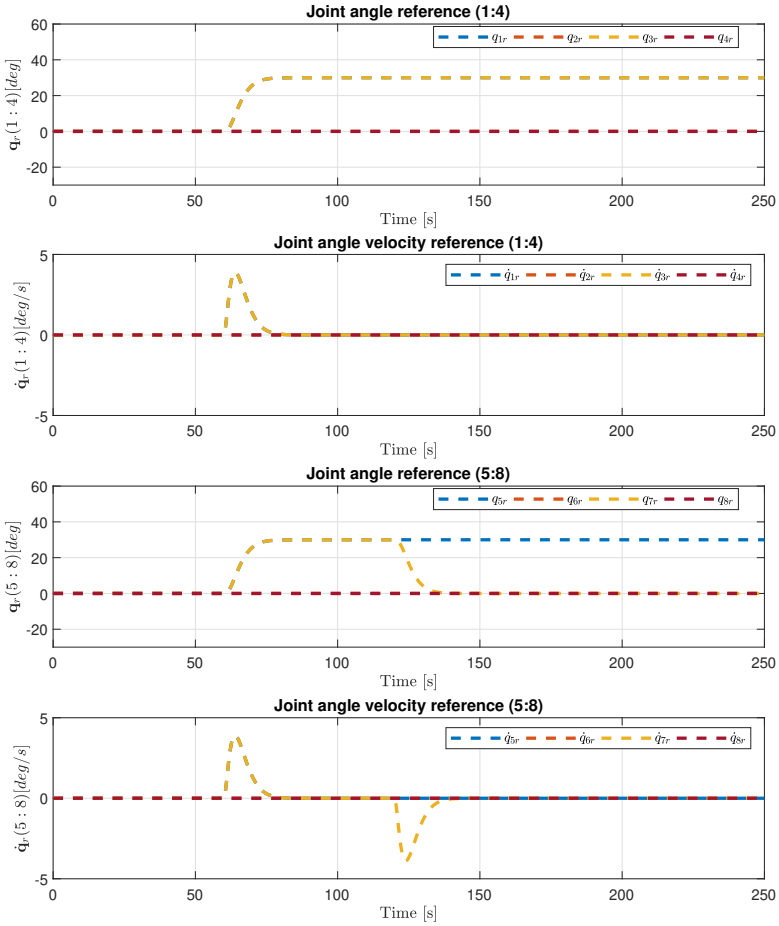

Figure 11.1: Scenario II: Joint angle reference trajectories.

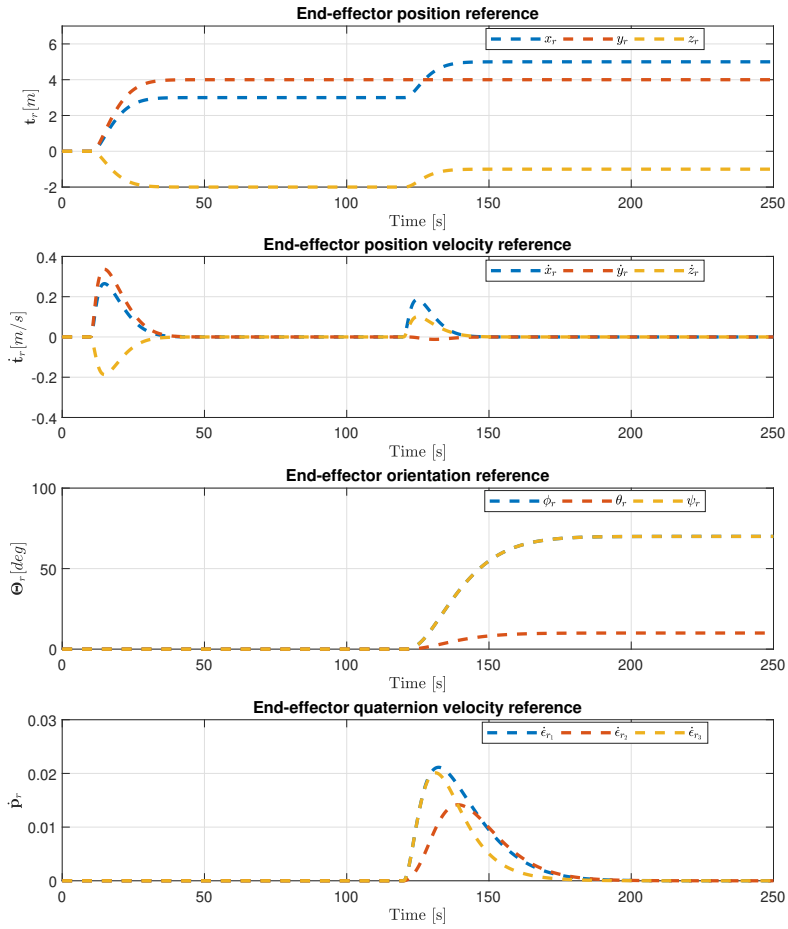

Figure 11.2: Scenario II: End-effector reference trajectories.

11.2 PID-control

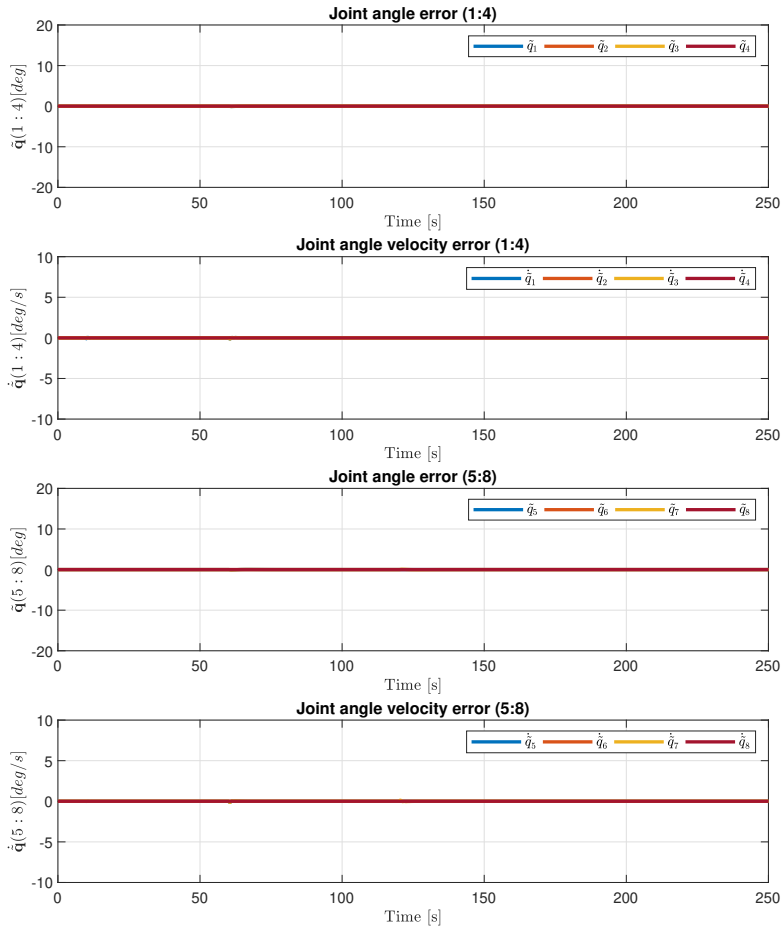

Figure 11.3: Scenario II: Joint angle and velocity errors with PID controller.

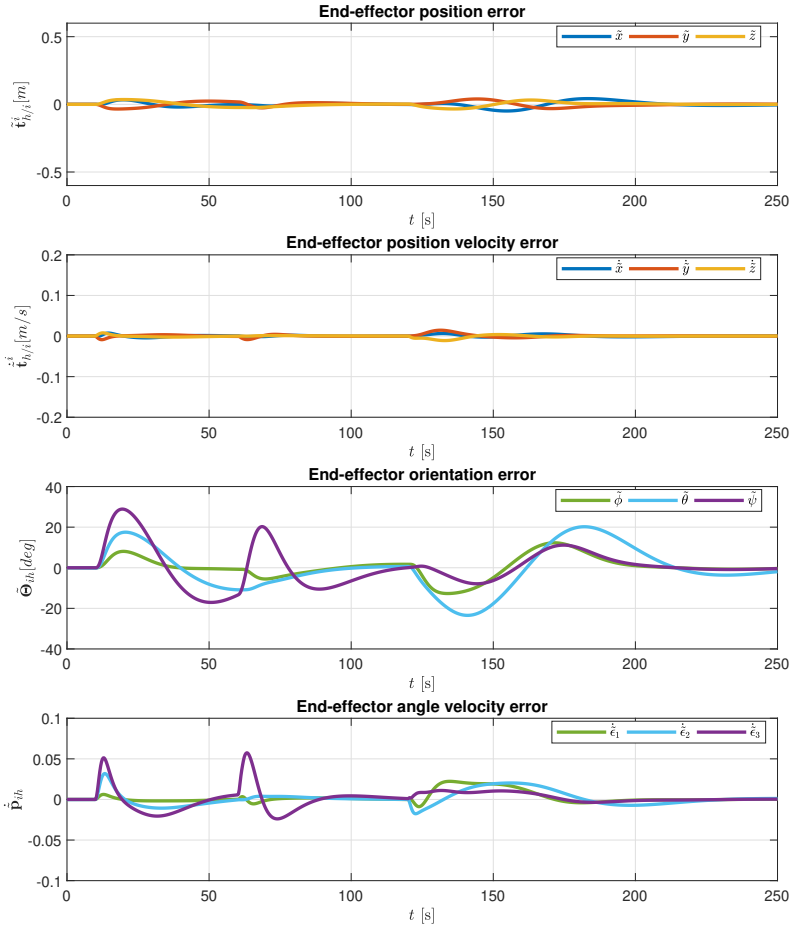

Figure 11.4: Scenario II: End-effector position and orientation error with PID controller.

11.3 Super-twisting with adaptive gains

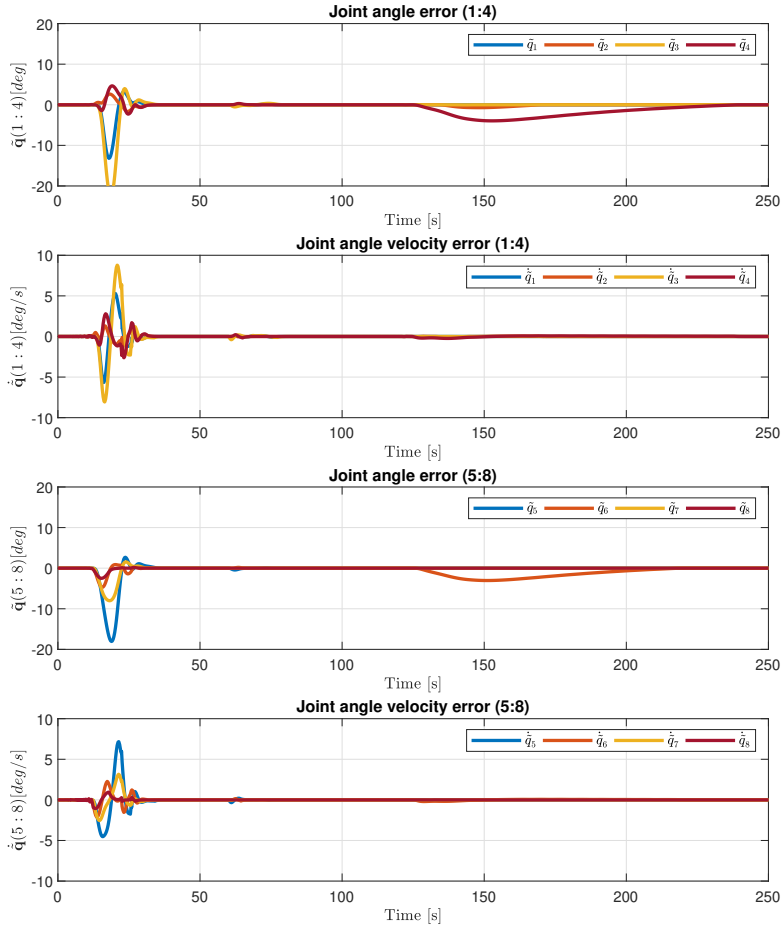

Figure 11.5: Scenario II: Joint angle and velocity errors with STA controller.

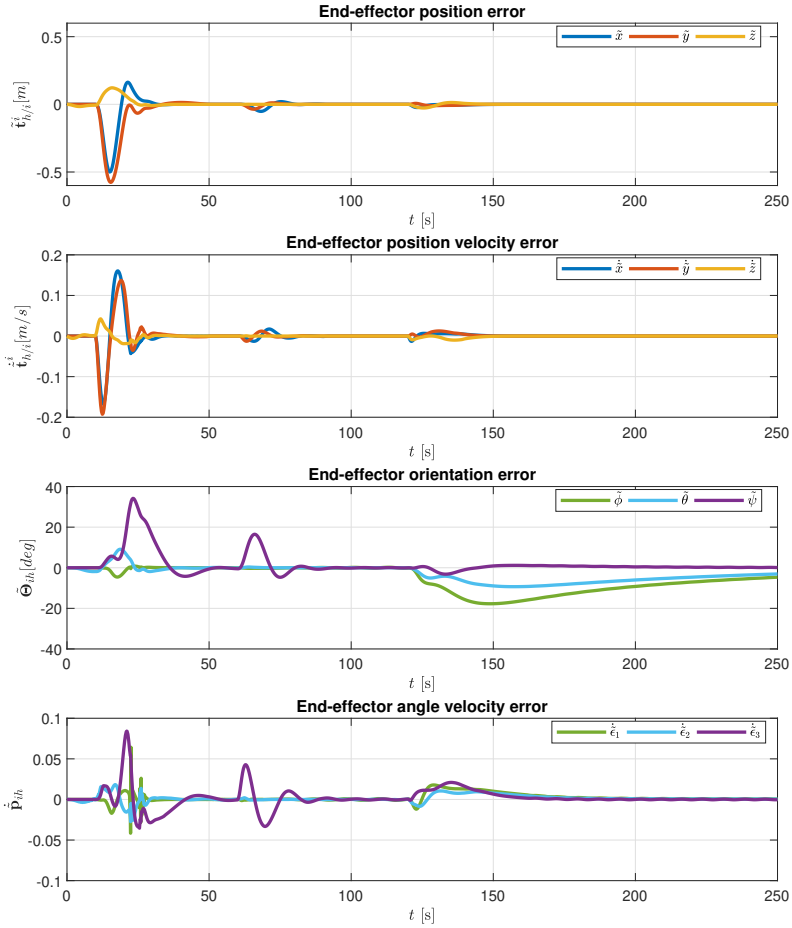

Figure 11.6: Scenario II: End-effector position and orientation error with STA controller.

11.4 Generalized super-twisting algorithm

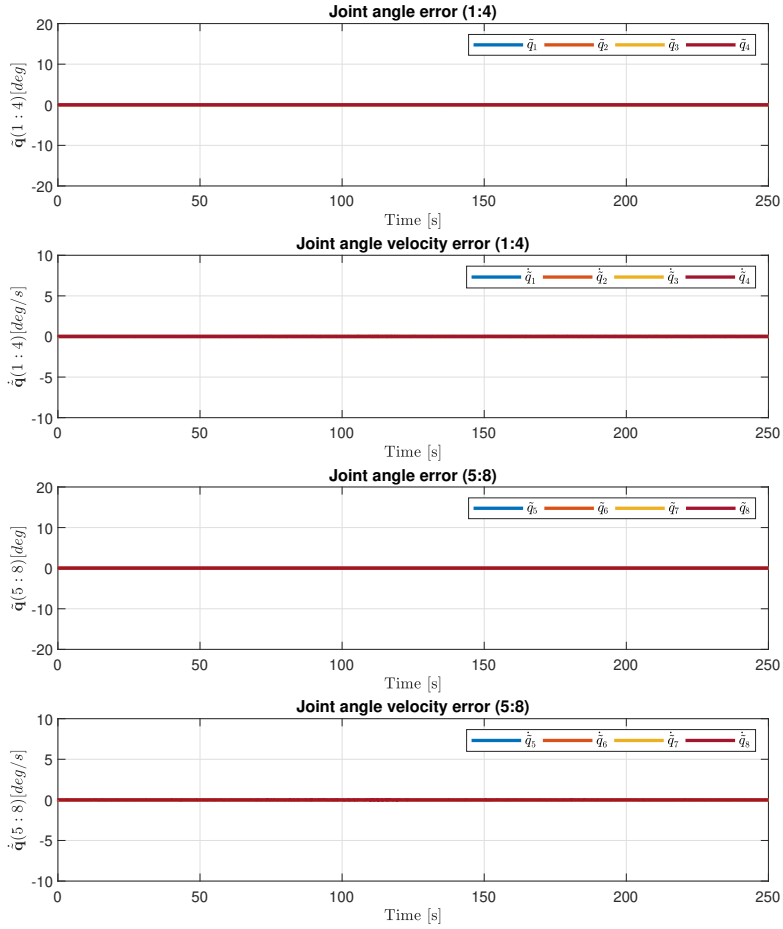

Figure 11.7: Scenario II: Joint angle and velocity errors with GSTA controller.

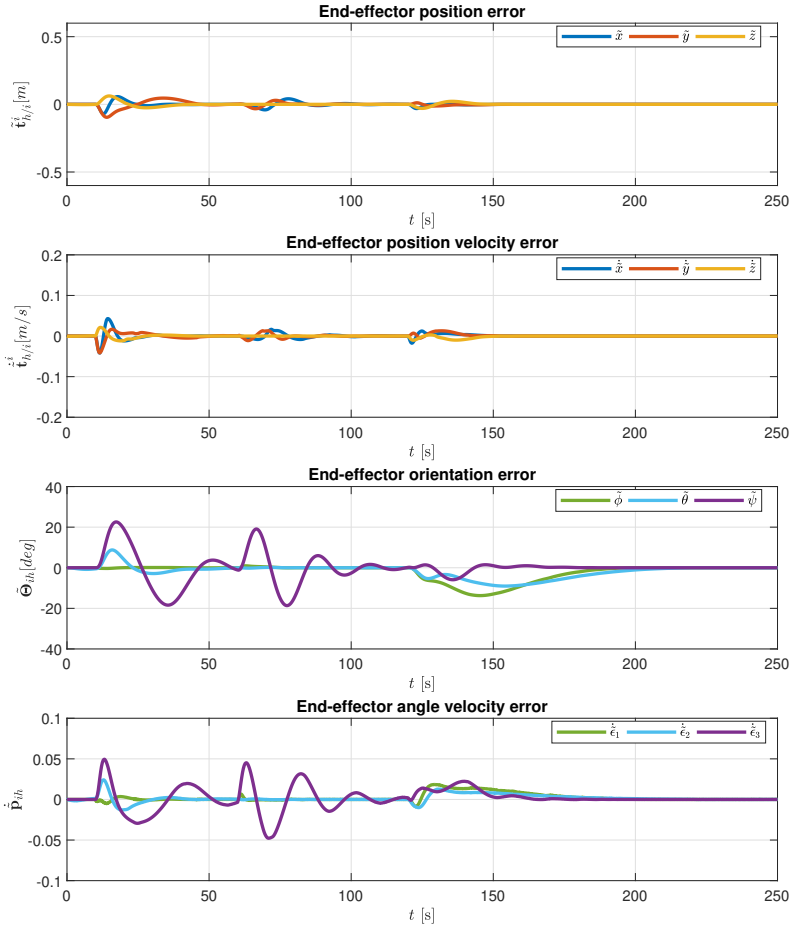

Figure 11.8: Scenario II: End-effector position and orientation error with GSTA controller.

11.5 Discussion

Figure 11.9 shows the traversed end-effector trajectories for each controller in scenario II. There are some deviations in the traversed path for the different controllers, most obvious for the STA controller, and in the next paragraphs the origin of these deviations, among other observations from the results in sections 11.2 to 11.4, will be discussed.

In figs. 11.3, 11.5 and 11.7 the behavior of the high priority joint angle task can be seen. Both the PID and GSTA controllers perform much better than the STA in joint angle control. This was also the case in scenario I.II and the RMSE values given in table 11.1 also support this. Looking at fig. 11.5 it is also clear that the STA controller struggles to follow the trajectory of the joint angles perfectly, as there is a large deviation at $t = 10\,\mathrm{s}$ at the first end-effector reference change, while both the PID and GSTA controllers are able to maintain the configuration under influence of the lower-priority end-effector task. However, STA only experiences this problem at the first reference shift, where the gains need to converge, while performing much better in subsequent reference changes. From table 11.1 it is also clear that the joint tracking for the PID and GSTA controllers perform very well, which is also evident in the figs. 11.3 and 11.7, respectively.

The end-effector position reference is well imposed for all controllers, however, they clearly struggle to maintain the orientation trajectory while enforcing the joint angles, which is evident from figs. 11.4, 11.6 and 11.8. There are rather large oscillations in roll θ and yaw ψ after the first position change for all controllers, with the STA controller hitting a larger amplitude than the others. Looking at the final reference change at $t = 120\,\mathrm{s}$ the GSTA controller outperforms the other controllers with a smaller error as shown in fig. 11.8, although not quite following the reference in the transient. However, the STA controller has upped its performance, but the gains still take awhile to converge as the Euler angles converge slowly. This is likely a result of the STA tuning, where a higher convergence rate for the adaptive gains could be achieved through more aggressive tuning. The PID controller has oscillations during the transient, and a rather large orientation deviation as seen in fig. 11.4.

Some of the oscillation issues from $t = 10\,\mathrm{s}$ can be attributed to the manipulator being in I-shape. As was mentioned in section 2.3, the manipulator is unactuated in roll for this configuration, which was also observed in the experimental study [38]. In addition, maintaining the configuration in I-shape is not ideal while

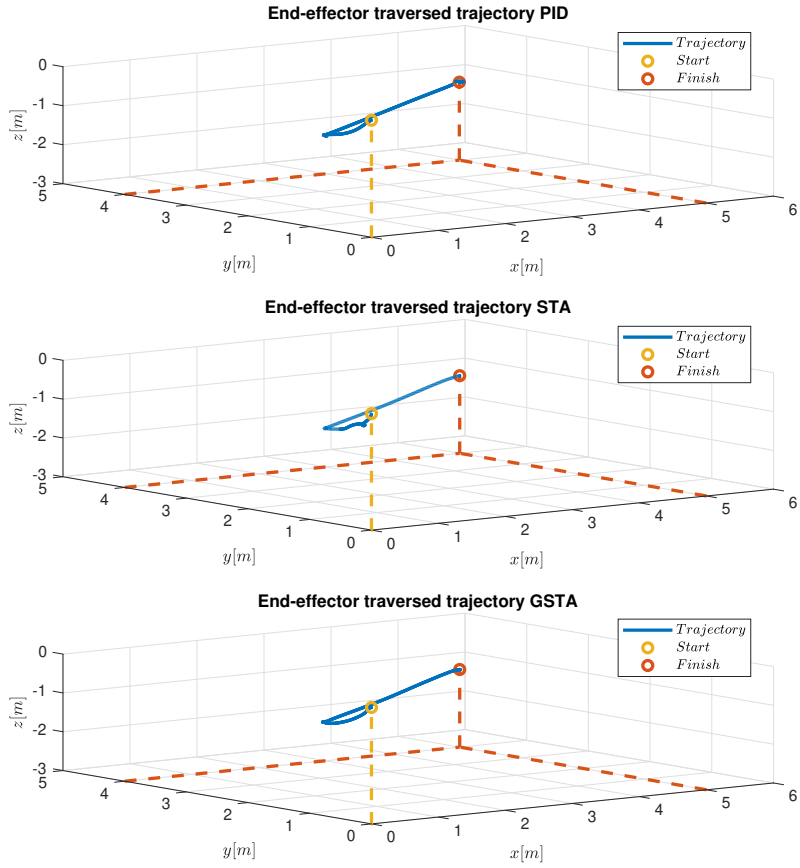

Figure 11.9: Scenario II: AIAUV end-effector trajectories.

moving the end-effector with a constant yaw angle, and this could be further reason for the oscillations in yaw angle where the tasks have to compete.

	RMSE		
	GSTA	STA	PID
x [m]	1.2868e-2	6.2906e-2	1.8669e-2
y [m]	1.7095e-2	7.9403e-2	1.6921e-2
z [m]	1.2221e-2	2.0058e-2	1.6317e-2
ϕ [deg]	4.3895	8.1206	5.1711
θ [deg]	3.5576	4.7965	10.511
ψ [deg]	6.6977	5.9549	9.4778
q_1 [deg]	8.6349e-6	1.4701	6.6338e-3
q_2 [deg]	3.6710e-5	3.8536e-1	2.2817e-7
q_3 [deg]	9.7078e-6	2.6094	6.6111e-3
q_4 [deg]	4.8736e-5	1.6716	1.0832e-7
q_5 [deg]	1.1512e-5	2.2658	6.6052e-3
q_6 [deg]	4.8094e-5	1.2417	9.5385e-8
q_7 [deg]	1.4163e-5	1.1231	9.3394e-3
q_8 [deg]	2.4259e-5	2.8454e-1	2.0156e-7

Table 11.1: Root-Mean-Square-Error for scenario II

Chapter 12

Scenario III: Online Singularity Avoidance

12.1 Scenario description

In scenario III, the actuation index method developed in the specialization project is examined. The project concluded that the method shows promise, but further testing with more advanced controllers was required. Here, the validity of the method is further examined while also comparing the robustness of the controllers tested in this thesis. The task priority is

	Higher priority	Lower priority
Online singularity avoidance	Actuation index	End-effector

where the actuation index is implemented as a set-based task as described in section 5.5. The initial conditions are set to the U-shape depicted in fig. 10.1.

$$ {}^{e}\boldsymbol{\eta}^{i}_{h/i} = \begin{bmatrix} 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix}^T \tag{12.1a} $$

$$ \boldsymbol{q}_0 = \begin{bmatrix} \frac{\pi}{6} & 0 & \frac{\pi}{6} & 0 & \frac{\pi}{6} & 0 & \frac{\pi}{6} & 0 \end{bmatrix}^T \tag{12.1b} $$

$$ {}^{e}\boldsymbol{\eta}^{i}_{b/i} = \begin{bmatrix} -0.8376 & 2.3514 & 0 & 0 & 0 & -\frac{4\pi}{3} \end{bmatrix}^T \tag{12.1c} $$

The end-effector desired position and orientation is

$$
\begin{aligned}
{}^{e}\boldsymbol{\eta}_d &= \begin{bmatrix} 1 & 2 & 2 & 0 & 0 & 0 \end{bmatrix}^T \quad t \geq 10 && \text{(12.2a)} \\
{}^{e}\boldsymbol{\eta}_d &= \begin{bmatrix} 3 & 3 & 3 & 20 & 10 & 40 \end{bmatrix}^T \quad 60 \leq t \leq 150 && \text{(12.2b)} \\
{}^{e}\boldsymbol{\eta}_d &= \begin{bmatrix} 5 & 5 & 5 & 40 & 30 & 50 \end{bmatrix}^T \quad t \geq 150 && \text{(12.2c)}
\end{aligned}
$$

and can be seen in fig. 12.1. When the actuation index is enforced as a set-based task, special considerations have to be made for the desired value. As with the references for the end-effector and joints, the actuation index reference is fed through a reference trajectory generator as described in section 9.3. Whenever the set-based task is activated, the system is in mode 2, the reference is then set equal to

$$\sigma_d = \sigma_{min} = 0.10 \tag{12.3}$$

and whenever the set-based task is inactive, the system is in mode 1, the reference is set equal to

$$\sigma_d = \sigma \tag{12.4}$$

As the mode is decided by the extended tangent cone function $T_{\mathbb{R},D}$ in (5.30), the actuation index reference can be written as

$$\sigma_d = \begin{cases} \sigma_{min} & \text{if } \dot{\sigma} \notin T_{\mathbb{R},D} \\ \sigma & \text{otherwise} \end{cases} \tag{12.5}$$

Since $\sigma_d = \sigma$ whenever $\dot{\sigma} \in T_{\mathbb{R},D}$, the reference will be fed through the reference trajectory generator when enforcing the actuation index task. This reduces the system velocities when σ reaches σ_{min}, such that the controller is more likely to stop the system before going past the limit.

Remark: Since the reference is set to $\sigma_d = \sigma$ when in mode 1 and then fed through the reference generator, the plots in sections 12.2 to 12.4 have a delayed actuation index reference signal whenever in mode 1. Since the reference is not actually used in any manner during mode 1 this is of no major consequence other than a slight deviation in reference whenever entering mode 2.

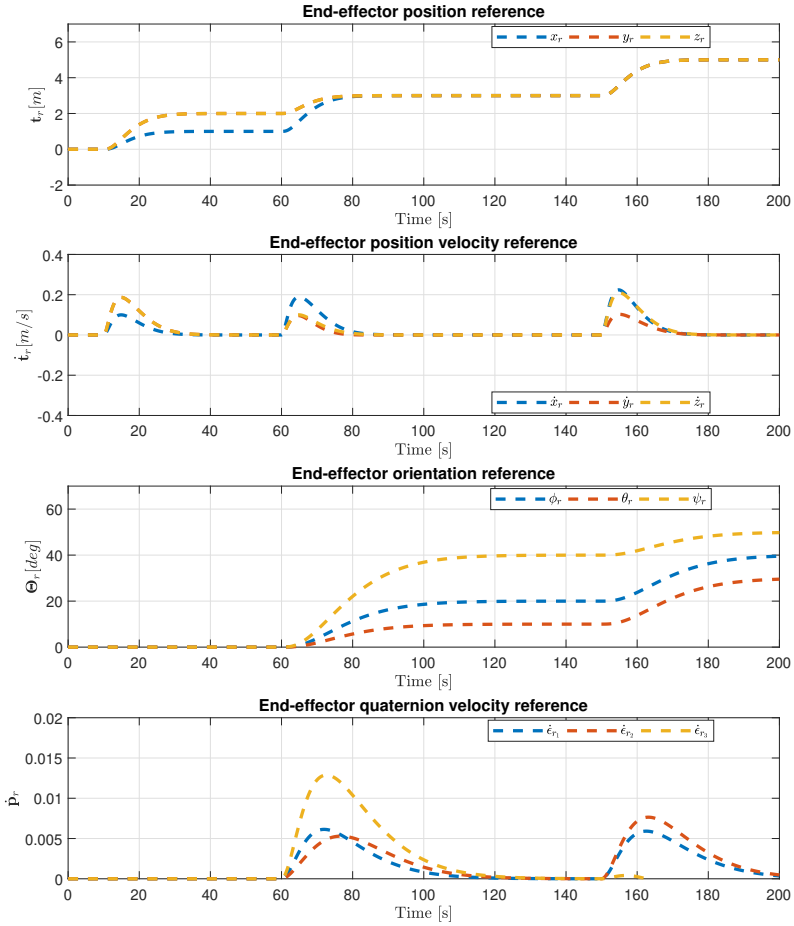

Figure 12.1: Scenario III: End-effector reference trajectories.

Control torque

The control torque in mode 2 within the operational space formulation for scenario III is given by (3.18) as

$$\boldsymbol{\tau} = \boldsymbol{\tau}_\sigma + \boldsymbol{N}_\sigma \boldsymbol{\tau}_e \tag{12.6}$$

where the torques $\boldsymbol{\tau}_\sigma$ and $\boldsymbol{\tau}_e$ are given by eqs. (8.7) and (8.11) for the PID-controller; eqs. (7.2) and (7.6) for the STA controller; and eqs. (7.8) and (7.12) for the GSTA controller. Whenever the system is in mode 1, the control torque is simply

$$\boldsymbol{\tau} = \boldsymbol{\tau}_e \tag{12.7}$$

In sections 12.2 to 12.4 the simulation results for the PID-controller, STA with adaptive gains and the GSTA are shown for scenario III, respectively. Figures 12.2, 12.4 and 12.6 contains four subplots which display the actuation index, derivative, error and the mode during simulation for the PID, STA and GSTA, respectively. Furthermore, figs. 12.3, 12.5 and 12.7 contains four subplots which display the end-effector position error, velocity error, orientation error and quaternion velocity error for the PID, STA and GSTA, respectively.

12.2 PID-control

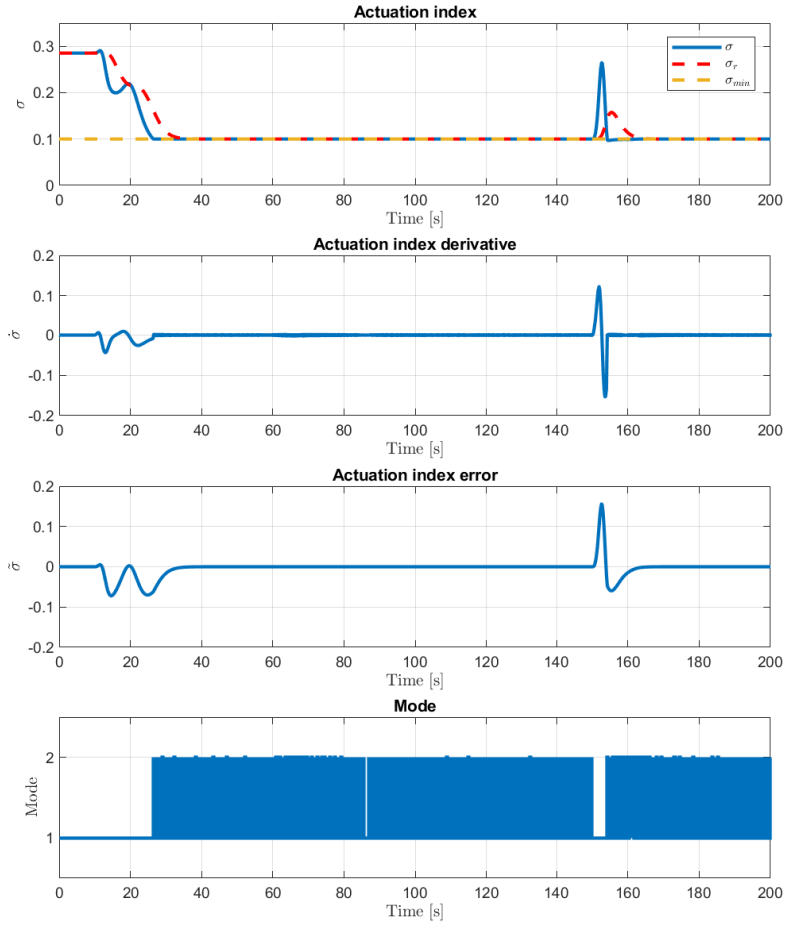

Figure 12.2: Scenario III: Actuation index and mode with PID controller.

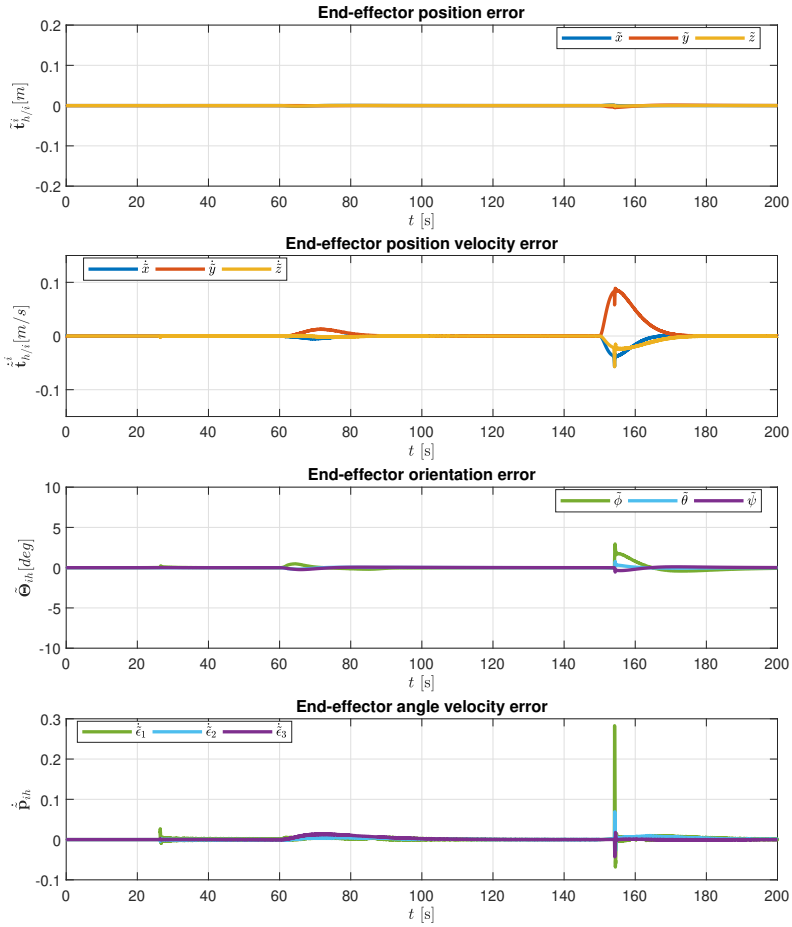

Figure 12.3: Scenario III: End-effector position and orientation error with PID controller.

12.3 Super-twisting with adaptive gains

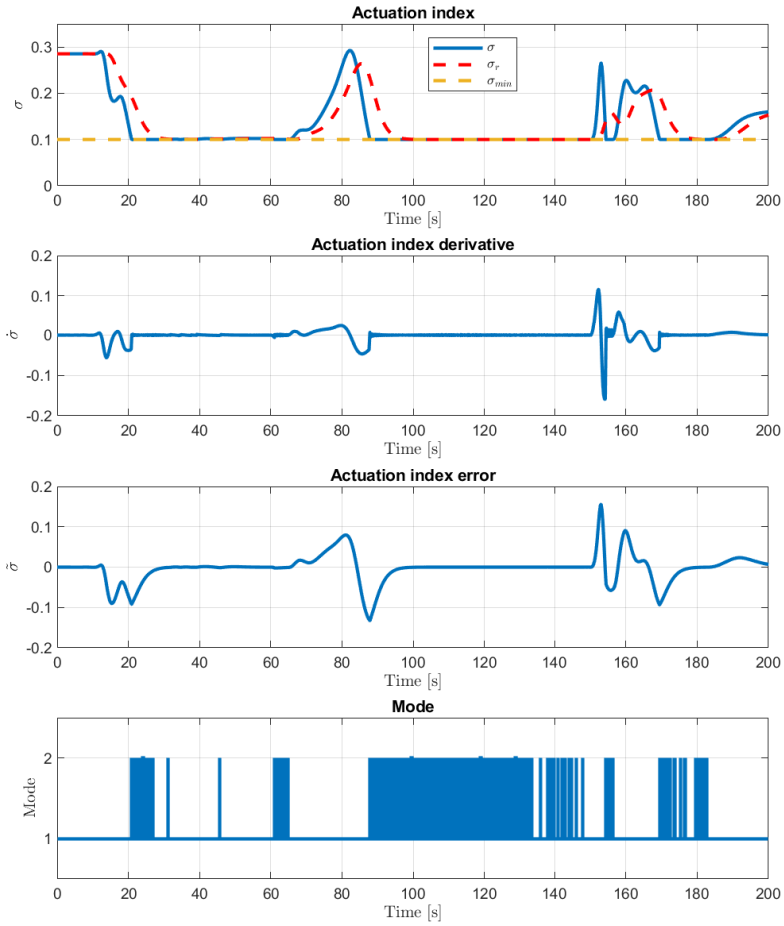

Figure 12.4: Scenario III: Actuation index and mode with STA controller.

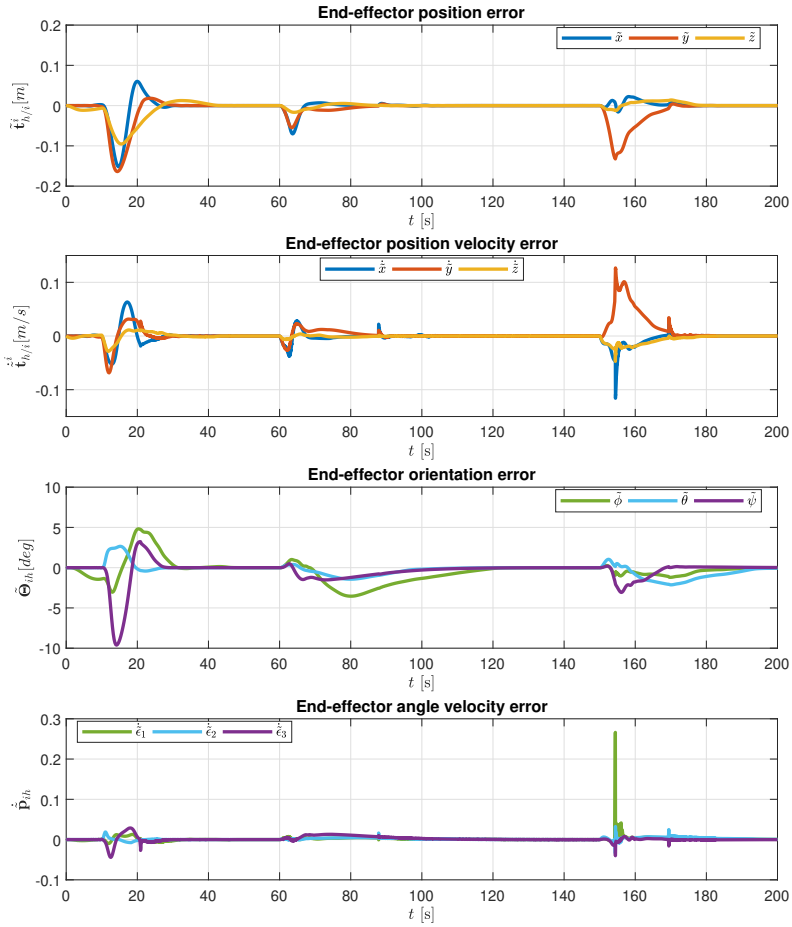

Figure 12.5: Scenario III: End-effector position and orientation error with STA controller.

12.4 Generalized super-twisting algorithm

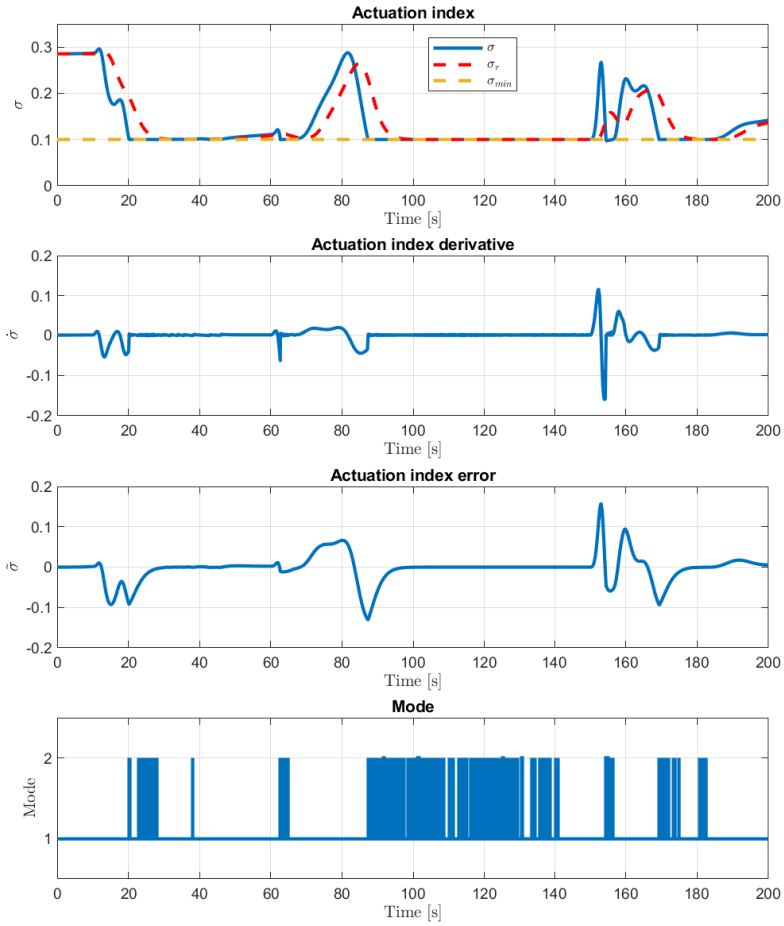

Figure 12.6: Scenario III: Actuation index and mode with GSTA controller.

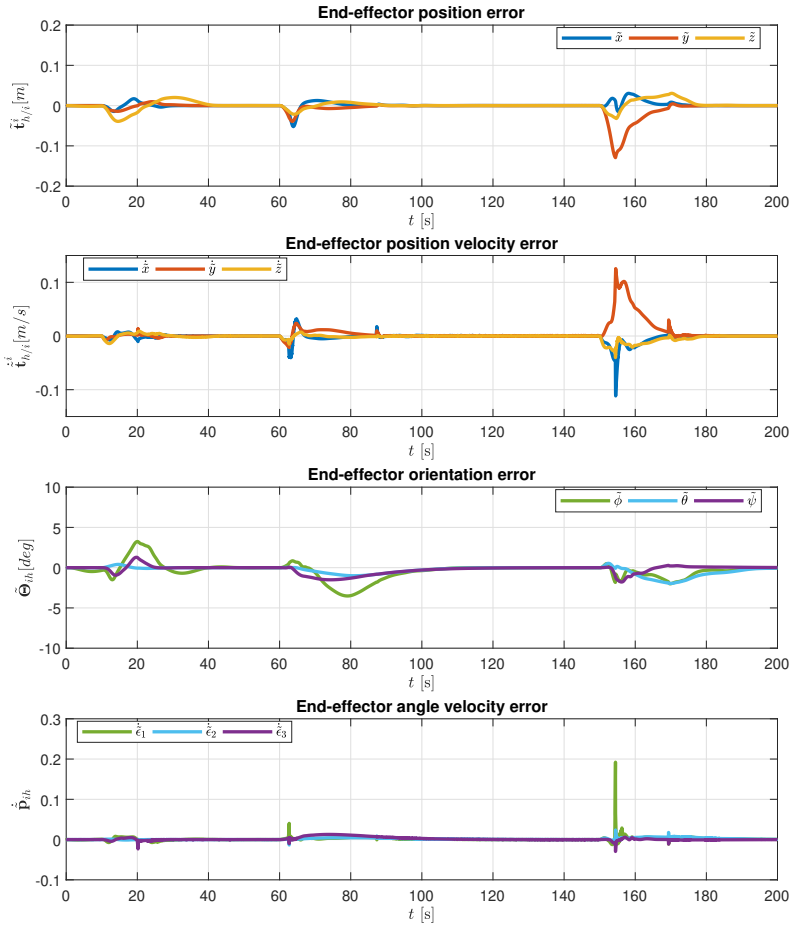

Figure 12.7: Scenario III: End-effector position and orientation error with GSTA controller.

12.5 Discussion

Figure 12.8 shows the traversed end-effector trajectories for each controller in scenario III. There are obvious deviations in the traversed path for the different controllers, and in the next paragraphs the origin of these deviations, among other observations from the results in sections 12.2 to 12.4, will be discussed.

The development of the actuation index can be seen in figs. 12.2, 12.4 and 12.6 and all controllers are able to enforce the actuation index when required, ensuring that the constraint σ_{min} is not violated. The mode is evolving as expected, where high frequency switching in the mode can be seen when the actuation index is at σ_{min}. This switching behavior is a result of the set-based task implementation [27] and can also be seen in the simulations done within the inverse kinematics framework in [9], where a manipulability index was implemented as a set-based task according to [27] to avoid kinematic singularities.

The most notable difference between the controllers for the actuation index is that the PID-controller is much stricter when it comes to reestablishing a consistent mode 1. As a result, the actuation index is allowed to evolve freely more often for the SMCs, which can be seen in subplot 4 of figs. 12.2, 12.4 and 12.6. The PID-controller keeps the high-frequent mode switching in the period $t \in [60, 90]$ where the SMCs transition into a consistent mode 1.

Figure 12.9 shows the actuation index derivative, which determines the mode through the extended tangent cone function (5.30), for every controller in scenario III and the corresponding mode. It focuses on the interval before the deviation where the SMCs enter mode 1 and the PID-controller remains stationary. The PID-controller is a lot more aggressive, due to the lower-priority task gains having to be tuned aggressively, which causes the derivative to cross through zero almost at every time-step. The SMCs are a lot less aggressive, allowing the mode to remain stationary through multiple time-steps. As a result, whenever an end-effector reference change which implies a configuration shift where the actuation index increases occurs, the higher-priority task does not interfere where it is not necessary. However, as the PID-controller switches so frequently, the configuration change is suppressed by the higher priority task, resulting in the actuation index remaining active.

The end-effector position and orientation behaves slightly differently for the PID-controller than the SMCs. The error coordinates in figs. 12.3, 12.5 and 12.7 shows a better tracking in both position and orientation for the PID-controller, although the difference is not large. It should also be taken into account that

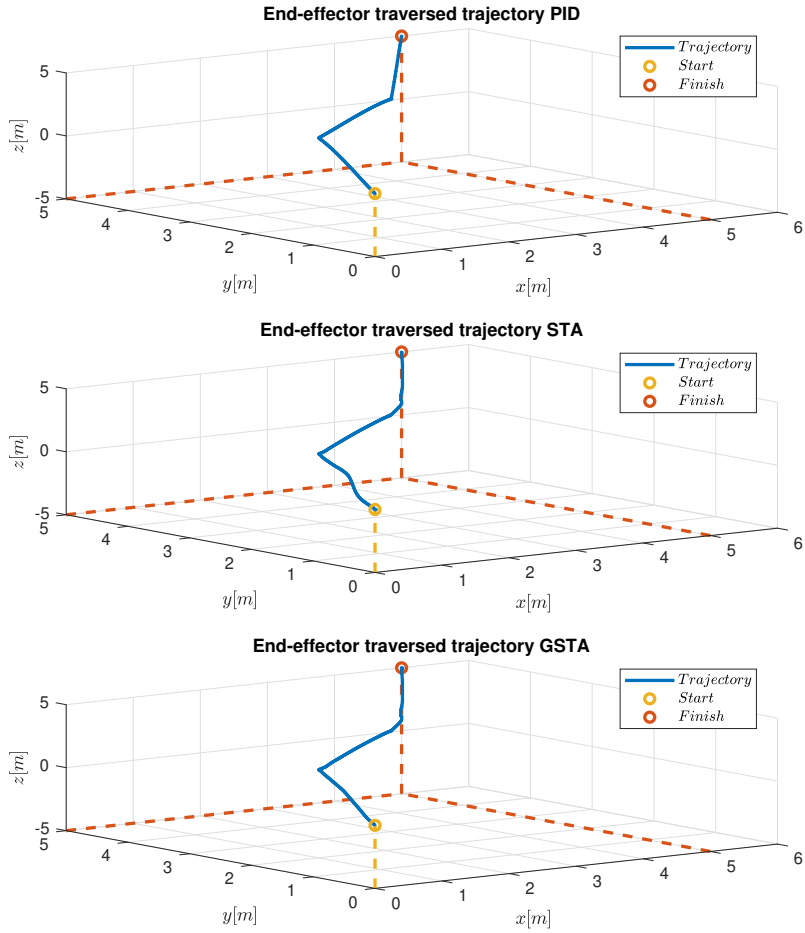

Figure 12.8: Scenario III: AIAUV end-effector trajectories.

the manipulator configuration as a whole, as mentioned previously, is evolving entirely differently in the SMCs. As such, a direct comparison of the transient errors in this scenario is difficult. The two SMC controllers behave very similarly when it comes to configuration, and it can be seen from figs. 12.5 and 12.7 that

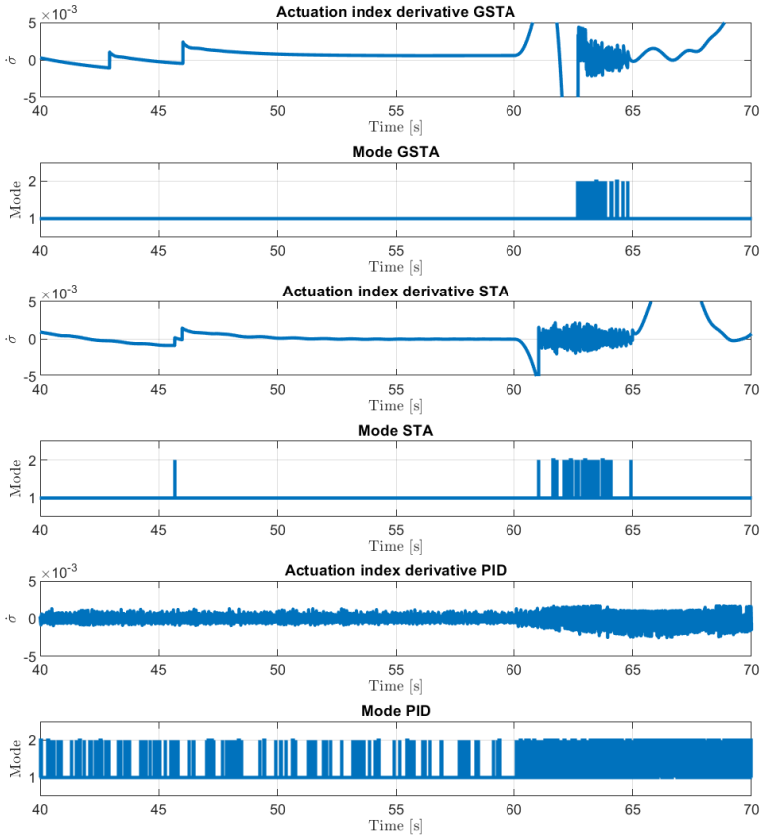

Figure 12.9: Scenario III: Actuation index derivative behavior.

	RMSE		
	GSTA	STA	PID
x [m]	7.4231e-3	2.1690e-2	3.1430e-4
y [m]	1.8574e-2	3.2079e-2	7.6578e-4
z [m]	1.0972e-2	1.7460e-2	2.7202e-4
ϕ [deg]	1.1046	1.4146	2.9738e-1
θ [deg]	5.8180e-1	8.1267e-1	5.6501e-2
ψ [deg]	5.4261e-1	1.4747	7.7629e-2

Table 12.1: Root-Mean-Square-Error for scenario III

the GSTA controller performs better than STA in the first transition for the end-effector error, while in the subsequent transitions they both perform the same.

All controllers have peaks in their velocities, both position and orientation, which is a result of the set-based task being introduced through a torque switching framework. As can be seen in fig. 12.10, the applied joint and thruster torques computed are no longer continuous due to the switching, which can lead to more wear and tear on the actuators. The higher-order SMC controllers, STA and GSTA, hide the non-continuous switching of the sign function behind an integrator to attain a continuous control signal. However, by introducing the set-based task, the controllers once again have a non-continuous control signal. Figure 12.10 also shows that the SMCs, especially GSTA, uses less force than the PID-controller. Since the PIC-controller is using more force, it is only natural to expect it should achieve better tracking. Even so, the GSTA shows very good performance. The extensive use of force for the PID-controller is a result of the aggressive tuning required to enforce the lower-priority task, which can be seen from the high gains in (8.6).

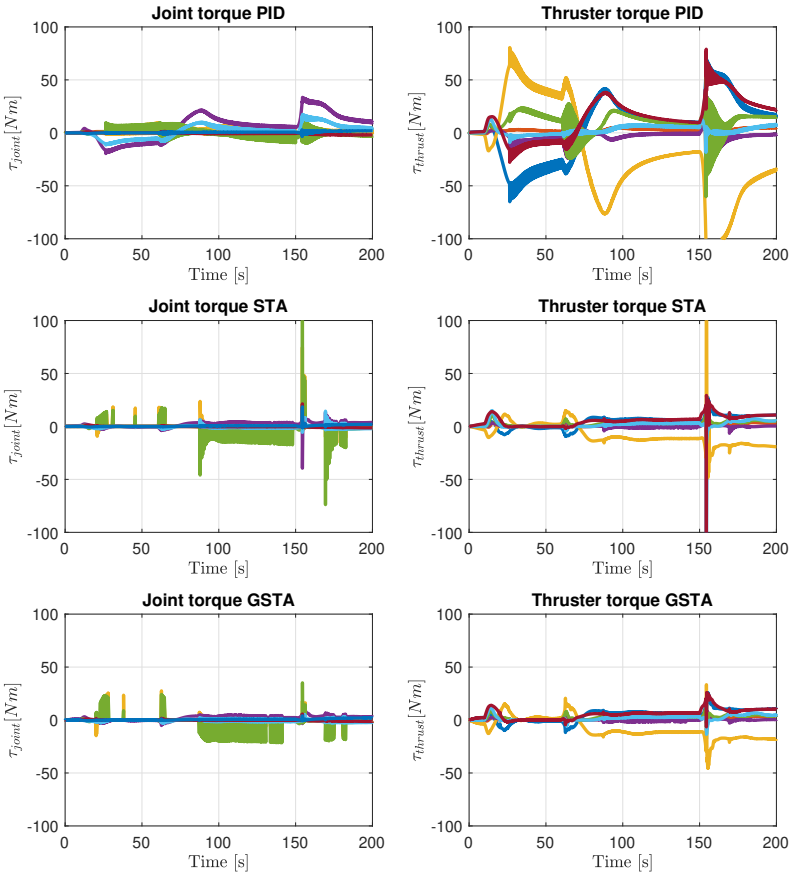

Figure 12.10: Scenario III: Applied joint and thruster torques.

Chapter 13

Analysis

This chapter takes a look at the overall performance of each controller based on the results in chapters 10 to 12. In addition, observations regarding other methods used in the simulations, e.g. the operational space formulation and the actuation index, are given.

13.1 Controller performance

From the simulations in chapters 10 to 12 it can be seen that all controllers perform well within some categories, while there are weaknesses evident in some situations. The PID controller performs well in tracking and precision in most scenarios, but there are some issues with oscillations as can be seen in section 11.3. Furthermore, as this thesis is primarily looking into robust control, it must not be forgotten that the PID-controller includes perfect feedback compensation, whereas the SMCs does not utilize any model dependent feedback. Control gain tuning was very difficult when attempting to use the controller without compensation, which led to the decision of using feedback compensation. Compared to the SMC solutions, the PID-controller was also more difficult to tune for the multi-priority framework. An example is scenario II, where the SMCs could be used with the same controller gains when the priorities of the tasks were swapped, while the PID-controller required extensive retuning when switching the priorities, and the lower-priority gains had to be tuned very aggressively to achieve convergence.

The STA controller shows issues with oscillations due to the adaptive controller gain having to converge early on in simulations. These issues could be reduced through further tuning of the static controller gains and adjustments of the initial values. However, the core of an adaptive approach is the ability to adapt to different scenarios and the convergence drawback is to be expected. Although the controller has problems in the transient, it repeatedly performs better when entering the stationary parts of the simulations, where it often performs on par with the other controllers.

Finally, the GSTA controller displays performance on par or better than the other controllers in all simulated scenarios. In the transient, the controller is often slightly worse that the PID-controller, although the PID uses compensation, and it performs better than the STA. The stationary accuracy outperforms the other methods in both scenarios in chapter 10.

13.2 Higher priority task effect on end-effector

The joint angles and actuation index tasks are enforced above the end-effector, and it can be clearly seen which task affects the end-effector the most. Maintaining the joint angles constant forces the manipulator to move in a set configuration at all times. From figs. 11.4, 11.6, 11.8, 12.3, 12.5 and 12.7, keeping in mind the difference on the y-axis, the error can be seen to be consistently higher for the joint angles scenario in chapter 11. The same can also be concluded from the coordinate plots, where oscillations are much more discernible, especially in the orientation. The actuation index, while also enforcing a physical constraint on the manipulator, allows the joints to change as long as the set-based task is achieved. As a result, the actuation index is less intrusive on the end-effector than the joint angles task.

As an additional observation, the operational space formulation minimizes the kinetic energy of the manipulator [18]. When moving the end-effector some distance, the configuration with the least kinetic energy will obviously be the I-shape. This means the end-effector lower-priority task computes torques which favor the I-shape, while the joint angles are actively working against this configuration. While the tasks are not incompatible, this could mean end-effector torques are prioritized in a way which causes more torque than necessary to be canceled in the null-space of the joint angle task. The same is also true for the actuation index, although more freedom is granted as the joint angles can evolve even on the limit.

13.3 Actuation index and the joint angle task

None of the simulations in chapters 10 to 12 uses the actuation index and joint angle tasks at the same time. Both tasks affect the configuration of the manipulator and problems may arise where the tasks are no longer compatible.

The actuation index minimum value σ_{min} creates a subspace $\mathcal{D}$ of feasible configurations $\mathcal{C}$ where

$$\mathcal{D} = \{\mathcal{C} | \sigma \geq \sigma_{min}\} \tag{13.1}$$

Furthermore, setting a step reference for the joint angles, which is fed through the trajectory generator, creates a sequence of configurations which the manipulator plans to go through. If this sequence contains an element $\mathcal{C}_q \notin \mathcal{D}$ the two tasks will actively work against each other. Provided that the sequence does not just barely drop below σ_{min} and straight up again, the joint angle task torque will be canceled by the actuation index null-space projector, or vice-versa if the priorities are reversed. Mixed with the switching behavior of the set-based task implementation, the combination of the tasks on σ_{min} will cause high-frequent oscillations in joint velocities.

A possibility is to use the actuation index to construct a configuration space, and then apply this space to a path planner for the joint angles. This way the strength of both tasks could be exploited, although the complexity of applying a path planner might not always be desired.

13.4 Controller torque

In section 2.2.4 it was mentioned that the actuator configuration matrix $\boldsymbol{B}(\boldsymbol{q})$ might become singular. If the matrix is singular, the torque allocation scheme will be affected and unnaturally large torque peaks will occur. Figure 13.1 displays the torque in scenario II with a GSTA controller where the simulation has been done without least squares damping, corresponding to (2.15), and with least squares damping, corresponding to (2.16). From the figures it is clear that the least squares damping has greatly improved the torque allocation, and the simulations in chapters 10 to 12 have not been noticeably changed compared to the simulation done before damping was introduced.

The actuator configuration matrix is singular whenever the actuation index is zero. As such, this problem was not encountered in any simulations in section 10.2

and chapter 12 where the actuation index high-priority task ensured that the matrix was never singular. For this reason, the actuation index task removes the necessity of least squares damping in the torque allocation scheme. However, if the task is to control the joint angles, the least-squares damping should be used.

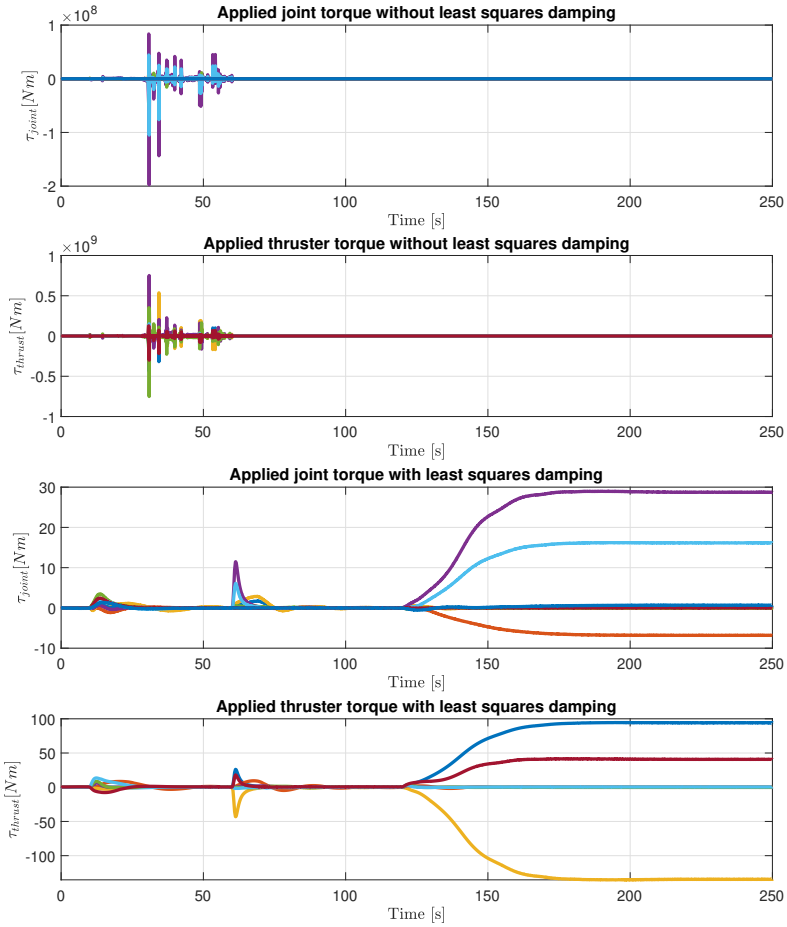

Figure 13.1: Scenario II: GSTA controller torque with and without least squares damping.

Chapter 14

Conclusions and Future Work

This chapter draws a conclusion based on the simulation results achieved in the thesis. Afterwards, suggested future work is presented.

14.1 Conclusion

This thesis has examined the performance of robust higher order sliding mode controllers in the operational space formulation for an Articulated Intervention AUV (AIAUV). Specifically, the generalized super-twisting algorithm and the super-twisting algorithm with adaptive gains has been compared to a PID-controller with feedback compensation. The controllers were tuned to a set of gains, and then applied to a set of scenarios demonstrating real applications of the AIAUV where different control tasks, namely the actuation index, joint angles and end-effector position and orientation, were used in several task-priorities.

In conclusion, the PID controller performs very well with perfect feedback compensation, although the GSTA achieves performance on par without any prior model knowledge, demonstrating robustness. The STA struggles with adaptive gain convergence, although the performance increases once the gains have converged after the initial transient.

The operational space task-priority framework was shown to be well suited for the AIAUV platform, where several tasks were implemented. Simulations show that lower-priority tasks were able to converge for all controllers even when projected into the null-space of a higher-priority task as long as the control objectives were compatible.

The actuation index task was introduced in the specialization project and further examined with more advanced controllers in this thesis. The positive results from the project was enforced by the sliding mode controllers where the minimum value was strictly enforced while being more dexterous in the task transitioning.

14.2 Future work

The thesis still leaves opportunities for potential future research projects, some of which are mentioned here.

The operational space formulation requires model certainty to completely decouple the null space and operational space dynamics through the weighted pseudoinverse [18]. Future research should examine the effects of uncertainties in the weighted pseudoinverse and how the task hierarchy and convergence is affected by the null space and operational space not being perfectly decoupled. In addition, simulations including external disturbances, e.g. current, should be performed.

This thesis has not delved deeply into the theoretical stability analysis of the sliding mode controllers in the operational space. The stability of single and multiply task frameworks for the AIAUV should be examined on a theoretic level. Also, experiments with the actuation index and operational space control should be conducted on the real Eely AIAUV.

An optional part of this thesis was implementing the adaptive backstepping controller in [18]. Background theory and some work in this area is given in appendix B, but further implementation in the AIAUV simulator has been left as future work.

Bibliography

[1] J. Sverdrup-Thygeson, E. Kelasidi, K. Y. Pettersen & J. T. Gravdahl, "Modeling of underwater swimming manipulators," *IFAC PapersOnLine*, vol. 49, no. 23, 2016, ISSN: 2405-8963.

[2] O. Khatib, "A unified approach for motion and force control of robot manipulators: The operational space formulation," *Robotics and Automation, IEEE Journal of*, vol. 3, no. 1, 1987, ISSN: 0882-4967.

[3] H. R. Widditsch, "SPURV - The First Decade," US Dept of the Navy, Tech. Rep., Oct. 1973. DOI: 10.21236/ADA050816. [Online]. Available: http://www.dtic.mil/docs/citations/ADA050816.

[4] Kongsberg, *Kongsberg's HUGIN AUV*. [Online]. Available: https://www.km.kongsberg.com/ks/web/nokbg0240.nsf/AllWeb/B3F87A63D8E419E5C1256A68004E946C?OpenDocument (visited on 03/05/2019).

[5] S. Hirose, *Biologically Inspired Robots: Snake-Like Locomotors and Manipulators*. Oxford University Press, 1993.

[6] P. Liljebäck, Ø. Stavdahl & A. Beitnes, "SnakeFighter - Development of a water hydraulic fire fighting snake robot," *9th International Conference on Control, Automation, Robotics and Vision, 2006, ICARCV '06*, no. 7465, 2006. DOI: 10.1109/ICARCV.2006.345311.

[7] A. M. Kohl, *Guidance and Control of Underwater Snake Robots Using Planar Sinusoidal Gaits*, 2017. [Online]. Available: http://hdl.handle.net/11250/2465169.

[8] G. Marani, S. K. Choi & J. Yuh, "Underwater autonomous manipulation for intervention missions AUVs," *Ocean Engineering*, vol. 36, no. 1, 2009, ISSN: 0029-8018. DOI: https://doi.org/10.1016/j.oceaneng.2008.08.007. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S002980180800173X.

[9] J. Sverdrup-Thygeson, S. Moe, K. Y. Pettersen & J. T. Gravdahl, "Kinematic singularity avoidance for robot manipulators using set-based manipulability tasks," in *Proc. IEEE Conf. Control Technology and Applications*, Kohala Coast, Hawaii USA, 2017.

[10] I.-L. Borlaug, K. Y. Pettersen & J. T. Gravdahl, "Trajectory tracking for an articulated intervention AUV using a super-twisting algorithm in 6 DOF," *IFAC-PapersOnLine*, vol. 51, no. 29, Jan. 2018, ISSN: 2405-8963. DOI: 10.1016/J.IFACOL.2018.09.506. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S2405896318321955?via%7B%5C%%7D3Dihub.

[11] I. L. G. Borlaug, K. Y. Pettersen & J. T. Gravdahl, "Tracking control of an articulated intervention-AUV in 6DOF using the generalized super-twisting algorithm," in *American Control Conference*, 2019.

[12] G. Herrmann, J. Jalani, M. Nasiruddin Mahyuddin & C. Melhuish, "Robotic hand posture and compliant grasping control using operational space and integral sliding mode control," *Robotica*, vol. 34, 2016. DOI: 10.1017/S0263574714002811. [Online]. Available: https://search.proquest.com/docview/1818758653?rfr%7B%5C_%7Did=info%7B%5C%%7D3Axri%7B%5C%%7D2Fsid%7B%5C%%7D3Aprimo.

[13] C. Barbalata, M. W. Dunnigan & Y. Petillot, "Position/force operational space control for underwater manipulation," *Robotics and Autonomous Systems*, vol. 100, Feb. 2018, ISSN: 0921-8890. DOI: 10.1016/J.ROBOT.2017.11.004. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S092188901730386X.

[14] D. Kim, Y. Zhao, G. Thomas, B. R. Fernandez & L. Sentis, "Stabilizing Series-Elastic Point-Foot Bipeds Using Whole-Body Operational Space Control," *IEEE Transactions on Robotics*, vol. 32, no. 6, Dec. 2016, ISSN: 1552-3098. DOI: 10.1109/TRO.2016.2597314. [Online]. Available: http://ieeexplore.ieee.org/document/7736085/.

[15] V. I. Utkin, *Sliding modes in control and optimization*, Berlin, Germany, 1992.

[16] P. A. Ioannou, *Robust adaptive control*, Mineola, N.Y, 2012.

[17] Y. Shtessel, C. Edwards, L. Fridman & A. Levant, *Sliding Mode Control and Observation*, ser. Control Engineering. New York, NY: Springer New York, 2014, ISBN: 978-0-8176-4892-3.

[18] K. P. Tee & R. Yan, "Adaptive operational space control of redundant robot manipulators," in *Proceedings of the 2011 American Control Conference*, IEEE, 2011, ISBN: 9781457700804.

[19] W. Lu & D. Liu, "Active Task Design in Adaptive Control of Redundant Robotic Systems," in *Australasian Conf. Robotics and Automation 2017*, 2017.

[20] I. L. G. Borlaug, J. T. Gravdahl, J. Sverdrup-Thygeson, K. Y. Pettersen & A. Loria, "Trajectory Tracking for Underwater Swimming Manipulators using a Super Twisting Algorithm," *Asian Journal of Control*, Aug. 2018, ISSN: 15618625. DOI: 10.1002/asjc.1840. [Online]. Available: http://doi.wiley.com/10.1002/asjc.1840.

[21] J. A. Moreno, "A linear framework for the robust stability analysis of a Generalized Super-Twisting Algorithm," in *2009 6th International Conference on Electrical Engineering, Computing Science and Automatic Control (CCE)*, IEEE, 2009, ISBN: 9781424446889.

[22] I. Castillo, L. Fridman & J. A. Moreno, "Super-Twisting Algorithm in presence of time and state dependent perturbations," *International Journal of Control*, vol. 91, no. 11, 2018, ISSN: 0020-7179.

[23] Y. B. Shtessel, J. A. Moreno, F. Plestan, L. M. Fridman & A. S. Poznyak, "Super-twisting adaptive sliding mode control: A Lyapunov design," in *49th IEEE Conference on Decision and Control (CDC)*, IEEE, Dec. 2010, ISBN: 978-1-4244-7745-6. DOI: 10.1109/CDC.2010.5717908. [Online]. Available: http://ieeexplore.ieee.org/document/5717908/.

[24] G. Marani, G. Jinhyun Kim, G. Junku Yuh & G. Wan Kyun Chung, "A real-time approach for singularity avoidance in resolved motion rate control of robotic manipulators," in *Robotics and Automation, 2002. Proceedings. ICRA '02. IEEE International Conference on*, vol. 2, USA: IEEE, 2002, ISBN: 0780372727.

[25] T. Yoshikawa, "Manipulability of Robotic Mechanisms," *The International Journal of Robotics Research*, vol. 4, no. 2, 1985. DOI: 10.1177/027836498500400201. [Online]. Available: https://doi.org/10.1177/027836498500400201.

[26] J. Sverdrup-Thygeson, E. Kelasidi, K. Y. Pettersen & J. T. Gravdahl, "The Underwater Swimming Manipulator—A Bioinspired Solution for Subsea Operations," *Oceanic Engineering, IEEE Journal of*, vol. 43, no. 2, 2018, ISSN: 0364-9059.

[27] S. Moe, G. Antonelli, A. R. Teel, K. Y. Pettersen & 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, 2016.

[28] G. Antonelli, P. D. Lillo & C. Natale, "Modelling Errors Analysis in Inverse Dynamics Approaches within a Task-Priority Framework," in *Proc. IEEE Conf. Control Technology and Applications*, I, Aug. 2018.

[29] H. Schmidt-Didlaukies, A. J. Sørensen & K. Y. Pettersen, "Modeling of Articulated Underwater Robots for Simulation and Control," Trondheim, 2018.

[30] M. W. Spong, *Robot modeling and control*, Hoboken, N.J, 2006.

[31] G. Antonelli, *Underwater robots*, 3rd ed. 20, ser. Springer tracts in advanced robotics. Cham, Switzerland: Springer, 2014, vol. Volume 96, ISBN: 3-319-02877-4. DOI: 10.1007/978-3-319-02877-4. [Online]. Available: http://link.springer.com/10.1007/978-3-319-02877-4.

[32] K. B. Petersen & M. S. Pedersen, "The Matrix Cookbook," Tech. Rep., 2012.

[33] A. Dietrich, C. Ott & A. Albu-Schäffer, "An overview of null space projections for redundant, torque-controlled robots," *The International Journal of Robotics Research*, vol. 34, no. 11, Sep. 2015, ISSN: 0278-3649. DOI: 10.1177/0278364914566516. [Online]. Available: http://journals.sagepub.com/doi/10.1177/0278364914566516.

[34] L. Sentis & O. Khatib, "Prioritized multi-objective dynamics and control of robots in human environments," in *2004 4th IEEE-RAS International Conference on Humanoid Robots*, vol. 2, 2004, ISBN: 0780388631.

[35] T. I. Fossen, *Handbook of Marine Craft Hydrodynamics and Motion Control*. Wiley, 2011.

[36] U. Joergensen & J. Tommy Gravdahl, "Observer Based Sliding Mode Attitude Control: Theoretical and Experimental Results," *Modeling, Identification and Control*, vol. 32, no. 3, 2011, ISSN: 1890-1328. [Online]. Available: http://search.proquest.com/docview/1136420774/.

[37] P. Bogacki & L. F. Shampine, "A 3(2) pair of Runge - Kutta formulas," *Applied Mathematics Letters*, vol. 2, no. 4, Jan. 1989, ISSN: 0893-9659. DOI: 10.1016/0893-9659(89)90079-7. [Online]. Available: https://www.sciencedirect.com/science/article/pii/0893965989900797?via%7B%5C%%7D3Dihub.

[38] I. L. G. Borlaug, K. Y. Pettersen & J. T. Gravdahl, "Tracking control of an articulated intervention autonomous underwater vehicle in 6DOF using generalized super-twisting: theory and Experiments," *IEEE Transactions of Control Systems Technology*, 2019.

[39] H. K. Khalil, *Nonlinear Systems*, 3rd. Prentice Hall, 2002.

[40] M. Krstić, *Nonlinear and adaptive control design*, New York, 1995.

[41] X. Xin & M. Kaneda, "Swing-Up Control for a 3-DOF Gymnastic Robot With Passive First Joint: Design and Analysis," *IEEE Transactions on Robotics*, vol. 23, no. 6, Dec. 2007, ISSN: 1552-3098. DOI: 10.1109/TRO.2007.909805. [Online]. Available: http://ieeexplore.ieee.org/document/4399957/.

Appendix A

Matrix Algebra

The following section introduces matrix algebra used in derivations of the actuation index task specific Jacobian and its derivative. All properties have been found in [32].

A.1 Matrix transpose

An element a_{ij} of a transposed matrix is defined as

$$\left[\mathbf{A}^T\right]_{ij} = [\mathbf{A}]_{ji} \tag{A.1}$$

The transpose of a matrix does not require the matrix to be square. The transpose of an $n \times m$ matrix is an $m \times n$ matrix with entries defined by (A.1). For the transpose, the following properties hold

$$(\mathbf{CBA})^T = \mathbf{A}^T\mathbf{B}^T\mathbf{C}^T \tag{A.2a}$$

$$\left(\mathbf{A}^{-1}\right)^T = \left(\mathbf{A}^T\right)^{-1} \tag{A.2b}$$

$$\left(\mathbf{A}^T\right)^T = \mathbf{A} \tag{A.2c}$$

$$\left(\mathbf{A}\mathbf{A}^T\right)^T = \mathbf{A}\mathbf{A}^T \tag{A.2d}$$

A.2 Trace

The trace of an $n \times n$ matrix A is given as the sum of the diagonal entries in the matrix as

$$\mathrm{Tr}(\mathbf{A}) = \mathrm{Tr}\left(\begin{bmatrix} a_{11} & a_{12} & \cdots & a_{1n} \\ a_{21} & a_{22} & \cdots & a_{2n} \\ \vdots & \vdots & \ddots & \vdots \\ a_{n1} & a_{n2} & \cdots & a_{nn} \end{bmatrix}\right) = \sum_{i}^{n} a_{ii} \tag{A.3}$$

Note that the trace is only defined for square matrices. The trace of a matrix has the following properties

$$\mathrm{Tr}\left(\mathbf{AB}\right) = \mathrm{Tr}\left(\mathbf{BA}\right) \tag{A.4a}$$
$$\mathrm{Tr}\left(\mathbf{A}+\mathbf{B}\right) = \mathrm{Tr}\left(\mathbf{A}\right) + \mathrm{Tr}\left(\mathbf{B}\right) \tag{A.4b}$$
$$\mathrm{Tr}\left(\mathbf{ABC}\right) = \mathrm{Tr}\left(\mathbf{BCA}\right) = \mathrm{Tr}\left(\mathbf{CAB}\right) \tag{A.4c}$$
$$\mathrm{Tr}\left(\mathbf{A}^T\right) = \mathrm{Tr}(\mathbf{A}) \tag{A.4d}$$

A.3 Matrix differentiation

For differentiation of matrices, the following properties hold

$$\partial\left(\mathbf{XY}\right) = \left(\partial\mathbf{X}\right)\mathbf{Y} + \mathbf{X}\left(\partial\mathbf{Y}\right) \tag{A.5a}$$
$$\partial\mathbf{X}^T = \left(\partial\mathbf{X}\right)^T \tag{A.5b}$$
$$\partial\,\mathrm{Tr}\left(\mathbf{X}\right) = \mathrm{Tr}\left(\partial\mathbf{X}\right) \tag{A.5c}$$
$$\frac{\partial\mathbf{Y}^{-1}}{\partial x} = -\mathbf{Y}^{-1}\frac{\partial\mathbf{Y}}{\partial x}\mathbf{Y}^{-1} \tag{A.5d}$$
$$\frac{\partial\det\left(\boldsymbol{Y}\right)}{\partial x} = \det\left(\boldsymbol{Y}\right)\mathrm{Tr}\left(\boldsymbol{Y}^{-1}\frac{\partial\boldsymbol{Y}}{\partial x}\right) \tag{A.5e}$$

In addition, the total derivative of a matrix $\boldsymbol{Y}$ is

$$\frac{d}{dt}\left(\boldsymbol{Y}\left(\boldsymbol{q}\left(t\right)\right)\right) = \frac{\partial\boldsymbol{Y}}{\partial t}\frac{\partial t}{\partial t} + \frac{\partial\boldsymbol{Y}}{\partial\boldsymbol{q}}\frac{\partial\boldsymbol{q}}{\partial t} \tag{A.6}$$

Appendix B

Adaptive Backstepping

The adaptive backstepping controller presented in [18] was provided as an optional controller in the problem description, to be implemented if there was time. Time has been spent looking into [18] and deriving a few derivatives, which are stated at the end of appendix B, but ultimately the implementation for the AIAUV has not been successful. To establish a foundation for future work with the method, the background theory is included here. Furthermore, this chapter will describe the work done on the implementation and the challenges encountered. In addition, some considerations and ideas concerning these challenges will be presented.

B.1 Adaptive backstepping in the operational space

The dynamically consistent generalized inverse $\bar{\boldsymbol{J}}_x$ [18], or the weighted pseudoinverse as it is named in [28], is given in (3.15b). In the AIAUV there are uncertainties present in the inertial matrix, $\boldsymbol{M}$, which leads to uncertainties in the weighted pseudoinverse. The weighted pseudoinverse is used in computation of the torque (3.18) as

$$\boldsymbol{\tau} = \boldsymbol{\tau}_a + \boldsymbol{N}_a \boldsymbol{\tau}_b \tag{B.1}$$

where the null-space projector is (3.17)

$$\boldsymbol{N}_x = \boldsymbol{I}_n - \boldsymbol{J}_x^T \bar{\boldsymbol{J}}_x^T \tag{B.2}$$

Remark: The derivations in the remainder of this chapter closely follows the derivations in [18]. They are restated in full to provide background for the choices of different parameters in the adaptive backstepping implementation. Some adaptations are made to the original derivations, namely including the hydrodynamic damping matrix as part of the derivations and extending the state vector.

In the spirit of robust adaptive control, the uncertainties in $\bar{\boldsymbol{J}}_x$ can be handled by defining

$$\tilde{\bar{\boldsymbol{J}}}_x = \hat{\bar{\boldsymbol{J}}}_x - \bar{\boldsymbol{J}}_x \tag{B.3}$$

where $\hat{\bar{\boldsymbol{J}}}_x$ is an estimate of $\bar{\boldsymbol{J}}_x$. Furthermore, the weighted pseudoinverse is nonlinearly parametrized due to the presence of $\boldsymbol{M}^{-1}$ in $\bar{\boldsymbol{J}}_x$, which means it can be written as [18]

$$\bar{\boldsymbol{J}}_x = \frac{1}{d\left(\boldsymbol{q}, \boldsymbol{\phi}_d\right)} \bar{\boldsymbol{J}}_n \tag{B.4}$$

where d is a scalar and the matrix $\bar{\boldsymbol{J}}_x$ is linearly parametrized as

$$d\left(\boldsymbol{q}, \boldsymbol{\phi}_d\right) = \boldsymbol{\psi}_d \boldsymbol{\phi}_d = \left(\det\left(\boldsymbol{M}\right)\right)^{n-1} \det\left(\boldsymbol{J}_x \boldsymbol{J}_x^T\right) \tag{B.5a}$$

$$\bar{\boldsymbol{J}}_n = \begin{bmatrix} \boldsymbol{\psi}_{p_{11}}\left(\boldsymbol{q}\right) \boldsymbol{\phi}_n & \cdots & \boldsymbol{\psi}_{p_{1m}}\left(\boldsymbol{q}\right) \boldsymbol{\phi}_n \\ \vdots & \ddots & \vdots \\ \boldsymbol{\psi}_{p_{n1}}\left(\boldsymbol{q}\right) \boldsymbol{\phi}_n & \cdots & \boldsymbol{\psi}_{p_{nm}}\left(\boldsymbol{q}\right) \boldsymbol{\phi}_n \end{bmatrix} \tag{B.5b}$$

The different vectors introduced are

$$\boldsymbol{\psi}_d\left(\boldsymbol{q}\right) \in \mathbb{R}^{l_d} \tag{B.6a}$$

$$\boldsymbol{\phi}_d \in \mathbb{R}^{l_d} \tag{B.6b}$$

$$\boldsymbol{\psi}_{p_{ij}}\left(\boldsymbol{q}\right) \in \mathbb{R}^{l_n} \tag{B.6c}$$

$$\boldsymbol{\phi}_n \in \mathbb{R}^{l_n} \tag{B.6d}$$

For any $\boldsymbol{\rho} \in \mathbb{R}^n$

$$\bar{\boldsymbol{J}}_n^T\left(\boldsymbol{q}, \boldsymbol{\phi}_n\right) \boldsymbol{\rho} = \boldsymbol{\psi}_n\left(\boldsymbol{q}, \boldsymbol{\rho}\right) \boldsymbol{\phi}_n \tag{B.7}$$

B.1.1 Controller synthesis

This section adapts the control method described in [18] to the AIAUV. Adaptive control laws will be derived to handle nonlinearly parametrized uncertainties found in the generalized inverse (3.15b) and in the operational space dynamics (3.14). The uncertainties in the generalized inverse is handled through a modified quadratic control Lyapunov function [18], while the uncertainties in dynamics are dealt with by rearranging the dynamics in a linear time-varying parametrization, before dominating the time-varying parameters.

A backstepping design method [39] is used to create the adaptive control law for the operational space. The first step is chosen as

$$\boldsymbol{z} = \boldsymbol{x} - \boldsymbol{x}_d \tag{B.8a}$$
$$\boldsymbol{\nu} = \dot{\boldsymbol{x}} - \boldsymbol{\alpha} \tag{B.8b}$$

where $\boldsymbol{z}$ and $\boldsymbol{\nu}$ is introduced to follow the framework presented in [18]. The first Lyapunov function is chosen as

$$V_1 = \frac{1}{2}\boldsymbol{z}^T\boldsymbol{z} \tag{B.9}$$
$$\dot{V}_1 = \boldsymbol{z}^T\dot{\boldsymbol{z}} \tag{B.10}$$
$$\dot{V}_1 = \boldsymbol{z}^T\left(\boldsymbol{\nu} + \boldsymbol{\alpha} - \dot{\boldsymbol{x}}_d\right) \tag{B.11}$$
$$\tag{B.12}$$

At this point, the stabilizing function $\boldsymbol{\alpha}$ is set to

$$\boldsymbol{\alpha} = \dot{\boldsymbol{x}}_d - \boldsymbol{K}_z\boldsymbol{z} \tag{B.13}$$

such that

$$\dot{V}_1 = -\boldsymbol{z}^T\boldsymbol{K}_z\boldsymbol{z} + \boldsymbol{z}^T\boldsymbol{\nu} \tag{B.14}$$

where $\boldsymbol{K}_z > 0$ ensures the left term is negative definite. In the Lyapunov function for the second step in the backstepping algorithm, several additional terms are introduced as described in [18] to eliminate nonlinear parametrization. The Lyapunov function is

$$V_2 = V_1 + \frac{d}{2}\boldsymbol{\nu}^T\boldsymbol{M}_x\boldsymbol{\nu} + \frac{1}{2\gamma}\tilde{\boldsymbol{\beta}}^2 + \frac{1}{2}\tilde{\phi}_d^T\boldsymbol{\Gamma}_d^{-1}\tilde{\phi}_d + \frac{1}{2}\tilde{\phi}_n^T\boldsymbol{\Gamma}_n^{-1}\tilde{\phi}_n + \frac{1}{2}\tilde{\phi}_g^T\boldsymbol{\Gamma}_g^{-1}\tilde{\phi}_g \tag{B.15}$$

Here $\boldsymbol{\Gamma}_i$ are constant symmetric positive definite matrices, $\tilde{(\cdot)} = \hat{(\cdot)} - (\cdot)$ is the estimate error, $\boldsymbol{\phi}_g$ a vector of unknown parameters, $\gamma > 0$ a constant, and d as described in (B.5a). The vectors $\boldsymbol{\phi}_d$ and $\boldsymbol{\phi}_n$ are defined as in (B.6). Differentiating V_2 with respect to time yields

$$\begin{aligned}\dot{V}_2 = &-\boldsymbol{z}^T \boldsymbol{K}_z \boldsymbol{z} + \boldsymbol{z}^T \boldsymbol{\nu} + \frac{\dot{d}}{2} \boldsymbol{\nu}^T \boldsymbol{M}_x \boldsymbol{\nu} + \frac{d}{2} \boldsymbol{\nu}^T \frac{d}{dt}\left(\boldsymbol{M}_x\right) \boldsymbol{\nu} + \boldsymbol{\nu}^T d \boldsymbol{M}_x \dot{\boldsymbol{\nu}} \\ &+ \frac{1}{\gamma} \tilde{\boldsymbol{\beta}} \dot{\tilde{\boldsymbol{\beta}}} + \tilde{\boldsymbol{\phi}}_d^T \boldsymbol{\Gamma}_d^{-1} \dot{\tilde{\boldsymbol{\phi}}}_d + \tilde{\boldsymbol{\phi}}_n^T \boldsymbol{\Gamma}_n^{-1} \dot{\tilde{\boldsymbol{\phi}}}_n + \tilde{\boldsymbol{\phi}}_g^T \boldsymbol{\Gamma}_g^{-1} \dot{\tilde{\boldsymbol{\phi}}}_g \end{aligned} \tag{B.16}$$

Now, inserting for $\dot{\boldsymbol{\nu}}$, where $\dot{\boldsymbol{x}}$ is the operational space dynamics described in (3.14). However, since the weighted pseudoinverse contains uncertainties, the operational space and null space dynamics are no longer completely decoupled. As a result, the operational space dynamics with a subtask κ is

$$\begin{aligned}\boldsymbol{M}_x \ddot{\boldsymbol{x}} + \bar{\boldsymbol{J}}_x^T \left(\boldsymbol{C}_q \boldsymbol{\zeta}_{b/i}^b + \boldsymbol{D} \boldsymbol{\zeta}_{b/i}^b + \boldsymbol{g} \right) &= \boldsymbol{f}_x + \bar{\boldsymbol{J}}_x^T \left(\boldsymbol{I}_n - \boldsymbol{J}_x^T \hat{\bar{\boldsymbol{J}}}_x^T \right) \boldsymbol{\tau}_\kappa && \text{(B.17a)} \\ &= \boldsymbol{f}_x - \tilde{\bar{\boldsymbol{J}}}_x^T \boldsymbol{\tau}_\kappa && \text{(B.17b)} \end{aligned}$$

where $\boldsymbol{D}$ and $\boldsymbol{g}$ is the hydrodynamic damping matrix and hydrostatic wrench respectively as given in (2.9b). $\boldsymbol{C}_q$ is defined as

$$\boldsymbol{C}_q = \boldsymbol{C} - \boldsymbol{M} \bar{\boldsymbol{J}}_x \frac{d}{dt}\left(\boldsymbol{J}_x\right) \tag{B.18}$$

The adaptive backstepping control method from [18] requires a different definition of the operational space dynamics to ensure som cancellations in the computation of the Lyapunov candidate derivative. This is why $\boldsymbol{c}_x$, $\boldsymbol{d}_x$ and $\boldsymbol{g}_x$ from (3.13) is not used here.

Inserting for $\dot{\boldsymbol{\nu}}$ in (B.16) yields

$$\begin{aligned}\dot{V}_2 = &\boldsymbol{\nu}^T d \left(\boldsymbol{f}_x - \tilde{\bar{\boldsymbol{J}}}_x^T \boldsymbol{\tau}_\kappa - \boldsymbol{c}_x - \boldsymbol{d}_x - \boldsymbol{g}_x - \boldsymbol{M}_x \dot{\boldsymbol{\alpha}} \right) - \boldsymbol{z}^T \boldsymbol{K}_z \boldsymbol{z} + \boldsymbol{z}^T \boldsymbol{\nu} + \frac{\dot{d}}{2} \boldsymbol{\nu}^T \boldsymbol{M}_x \boldsymbol{\nu} \\ &+ \frac{d}{2} \boldsymbol{\nu}^T \frac{d}{dt}\left(\boldsymbol{M}_x\right) \boldsymbol{\nu} + \frac{1}{\gamma} \tilde{\boldsymbol{\beta}} \dot{\tilde{\boldsymbol{\beta}}} + \tilde{\boldsymbol{\phi}}_d^T \boldsymbol{\Gamma}_d^{-1} \dot{\tilde{\boldsymbol{\phi}}}_d + \tilde{\boldsymbol{\phi}}_n^T \boldsymbol{\Gamma}_n^{-1} \dot{\tilde{\boldsymbol{\phi}}}_n + \tilde{\boldsymbol{\phi}}_g^T \boldsymbol{\Gamma}_g^{-1} \dot{\tilde{\boldsymbol{\phi}}}_g \end{aligned} \tag{B.19}$$

From (B.3), (B.4) and (3.13a), which can be rewritten as

$$\boldsymbol{M}_x = \bar{\boldsymbol{J}}_x^T \boldsymbol{M} \bar{\boldsymbol{J}}_x \tag{B.20}$$

equation (B.19) can be written as

$$\dot{V}_2 = \boldsymbol{\nu}^T \left(d \left(\boldsymbol{f}_x - \hat{\bar{\boldsymbol{J}}}_x^T \boldsymbol{\tau}_\kappa + \frac{1}{2} \frac{d}{dt} \left(\boldsymbol{M}_x \right) \boldsymbol{\nu} \right) + \bar{\boldsymbol{J}}_n^T \left(\boldsymbol{\tau}_\kappa - \boldsymbol{C}_q \boldsymbol{\zeta}_{b/i}^b - \boldsymbol{D} \boldsymbol{\zeta}_{b/i}^b - \boldsymbol{g} - \boldsymbol{M} \bar{\boldsymbol{J}}_x \dot{\boldsymbol{\alpha}} \right) \right.$$
$$\left. + \frac{\dot{d}}{2} \boldsymbol{M}_x \boldsymbol{\nu} + \boldsymbol{z} \right) - \boldsymbol{z}^T \boldsymbol{K}_z \boldsymbol{z} + \frac{1}{\gamma} \tilde{\boldsymbol{\beta}} \dot{\tilde{\boldsymbol{\beta}}} + \tilde{\boldsymbol{\phi}}_d^T \boldsymbol{\Gamma}_d^{-1} \dot{\tilde{\boldsymbol{\phi}}}_d + \tilde{\boldsymbol{\phi}}_n^T \boldsymbol{\Gamma}_n^{-1} \dot{\tilde{\boldsymbol{\phi}}}_n + \tilde{\boldsymbol{\phi}}_g^T \boldsymbol{\Gamma}_g^{-1} \dot{\tilde{\boldsymbol{\phi}}}_g \tag{B.21}$$

Now, the force $\boldsymbol{f}_x$ can be designed as

$$\boldsymbol{f}_x = \hat{\bar{\boldsymbol{J}}}_x^T \boldsymbol{\tau}_\kappa + \boldsymbol{f}_u \tag{B.22}$$

where

$$\hat{\bar{\boldsymbol{J}}}_x^T = \frac{\boldsymbol{\psi}_n \hat{\boldsymbol{\phi}}_d}{\boldsymbol{\psi}_d \hat{\boldsymbol{\phi}}_d} \tag{B.23}$$

By defining

$$\boldsymbol{Y} \left(\boldsymbol{q}, \boldsymbol{\zeta}_{n/i}^b, \dot{\boldsymbol{\alpha}}, \boldsymbol{\nu}, \boldsymbol{\phi} \right) = \bar{\boldsymbol{J}}_n^T \left(\boldsymbol{C}_q \boldsymbol{\zeta}_{b/i}^b + \boldsymbol{D} \boldsymbol{\zeta}_{b/i}^b + \boldsymbol{M} \bar{\boldsymbol{J}}_x \dot{\boldsymbol{\alpha}} \right) - \frac{1}{2} \left(d \frac{d}{dt} \left(\boldsymbol{M}_x \right) + \dot{d} \boldsymbol{M}_x \right) \boldsymbol{\nu} \tag{B.24}$$

the Lyapunov derivative becomes

$$\dot{V}_2 = \boldsymbol{\nu}^T \left(d \boldsymbol{f}_u + \bar{\boldsymbol{J}}_n^T \left(\boldsymbol{\tau}_\kappa - \boldsymbol{g} \right) - \boldsymbol{Y} \left(\boldsymbol{q}, \boldsymbol{\zeta}_{n/i}^b, \dot{\boldsymbol{\alpha}}, \boldsymbol{\nu}, \boldsymbol{\phi} \right) + \boldsymbol{z} \right)$$
$$- \boldsymbol{z}^T \boldsymbol{K}_z \boldsymbol{z} + \frac{1}{\gamma} \tilde{\boldsymbol{\beta}} \dot{\tilde{\boldsymbol{\beta}}} + \tilde{\boldsymbol{\phi}}_d^T \boldsymbol{\Gamma}_d^{-1} \dot{\tilde{\boldsymbol{\phi}}}_d + \tilde{\boldsymbol{\phi}}_n^T \boldsymbol{\Gamma}_n^{-1} \dot{\tilde{\boldsymbol{\phi}}}_n + \tilde{\boldsymbol{\phi}}_g^T \boldsymbol{\Gamma}_g^{-1} \dot{\tilde{\boldsymbol{\phi}}}_g \tag{B.25}$$

From [18]

$$\bar{\boldsymbol{J}}_n^T \left(\boldsymbol{\tau}_\kappa - \boldsymbol{g} \right) = \boldsymbol{\psi}_n \left(\boldsymbol{q}, \boldsymbol{\tau}_\kappa \right) \boldsymbol{\phi}_n + \boldsymbol{\psi}_g (\boldsymbol{q}) \boldsymbol{\phi}_g \tag{B.26}$$

According to [40] and the fact that $\boldsymbol{g}$ is linearly parametrized [18], equation (B.26) is linearly parametrized in $\boldsymbol{\phi}_n$. This means the parameters $\boldsymbol{\phi}_n$ and $\boldsymbol{\phi}_g$ can be adaptively estimated.

The matrices in (B.24) are still nonlinearly parametrized, and these are treated as time-varying uncertainties which can be dominated. The matrix $\bar{\boldsymbol{J}}_n^T \boldsymbol{M} \bar{\boldsymbol{J}}$ is

bounded $\forall \boldsymbol{q}$. The remaining elements are linearly dependent on $\boldsymbol{\zeta}_{b/i}^{b}$, but this can be factored into the regressor [18]. By defining

$$\boldsymbol{P} := d\frac{d}{dt}\left(\boldsymbol{M}_x\right) + \dot{d}\boldsymbol{M}_x \tag{B.27a}$$

$$\boldsymbol{R} := \bar{\boldsymbol{J}}_n^T \boldsymbol{C}_q \tag{B.27b}$$

$$\boldsymbol{S} := \bar{\boldsymbol{J}}_n^T \boldsymbol{D} \tag{B.27c}$$

where

$$\boldsymbol{p}_{ij} \in \mathbb{R}^{n+5} \quad \boldsymbol{r}_{ij} \in \mathbb{R}^{n+5} \quad \boldsymbol{s}_{ij} \in \mathbb{R}^{n+5} \tag{B.28}$$

are defined such that

$$\left(\boldsymbol{\zeta}_{b/i}^{b}\right)^T \boldsymbol{p}_{ij} = \boldsymbol{P}_{ij} \tag{B.29a}$$

$$\left(\boldsymbol{\zeta}_{b/i}^{b}\right)^T \boldsymbol{r}_{ij} = \boldsymbol{R}_{ij} \tag{B.29b}$$

$$\left(\boldsymbol{\zeta}_{b/i}^{b}\right)^T \boldsymbol{s}_{ij} = \boldsymbol{S}_{ij} \tag{B.29c}$$

The time-varying parameters can be factored into the regressor. By writing

$$-\left(d\frac{d}{dt}\left(\boldsymbol{M}_x\right) + \dot{d}\boldsymbol{M}_x\right)\frac{\boldsymbol{\nu}}{2} = \boldsymbol{\psi}_P\left(\boldsymbol{\nu}, \boldsymbol{\zeta}_{b/i}^{b}\right)\boldsymbol{\theta}_P(t) \tag{B.30a}$$

$$\bar{\boldsymbol{J}}_n^T \boldsymbol{C}_q \boldsymbol{\zeta}_{b/i}^{b} = \boldsymbol{\psi}_R\left(\boldsymbol{\zeta}_{b/i}^{b}\right)\boldsymbol{\theta}_R(t) \tag{B.30b}$$

$$\bar{\boldsymbol{J}}_n^T \boldsymbol{D} \boldsymbol{\zeta}_{b/i}^{b} = \boldsymbol{\psi}_S\left(\boldsymbol{\zeta}_{b/i}^{b}\right)\boldsymbol{\theta}_S(t) \tag{B.30c}$$

$$\bar{\boldsymbol{J}}_n^T \boldsymbol{M} \bar{\boldsymbol{J}}_x \dot{\boldsymbol{\alpha}} = \boldsymbol{\psi}_Q\left(\dot{\boldsymbol{\alpha}}\right)\boldsymbol{\theta}_Q(t) \tag{B.30d}$$

The time varying parameters $\boldsymbol{\theta}_i$ are defined as

$$\boldsymbol{\theta}_P(t) = \begin{bmatrix} \boldsymbol{\theta}_{P_1}^T & \cdots & \boldsymbol{\theta}_{P_{m_x}}^T \end{bmatrix}^T \tag{B.31a}$$

$$\boldsymbol{\theta}_{P_i}(t) = \begin{bmatrix} \boldsymbol{p}_{i1}^T & \cdots & \boldsymbol{p}_{im_x}^T \end{bmatrix}^T, \quad i = 1, ..., m_x \tag{B.31b}$$

$$\boldsymbol{\theta}_R(t) = \begin{bmatrix} \boldsymbol{\theta}_{R_1}^T & \cdots & \boldsymbol{\theta}_{R_{m_x}}^T \end{bmatrix}^T \tag{B.31c}$$

$$\boldsymbol{\theta}_{R_i}(t) = \begin{bmatrix} \boldsymbol{r}_{i1}^T & \cdots & \boldsymbol{r}_{im_x}^T \end{bmatrix}^T, \quad i = 1, ..., m_x \tag{B.31d}$$

$$\boldsymbol{\theta}_S(t) = \begin{bmatrix} \boldsymbol{\theta}_{S_1}^T & \cdots & \boldsymbol{\theta}_{S_{m_x}}^T \end{bmatrix}^T \tag{B.31e}$$

$$\boldsymbol{\theta}_{S_i}(t) = \begin{bmatrix} \boldsymbol{s}_{i1}^T & \cdots & \boldsymbol{s}_{im_x}^T \end{bmatrix}^T, \quad i = 1, ..., m_x \tag{B.31f}$$

$$\boldsymbol{\theta}_Q(t) = \begin{bmatrix} \boldsymbol{\theta}_{Q_1}^T & \cdots & \boldsymbol{\theta}_{Q_{m_x}}^T \end{bmatrix}^T \tag{B.31g}$$

$$\boldsymbol{\theta}_{Q_i}(t) = d \begin{bmatrix} \boldsymbol{M}_{x_{ii}} & \cdots & \boldsymbol{M}_{x_{im_x}} \end{bmatrix}^T, \quad i = 1, ..., m_x \tag{B.31h}$$

The regressors in (B.30) are defined in [18], omitting the frame specifications in $\boldsymbol{\zeta}$ (these are all $\boldsymbol{\zeta}_{b/i}^b$ in any case), the regressors are

$$\boldsymbol{\psi}_P(\boldsymbol{\nu}, \boldsymbol{\zeta}) = \begin{bmatrix} (\boldsymbol{\nu} \otimes \boldsymbol{\zeta})^T & \cdots & 0 \\ \vdots & \ddots & \vdots \\ 0 & \cdots & (\boldsymbol{\nu} \otimes \boldsymbol{\zeta})^T \end{bmatrix} \tag{B.32a}$$

$$\boldsymbol{\psi}_R(\boldsymbol{\zeta}) = \begin{bmatrix} [\boldsymbol{\zeta}^2] & [\boldsymbol{\zeta}\boldsymbol{\zeta}]] & \cdots & 0 \\ \vdots & & \ddots & \vdots \\ 0 & & \cdots & [[\boldsymbol{\zeta}^2] \quad [\boldsymbol{\zeta}\boldsymbol{\zeta}]] \end{bmatrix} = \boldsymbol{\psi}_S(\boldsymbol{\zeta}) \tag{B.32b}$$

$$[\boldsymbol{\zeta}^2] = \begin{bmatrix} \boldsymbol{\zeta}_1^2 & \boldsymbol{\zeta}_2^2 & \cdots & \boldsymbol{\zeta}_n^2 \end{bmatrix} \tag{B.32c}$$

$$[\boldsymbol{\zeta}\boldsymbol{\zeta}] = \begin{bmatrix} \boldsymbol{\zeta}_1\boldsymbol{\zeta}_2 & \boldsymbol{\zeta}_1\boldsymbol{\zeta}_3 & \cdots & \boldsymbol{\zeta}_{n-1}\boldsymbol{\zeta}_n \end{bmatrix} \tag{B.32d}$$

$$\boldsymbol{\psi}_Q(\dot{\boldsymbol{\alpha}}) = \begin{bmatrix} \dot{\boldsymbol{\alpha}}^T & 0 & \cdots & \cdots & 0 \\ \dot{\boldsymbol{\alpha}}_1 ({}^m e_2) & \dot{\boldsymbol{\alpha}}_{2:m}^T & \cdots & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots & \vdots \\ \dot{\boldsymbol{\alpha}}_1 ({}^m e_{m-1}) & \dot{\boldsymbol{\alpha}}_1 ({}^{m-1} e_{m-2}) & \cdots & \dot{\boldsymbol{\alpha}}_{m-1:m}^T & 0 \\ \dot{\boldsymbol{\alpha}}_1 ({}^m e_m) & \dot{\boldsymbol{\alpha}}_1 ({}^{m-1} e_{m-1}) & \cdots & \dot{\boldsymbol{\alpha}}_{m-1} ({}^2 e_2) & \dot{\boldsymbol{\alpha}}_m \end{bmatrix} \tag{B.32e}$$

$${}^k e_i = \begin{bmatrix} p_1 & \cdots & p_j & \cdots & p_k \end{bmatrix}$$

$$p_j = \begin{cases} 1, & \text{if } j = i \\ 0, & \text{otherwise} \end{cases} \quad j = 1, 2, \cdots, k \tag{B.32f}$$

Now, the nonlinearly parametrized term $\boldsymbol{Y}$ in (B.24) can be linearly parametrized as

$$\boldsymbol{Y} = \boldsymbol{\psi}\left(\boldsymbol{\zeta}_{b/i}^{b}, \boldsymbol{\nu}, \dot{\boldsymbol{\alpha}}\right)\boldsymbol{\theta}\left(t\right) \tag{B.33}$$

where

$$\boldsymbol{\psi} = \begin{bmatrix}\boldsymbol{\psi}_P & \boldsymbol{\psi}_R & \boldsymbol{\psi}_S & \boldsymbol{\psi}_Q\end{bmatrix} \tag{B.34a}$$

$$\boldsymbol{\theta}(t) = \begin{bmatrix}\boldsymbol{\theta}_P^T(t) & \boldsymbol{\theta}_R^T(t) & \boldsymbol{\theta}_S^T(t) & \boldsymbol{\theta}_Q^T(t)\end{bmatrix}^T \tag{B.34b}$$

The time-varying terms are bounded below and above as $\boldsymbol{\theta}_i \in [\underline{\boldsymbol{\theta}}_i, \bar{\boldsymbol{\theta}}_i] \forall t \geq 0, i = P, R, S, Q$. The bound is defined as $\boldsymbol{\beta}$, which has been present in the Lyapunov function previously, and is defined as [18]

$$\boldsymbol{\beta} := \left(\sum_{i=1}^{L} \max\left(\underline{\boldsymbol{\theta}}_i^2, \bar{\boldsymbol{\theta}}_i^2\right)\right)^{\frac{1}{2}} \tag{B.35}$$

Having found linear parameterizations in eqs. (B.5a), (B.26), (B.30) and (B.33), the derivations from (B.25) may continue. Remembering that $\tilde{\boldsymbol{\phi}}_d = \hat{\boldsymbol{\phi}}_d - \boldsymbol{\phi}$, (B.25) becomes

$$\begin{aligned}\dot{V}_2 = \boldsymbol{\nu}^T &\left(-\boldsymbol{f}_u\boldsymbol{\psi}_d\tilde{\boldsymbol{\phi}}_d + \boldsymbol{f}_u\boldsymbol{\psi}_d\hat{\boldsymbol{\phi}}_d + \boldsymbol{\psi}_n\boldsymbol{\phi}_n + \boldsymbol{\psi}_g\boldsymbol{\phi}_g - \boldsymbol{\psi}\boldsymbol{\theta}(t) + \boldsymbol{z}\right) \\ &- \boldsymbol{z}^T\boldsymbol{K}_z\boldsymbol{z} + \frac{1}{\gamma}\tilde{\boldsymbol{\beta}}\dot{\tilde{\boldsymbol{\beta}}} + \tilde{\boldsymbol{\phi}}_d^T\boldsymbol{\Gamma}_d^{-1}\dot{\hat{\boldsymbol{\phi}}}_d + \tilde{\boldsymbol{\phi}}_n^T\boldsymbol{\Gamma}_n^{-1}\dot{\hat{\boldsymbol{\phi}}}_n + \tilde{\boldsymbol{\phi}}_g^T\boldsymbol{\Gamma}_g^{-1}\dot{\hat{\boldsymbol{\phi}}}_g\end{aligned} \tag{B.36}$$

Using the bound on $\boldsymbol{\theta}$ in (B.35), equation (B.36) becomes

$$\begin{aligned}\dot{V}_2 \leq \boldsymbol{\nu}^T &\left(\boldsymbol{f}_u\boldsymbol{\psi}_d\hat{\boldsymbol{\phi}}_d + \boldsymbol{\psi}_n\boldsymbol{\phi}_n + \boldsymbol{\psi}_g\boldsymbol{\phi}_g + \boldsymbol{z}\right) + \left\|\boldsymbol{\nu}^T\boldsymbol{\psi}\right\|\boldsymbol{\beta} - \boldsymbol{z}^T\boldsymbol{K}_z\boldsymbol{z} \\ &+ \frac{1}{\gamma}\tilde{\boldsymbol{\beta}}\dot{\tilde{\boldsymbol{\beta}}} + \tilde{\boldsymbol{\phi}}_d^T\left(\boldsymbol{\Gamma}_d^{-1}\dot{\hat{\boldsymbol{\phi}}}_d - \boldsymbol{\psi}_d^T\boldsymbol{\nu}^T\boldsymbol{f}_u\right) + \tilde{\boldsymbol{\phi}}_n^T\boldsymbol{\Gamma}_n^{-1}\dot{\hat{\boldsymbol{\phi}}}_n + \tilde{\boldsymbol{\phi}}_g^T\boldsymbol{\Gamma}_g^{-1}\dot{\hat{\boldsymbol{\phi}}}_g\end{aligned} \tag{B.37}$$

Now, $\boldsymbol{f}_u$ can be designed as

$$\boldsymbol{f}_u = \frac{1}{\boldsymbol{\psi}_d\hat{\boldsymbol{\phi}}_d}\left(-\boldsymbol{K}_\nu\boldsymbol{\nu} - \boldsymbol{z} - \boldsymbol{\psi}_n\hat{\boldsymbol{\phi}}_n - \boldsymbol{\psi}_g\hat{\boldsymbol{\phi}}_g - \frac{\boldsymbol{\psi}\boldsymbol{\psi}^T\boldsymbol{\nu}\hat{\boldsymbol{\beta}}^2}{\left\|\boldsymbol{\nu}^T\boldsymbol{\psi}\right\|\hat{\boldsymbol{\beta}} + \epsilon\left\|\boldsymbol{\nu}\right\|^2}\right) \tag{B.38}$$

where $\epsilon < \lambda_{min}(\boldsymbol{K}_v)$. Inserting the control law into the Lyapunov function derivative yields

$$\dot{V}_2 \leq \left\|\boldsymbol{\nu}^T\boldsymbol{\psi}\right\|\boldsymbol{\beta} - \frac{\boldsymbol{\psi}\boldsymbol{\psi}^T\boldsymbol{\nu}\hat{\boldsymbol{\beta}}^2}{\left\|\boldsymbol{\nu}^T\boldsymbol{\psi}\right\|\hat{\boldsymbol{\beta}} + \epsilon\left\|\boldsymbol{\nu}\right\|^2} - \boldsymbol{z}^T\boldsymbol{K}_z\boldsymbol{z} - \boldsymbol{\nu}^T\boldsymbol{K}_\nu\boldsymbol{\nu} + \frac{1}{\gamma}\tilde{\boldsymbol{\beta}}\dot{\hat{\boldsymbol{\beta}}}$$
$$+ \tilde{\boldsymbol{\phi}}_d^T\left(\boldsymbol{\Gamma}_d^{-1}\dot{\hat{\boldsymbol{\phi}}}_d - \boldsymbol{\psi}_d^T\boldsymbol{\nu}^T\boldsymbol{f}_u\right) + \tilde{\boldsymbol{\phi}}_n^T\left(\boldsymbol{\Gamma}_n^{-1}\dot{\hat{\boldsymbol{\phi}}}_n - \boldsymbol{\psi}_n^T\boldsymbol{\nu}\right) + \tilde{\boldsymbol{\phi}}_g^T\left(\boldsymbol{\Gamma}_g^{-1}\dot{\hat{\boldsymbol{\phi}}}_g - \boldsymbol{\psi}_g^T\boldsymbol{\nu}\right) \tag{B.39a}$$

$$\leq \frac{\left\|\boldsymbol{\nu}^T\boldsymbol{\psi}\right\|\hat{\boldsymbol{\beta}}\epsilon\left\|\boldsymbol{\nu}\right\|^2}{\left\|\boldsymbol{\nu}^T\boldsymbol{\psi}\right\|\hat{\boldsymbol{\beta}} + \epsilon\left\|\boldsymbol{\nu}\right\|^2} + \tilde{\boldsymbol{\beta}}\left(\frac{1}{\gamma}\dot{\hat{\boldsymbol{\beta}}} - \left\|\boldsymbol{\nu}^T\boldsymbol{\psi}\right\|\right) - \boldsymbol{z}^T\boldsymbol{K}_z\boldsymbol{z} - \boldsymbol{\nu}^T\boldsymbol{K}_\nu\boldsymbol{\nu}$$
$$+ \tilde{\boldsymbol{\phi}}_d^T\left(\boldsymbol{\Gamma}_d^{-1}\dot{\hat{\boldsymbol{\phi}}}_d - \boldsymbol{\psi}_d^T\boldsymbol{\nu}^T\boldsymbol{f}_u\right) + \tilde{\boldsymbol{\phi}}_n^T\left(\boldsymbol{\Gamma}_n^{-1}\dot{\hat{\boldsymbol{\phi}}}_n - \boldsymbol{\psi}_n^T\boldsymbol{\nu}\right) + \tilde{\boldsymbol{\phi}}_g^T\left(\boldsymbol{\Gamma}_g^{-1}\dot{\hat{\boldsymbol{\phi}}}_g - \boldsymbol{\psi}_g^T\boldsymbol{\nu}\right) \tag{B.39b}$$

It can be shown that [18]

$$\dot{V}_2 \leq \tilde{\boldsymbol{\beta}}\left(\frac{1}{\gamma}\dot{\hat{\boldsymbol{\beta}}} - \left\|\boldsymbol{\nu}^T\boldsymbol{\psi}\right\|\right) - \boldsymbol{z}^T\boldsymbol{K}_z\boldsymbol{z} - \boldsymbol{\nu}^T\left(\boldsymbol{K}_\nu - \epsilon\boldsymbol{I}_{m\times m}\right)\boldsymbol{\nu}$$
$$+ \tilde{\boldsymbol{\phi}}_d^T\left(\boldsymbol{\Gamma}_d^{-1}\dot{\hat{\boldsymbol{\phi}}}_d - \boldsymbol{\psi}_d^T\boldsymbol{\nu}^T\boldsymbol{f}_u\right) + \tilde{\boldsymbol{\phi}}_n^T\left(\boldsymbol{\Gamma}_n^{-1}\dot{\hat{\boldsymbol{\phi}}}_n - \boldsymbol{\psi}_n^T\boldsymbol{\nu}\right) + \tilde{\boldsymbol{\phi}}_g^T\left(\boldsymbol{\Gamma}_g^{-1}\dot{\hat{\boldsymbol{\phi}}}_g - \boldsymbol{\psi}_g^T\boldsymbol{\nu}\right) \tag{B.40}$$

Using (B.40), the adaptation laws can be designed as [18]

$$\dot{\hat{\boldsymbol{\beta}}} = \gamma\left\|\boldsymbol{\nu}^T\boldsymbol{\psi}\right\| \tag{B.41a}$$

$$\dot{\hat{\boldsymbol{\phi}}}_d = \begin{cases} \boldsymbol{\Gamma}_d\boldsymbol{\psi}_d^T\boldsymbol{\nu}^T\boldsymbol{f}_u, & \text{if } \hat{\boldsymbol{\phi}}_d \in int(\Phi) \\ & \text{or if } \hat{\boldsymbol{\phi}}_d \in \partial\Phi \\ & \text{and } \boldsymbol{\psi}_d\boldsymbol{\Gamma}_1\boldsymbol{\psi}_d^T\boldsymbol{\nu}^T\boldsymbol{f}_u \geq 0 \\ 0, & \text{otherwise} \end{cases} \tag{B.41b}$$

$$\dot{\hat{\boldsymbol{\phi}}}_g = \boldsymbol{\Gamma}_g\boldsymbol{\psi}_g^T\boldsymbol{\nu} \tag{B.41c}$$

$$\dot{\hat{\boldsymbol{\phi}}}_n = \boldsymbol{\Gamma}_n\boldsymbol{\psi}_n^T\boldsymbol{\nu} \tag{B.41d}$$

$int(\Phi)$ and $\partial\Phi$ is the interior and boundary of the set defined as

$$\Phi := \bigcap_{\boldsymbol{\psi}_d\in\Omega_d}\left\{\hat{\boldsymbol{\phi}}_d \in \mathbb{R}^l : \boldsymbol{\psi}_d\hat{\boldsymbol{\phi}}_d > 0\right\} \tag{B.42}$$

As a result, the Lyapunov function candidate derivative satisfies

$$\dot{V}_2 \leq -\boldsymbol{z}^T \boldsymbol{K}_z \boldsymbol{z} - \boldsymbol{\nu}^T \left(\boldsymbol{K}_\nu - \epsilon \boldsymbol{I}_{m \times m} \right) \boldsymbol{\nu} \tag{B.43}$$

which is negative semidefinite as $\epsilon < \lambda_{min} \left(\boldsymbol{K}_v \right)$. In [18], the control law is shown to be uniformly asymptotically stable.

B.2 Simulink model

The controller equations given in eqs. (B.38) and (B.41) have been implemented in the Simulink model as a fourth controller. The implementation currently only covers the end-effector task, but can easily be extended to function with joint angles and the actuation index using the results in sections 6.2 and 6.3 once the end-effector control is working.

B.3 Challenges

B.3.1 Projection algorithm

The projection algorithm in (B.41b) should be implemented according to the set in (B.42). The implementation was made according to the set definition, but [18] has implemented the projection as functions of the joint angles. Determining how this translates to the AIAUV is not necessarily simple, but an idea could be to apply the all n_q joint angles in the same way as the paper.

B.3.2 Linear parametrization and regressor choice

Some of the parameters that must be chosen as part of the linear parametrization described in chapter B are not explicitly defined in [18]. The parameters are $\boldsymbol{\psi}_d$, $\boldsymbol{\psi}_n$, $\boldsymbol{\psi}_g$. The dimensions can to some extent be deduced from the example in [18], which for the AIAUV implies

$$\boldsymbol{\psi}_d(\boldsymbol{q}) \in \mathbb{R}^{1 \times n_q} \tag{B.44a}$$

$$\boldsymbol{\psi}_n(\boldsymbol{q}) \in \mathbb{R}^{6 \times n_q} \tag{B.44b}$$

$$\boldsymbol{\psi}_g(\boldsymbol{q}) \in \mathbb{R}^{6 \times n_q} \tag{B.44c}$$

where it should be noted that all equations using these parameters cancel out the dimension, such that the dimension is not directly implied. Furthermore,

the dimensions of the adaptive terms

$$\boldsymbol{\phi}_d \in \mathbb{R}^{n_q \times 1} \tag{B.45a}$$
$$\boldsymbol{\phi}_n \in \mathbb{R}^{n_q \times 1} \tag{B.45b}$$
$$\boldsymbol{\phi}_g \in \mathbb{R}^{n_q \times 1} \tag{B.45c}$$

are implied by the dimensions in (B.44).

In the example in [18], $\boldsymbol{\psi}_d$ has been chosen from (B.5a). It is dependent on the joint angles $\boldsymbol{q}$ and deciding how to choose this vector is a challenge. Using the physical parameters given in the example, an attempt was made to simplify the dynamic model [41] used in the paper to see if there was an indication to how $\boldsymbol{\psi}_d$ was chosen, but with no success. Choosing $\boldsymbol{\psi}_d$ directly from (B.5a) was also an option, but d is a scalar such that the equation becomes one equation with n_q unknowns.

The regressor matrices $\boldsymbol{\psi}_n$ and $\boldsymbol{\psi}_g$ are also configuration dependent and should be set prior to the simulation. They are not specified in the example, and finding the correct values for these matrices is difficult.

This appendix includes derivations for the derivative $\frac{d}{dt}\left(\boldsymbol{M}_x\right)$ and $\dot{d}$, where the first is the task specific operational space matrix and the latter is a scalar defined in [18]. The derivatives were found, but ultimately not necessary for the direct implementation of the adaptive backstepping scheme. However, as the method does not work yet, they are provided in the appendix for future use if required.

B.4 Task rigid body mass matrix derivative

The task rigid body mass matrix is given in (3.13a) as

$$\boldsymbol{M}_x = \left(\boldsymbol{J}_x\left(\boldsymbol{q}(t)\right)\boldsymbol{M}^{-1}\left(\boldsymbol{q}(t)\right)\boldsymbol{J}_x^T\left(\boldsymbol{q}(t)\right)\right)^{-1} \tag{B.46}$$

where the dependencies on $\boldsymbol{q}(t)$ has been included. These dependencies will be included below when required to illustrate the differentiation with the total derivative. Differentiating the inverse in (B.46) by (A.5d) yields

$$\frac{d}{dt}\left(\boldsymbol{M}_x\right) = -\left(\boldsymbol{J}_x\boldsymbol{M}^{-1}\boldsymbol{J}_x^T\right)^{-1}\frac{d}{dt}\left(\boldsymbol{J}_x\boldsymbol{M}^{-1}\boldsymbol{J}_x^T\right)\left(\boldsymbol{J}_x\boldsymbol{M}^{-1}\boldsymbol{J}_x^T\right)^{-1} \tag{B.47}$$

Applying the product rule (A.5a) results in

$$\frac{d}{dt}\left(\boldsymbol{M}_x\right) = -\boldsymbol{M}_x\left(\frac{d}{dt}\left(\boldsymbol{J}_x\left(\boldsymbol{q}(t)\right)\right)\boldsymbol{M}^{-1}\boldsymbol{J}_x^T + \boldsymbol{J}_x\frac{d}{dt}\left(\boldsymbol{M}^{-1}\left(\boldsymbol{q}(t)\right)\right)\boldsymbol{J}_x^T + \boldsymbol{J}_x\boldsymbol{M}^{-1}\left(\frac{d}{dt}\left(\boldsymbol{J}_x\left(\boldsymbol{q}(t)\right)\right)\right)^T\right)\boldsymbol{M}_x \tag{B.48}$$

The matrix differentiation required in (B.48) requires the total derivative given in (A.6). The two derivatives in (B.48) are

$$\frac{d}{dt}\left(\boldsymbol{J}_x\left(\boldsymbol{q}(t)\right)\right) = \underbrace{\frac{\partial \boldsymbol{J}_x}{\partial t}\frac{\partial t}{\partial t}}_{=0} + \frac{\partial \boldsymbol{J}_x}{\partial \boldsymbol{q}}\frac{\partial \boldsymbol{q}}{\partial t} = \frac{\partial \boldsymbol{J}_x}{\partial \boldsymbol{q}}\frac{\partial \boldsymbol{q}}{\partial t} \tag{B.49a}$$

$$\frac{d}{dt}\left(\boldsymbol{M}^{-1}\left(\boldsymbol{q}(t)\right)\right) = -\boldsymbol{M}^{-1}\frac{d}{dt}\left(\boldsymbol{M}\right)\boldsymbol{M}^{-1} \tag{B.49b}$$

Where $\frac{d}{dt}\left(\boldsymbol{J}_x\right)$ can be found in chapter 6 for each task and $\frac{d}{dt}\left(\boldsymbol{M}\right)$ is given in [31] as

$$\frac{d}{dt}\left(\boldsymbol{M}\right) = \boldsymbol{C} + \boldsymbol{C}^T \tag{B.50}$$

Finally, the task rigid body mass matrix derivative is

$$\frac{d}{dt}\left(\boldsymbol{M}_x\right) = -\boldsymbol{M}_x\left(\frac{d}{dt}\left(\boldsymbol{J}_x\right)\boldsymbol{M}^{-1}\boldsymbol{J}_x^T - \boldsymbol{J}_x\boldsymbol{M}^{-1}\frac{d}{dt}\left(\boldsymbol{M}\right)\boldsymbol{M}^{-1}\boldsymbol{J}_x^T + \boldsymbol{J}_x\boldsymbol{M}^{-1}\left(\frac{d}{dt}\left(\boldsymbol{J}_x\right)\right)^T\right)\boldsymbol{M}_x \tag{B.51}$$

B.5 Linear parametrization coefficient derivative

From (B.5a), the coefficient is

$$d = \left(\det\left(\boldsymbol{M}\right)\right)^{n-1}\det\left(\boldsymbol{J}_x\boldsymbol{J}_x^T\right) \tag{B.52}$$

Where the derivative is

$$\frac{d}{dt}\left(d\right) = \left(\det\left(\boldsymbol{M}\right)\right)^{n-1}\det\left(\boldsymbol{J}_x\boldsymbol{J}_x^T\right) \tag{B.53}$$

First, applying the product rule (A.5a) yields

$$\frac{d}{dt}(d) = \frac{d}{dt}\left((\det(\boldsymbol{M}))^{n-1}\right)\det\left(\boldsymbol{J}_x\boldsymbol{J}_x^T\right) + (\det(\boldsymbol{M}))^{n-1}\frac{d}{dt}\left(\det\left(\boldsymbol{J}_x\boldsymbol{J}_x^T\right)\right) \tag{B.54}$$

Performing the first part of the differentiation on the left side gives

$$\begin{aligned}\frac{d}{dt}(d) = (n-1)(\det(\boldsymbol{M}))^{n-2}\frac{d}{dt}(\det(\boldsymbol{M}))\det\left(\boldsymbol{J}_x\boldsymbol{J}_x^T\right)\\ + (\det(\boldsymbol{M}))^{n-1}\frac{d}{dt}\left(\det\left(\boldsymbol{J}_x\boldsymbol{J}_x^T\right)\right)\end{aligned} \tag{B.55}$$

The derivative of the determinant is given by (A.5e), which yields

$$\begin{aligned}\frac{d}{dt}(d) = (n-1)(\det(\boldsymbol{M}))^{n-2}\det(\boldsymbol{M})\operatorname{Tr}\left(\boldsymbol{M}^{-1}\frac{d}{dt}(\boldsymbol{M})\right)\det\left(\boldsymbol{J}_x\boldsymbol{J}_x^T\right)\\ + (\det(\boldsymbol{M}))^{n-1}\det\left(\boldsymbol{J}_x\boldsymbol{J}_x^T\right)\operatorname{Tr}\left(\left(\boldsymbol{J}_x\boldsymbol{J}_x^T\right)^{-1}\frac{d}{dt}\left(\boldsymbol{J}_x\boldsymbol{J}_x^T\right)\right)\end{aligned} \tag{B.56}$$

where $\frac{d}{dt}(\boldsymbol{M})$ is can be found through (B.50). The second derivative is found through the product rule

$$\frac{d}{dt}\left(\boldsymbol{J}_x\boldsymbol{J}_x^T\right) = \frac{d}{dt}(\boldsymbol{J}_x)\boldsymbol{J}_x^T + \boldsymbol{J}_x\left(\frac{d}{dt}(\boldsymbol{J}_x)\right)^T \tag{B.57}$$

which results in the full derivative as

$$\begin{aligned}\frac{d}{dt}(d) = (n-1)(\det(\boldsymbol{M}))^{n-1}\operatorname{Tr}\left(\boldsymbol{M}^{-1}\frac{d}{dt}(\boldsymbol{M})\right)\det\left(\boldsymbol{J}_x\boldsymbol{J}_x^T\right)\\ + (\det(\boldsymbol{M}))^{n-1}\det\left(\boldsymbol{J}_x\boldsymbol{J}_x^T\right)\operatorname{Tr}\left(\left(\boldsymbol{J}_x\boldsymbol{J}_x^T\right)^{-1}\left(\frac{d}{dt}(\boldsymbol{J}_x)\boldsymbol{J}_x^T\right.\right.\\ \left.\left. + \boldsymbol{J}_x\left(\frac{d}{dt}(\boldsymbol{J}_x)\right)^T\right)\right)\end{aligned} \tag{B.58}$$

Arnt Erik Stene

Robust Control for AIAUVs in the Operational Space

NTNU
Norwegian University of
Science and Technology