Oscar Nissen

Automating Tank Operations in Smolt Production - Control of an Underwater Manipulator

Master's thesis in Cybernetics and Robotics
Supervisor: Martin Føre
Co-supervisor: Herman Biørn Amundsen
June 2021

Oscar Nissen

Automating Tank Operations in Smolt Production - Control of an Underwater Manipulator

Master's thesis in Cybernetics and Robotics
Supervisor: Martin Føre
Co-supervisor: Herman Biørn Amundsen
June 2021

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

NTNU
Norwegian University of
Science and Technology

Sammendrag

Formålet med denne master oppgaven var å utlede en matematisk modell av en robotarm (manipulator) til bruk for automatisering av identifiserte operasjoner innen smolt produksjonsanlegg. Arbeidet er utført som en del av en masteroppgave ved NTNU Trondheim i samarbeid med SINTEF Ocean.

Et litteratursøk innen manipulator fluid dynamikk har blitt gjennomført. Basert på funnene i litteratursøket, ble en rekke metoder relatert til undervannsmanipulatorer og andre robotsystemer kombinert til en generalisert fremgangsmetode for robot armer til bruk under vann. Den matematiske modellen som beskriver manipulatorens dynamiske oppførsel under vann, ble deretter brukt i et Inverse Dynamics Controller system for å kontrollere posisjonen til robotarmens end-effector.

Inverse Dynamics Control systemet ble utviklet og simulert i Matlab Simulink ved bruk av en B-spline Trajectory Planner fra Robotic System Toolbox. Simuleringene ble utført i form av to casestudier relatert til typiske produksjonsoperasjoner i et smoltanlegg.

Denne oppgaven har lagt et godt grunnlag for videre utvikling og forskning innen den matematiske modelleringen og kontrollen av robotarm systemer innen smoltproduksjon gjennom å lage en metode for å definere og simulere manipulatorer i undervannsmiljøer ved bruk av Matlab og Simulink.

Abstract

This thesis aims at creating a mathematical model for a mobile robotic manipulator system to automate identified operations within smolt production facilities. The work has been done as part of a masters thesis at NTNU Trondheim in collaboration with SINTEF Ocean.

A literature review within the field of manipulator fluid dynamics has been conducted. Based on these findings, a set of methods from different studies and literature, related to underwater manipulators and vehicle-manipulator-systems, were combined and a generalized method for underwater robot manipulators was proposed. The mathematical model describing the manipulator underwater dynamics was then applied in an Inverse Dynamics Controller for end-effector position tracking. The input to the control system was formed by trajectory planner.

The inverse dynamics control system was created and simulated in Matlab Simulink using a B-spline trajectory planner provided by the Robotics System Toolbox. The simulations were performed in the form of two case studies related to smolt production operations.

This thesis has laid a good foundation for further development and research within the mathematical modelling and control of robotic manipulator systems within smolt production by creating a method for defining and simulating manipulators in underwater environments using Matlab and Simulink.

Preface

This master's thesis was written as a part of a 5-year master's degree within Cybernetics and Robotics at the Faculty of Information Technology and Electrical Engineering at Norwegian University of Science and Technology (NTNU). The thesis is a part of the Autosmolt 2025 project and was supervised by Martin Føre, and co-supervised by Herman Biørn Amundsen.

First and foremost, I would like to thank my supervisors for guidance and consistent encouragement throughout the writing process, and for giving me feedback and ideas after listening to my ramblings. I am also grateful for the support from Eleni Kelasidi and Bent Oddvar Arnesen Haugaløkken at SINTEF Ocean.

Personally, I have learned a lot from writing this thesis, both from a technical, industrial and structural perspective. First of all, I have learned a lot about smolt- and land-based production, both with regards to state-of-the-art operations, present challenges and possible solutions. From a technical perspective, I have learned how to describe, research, model and simulate robotic solutions with a specific purpose in mind. Finally, I have learned much about research methodology, and gained confidence within the field of robotics. I hope this thesis can be a step in the direction of increasing the level of autonomy in smolt production facilities.

Oscar Nissen
Trondheim, June 2021

Contents

List of Figures

List of Tables

Abbreviations

DoF	degree of freedom or degrees of freedom
DH	Denavit Hartenberg
pose	position and orientation
IDC	Inverse Dynamics Controller
ROV	Remotely Operated Underwater Vehicle
UUVs	Unmanned Underwater Vehicles
PFF	Precision Fish Farming

Chapter 1

Introduction

The aquaculture industry is predicted to be an influential contributor to keeping up with the ever-increasing toll on the world's food supplies [3]. In Norway, the aquaculture industry is dominated by Atlantic salmon production, which is currently the most significantly farmed finfish in marine aquaculture [12]. To accommodate the required increase in the production of salmon, technological advancements are needed. Increased automation of the day-to-day operations within aquaculture can contribute to an improved production level. Part of the solution in the day-to-day operations of fish farms is to increase the use of Unmanned Underwater Vehicles (UUVs) and other technological solutions. Adapting autonomous and robotic solutions is already an ongoing process. There is a general trend in the aquaculture industry of shifting production methods from manual operations and experienced-based reasoning towards a more objective approach using autonomous systems in different stages of salmon production [24]. This trend, going from experienced-based to knowledge-based reasoning to improve fish farming operations, is based on the concept of Precision Fish Farming (PFF) [14].

1.1 Smolt Production

Improving the aquaculture industry's production of Atlantic salmon through the concept of PFF and the use of new technology is not only for the fish farming operations out in the sea but also for the stages of land-based production. One particularly interesting stage of salmon production is the smolt phase. In this phase, the smoltification process occurs, a process in which the salmon changes appearance and adapts the ability to live in seawater (see Figure 1.1). Following the smoltification process in a land-base facility, the salmon has grown to the stage of post smolt, becoming ready for a life in one of the fish farms out at sea.

The smolt stage of salmon fish farming is decisive for its further growth and survival rate. Modern fish farming techniques accentuate the production of larger post-smolt because they are older and more robust when transferred from land-based facilities to sea cages. The reduced time spent at sea also means less exposure to sea lice. This parasite is currently the biggest concern for the Norwegian fish farming industry due to the treatment producing reduced growth and fish quality, as well as higher mortality [61]. More robust fish when transported to sea and less exposure to sea lice have resulted in a noticeable improvement in the average survival rate during the sea phase [44]. Although the longer land-based production results in increased production costs, based on current prices, it remains profitable compared to the risk of lower production

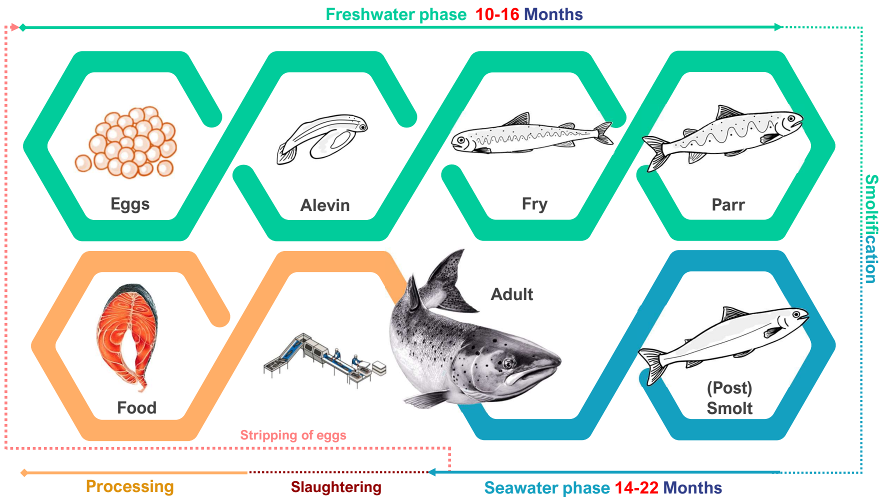


Figure 1.1: The life stages of the Atlantic salmon in captivity.

volumes due to sea lice. As a result, many new post-smolt facilities are now planned for construction [5]. Increasing the number of post-smolt facilities will eventually increase the burden on land-based facilities to be able to undertake the challenges of up-scaling production.

1.2 Autosmolt 2025 and Robot Manipulators

Today, smolt production plants are still based on the same principles and methods as the first generation of such plants established 40 years ago. Autosmolt 2025 is an ongoing project where the objective is to bring smolt production closer to the realization within the framework of Industry 4.0 by applying principles of PFF [54]. One of the research areas of the Autosmolt project is about creating a foundation for future fully unmanned smolt production sites. Smolt production is a biological factory process with different tasks that need to be finished daily. To complete the tasks and to optimize production, without human intervention, the introduction of robots will be necessary (see Figure 1.2.

Robot manipulators (robot arms) are considered to be suitable instruments for performing underwater operations and are typically equipped on UUVs [55]. These manipulators are composed of a sequence of rigid bodies (links) interconnected by joints with a suitable interchangeable tool attached to the end-effector. Underwater manipulators have been used for a variety of subsea tasks in different applications within offshore oil and gas, marine renewable energy and marine civil engineering industries as well as in marine science applications [48]. A common work scenario for an underwater robot on a science mission is to

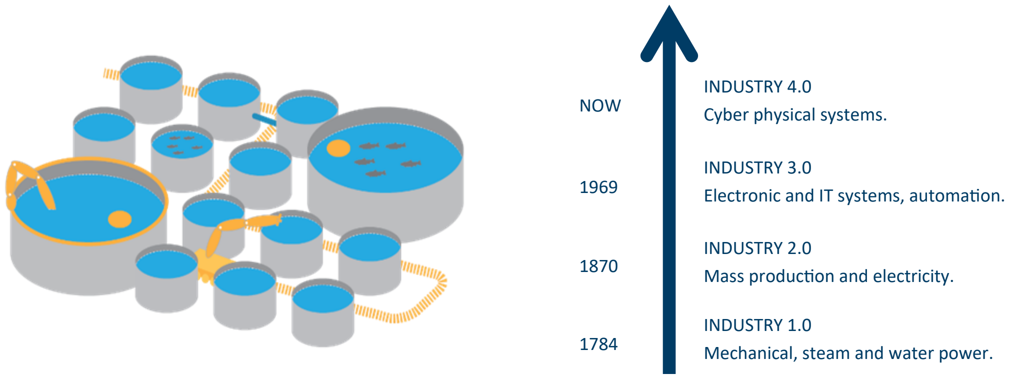

NOW
1969
1870
1784
INDUSTRY 4.0
Cyber physical systems.
INDUSTRY 3.0
Electronic and IT systems, automation.
INDUSTRY 2.0
Mass production and electricity.
INDUSTRY 1.0
Mechanical, steam and water power.

Figure 1.2: Autosmolt 2025: Foundation for the next generation of autonomous smolt production [54]

perform a task, such as picking up a rock or sampling a biological specimen from the water column, while holding the position and attitude of the vehicle fixed (stationary) [37]. These are similar operations to the day-to-day tasks required in a smolt tank.

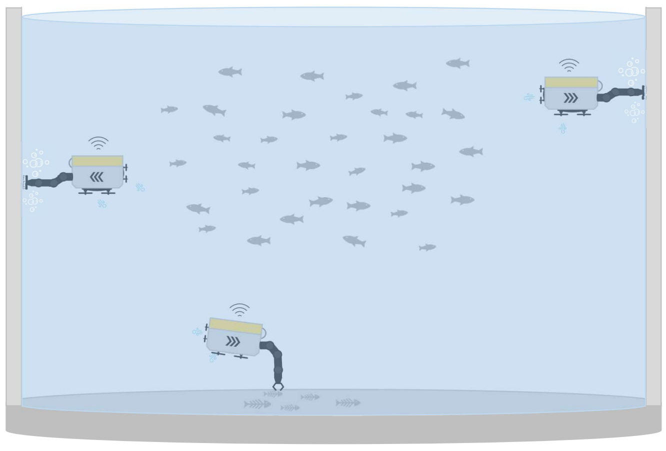


Figure 1.3: Day-to-day operations of robot manipulators in a smolt tank.

An illustration of how different day-to-day operations in a smolt tank would look like, is presented in Figure 1.3. Not only is the use of underwater robots practical in terms of repetitive tasks and the general operation of smolt production, but it might also be advantageous for securing fish welfare during daily operations. Studies have shown that stress levels is an important factor in fish welfare and mortality, in particular during transportation [20]. Cleaning a smolt tank currently requires transportation of the fish to another tank, and emptying the tank for manual cleaning. This is both time consuming and stressful for the fish. To avoid scaring or stressing the fish by using robots, it is however important that the robot is small in size and slow moving [27]. Although, there are no specific studies related to the presence of robots performing day-to-day operations in smolt tanks, UUVs with robot manipulators attached, might be favorable for fish welfare as well as for optimizing the day-to-day opeartions.

1.2.1 The Reach Bravo 7 Robot Manipulator

This master thesis will be a continuation of the authors' specialization project thesis (see Appendix B for the delivery files) related to a robot arm produced by Blueprint Lab, called Reach Bravo 7. The specialization project thesis was inspired by an earlier master thesis that explored the possibility for automated operations using a robot arm in smolt production facilities [56]. Selected operations for automation were cleaning of tanks, controlled feeding, and removal of dead fish and waste. To perform these operations Reach Bravo 7 was chosen on the basis of a comparative study.

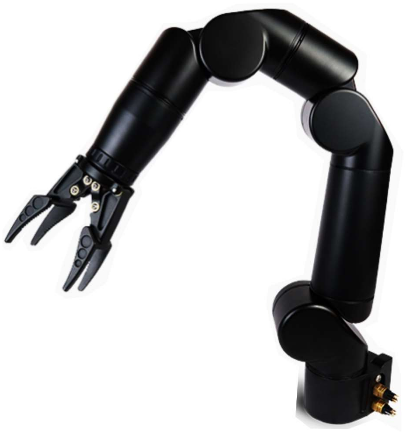


Figure 1.4: The Blueprint Lab Reach Bravo 7 robot manipulator [29]

1.3 Research Objectives

In order to add to the vision of the AUTOSMOLT 2025 project, this master thesis will be a continuation of the previously performed work on the Reach Bravo 7 robot manipulator. The central research question of the master thesis is defined as:

How do you go about creating a mathematical model of an underwater robotic manipulator to be controlled and virtually tested for automated smolt production?

Following the research question, a set of sub-tasks were determined:

1. **Perform literature study of manipulator fluid dynamics modelling.**

   Methods for implementing fluid dynamics on submerged robot manipulators should be investigated and evaluated in order to create an accurate model of the underwater effects.

2. **Create a mathematical model describing the underwater dynamics.**

   Based on the literature study and building on the land-based model developed in the specialization project, an expanded model for an underwater manipulator is to be made.

3. **Suggest and implement a suitable control system.**

   In order to control the manipulator position in an underwater environment, a suitable control system will be chosen and implemented in MATLAB while considering the nonlinearities of the manipulator and fluid dynamics.

4. **Suggest and implement methods for trajectory planning.**

   There are different methods for planning the trajectory of the robot manipulator. These will be explored, and a selected method is to be implemented in conjunction with the control system in MATLAB.

5. **Simulation experiments.**

   Simulations will be performed in MATLAB, verifying the manipulator model and evaluating the control system performance in a smolt tank through virtual experiments, conducting operations suitable for smolt production.

1.4 Outline

The rest of the thesis is organised as follows:

Part I
Part I includes the requisite theory to create a control framework for a robot manipulator in an underwater environment. Chapter 2 establishes some essential building blocks by covering the setup of coordinate frames, and some conventional symbolic representations typically used when developing mathematical models for robot manipulators. Building on the previous chapter, Chapter 3 presents solutions to the problems of forward and inverse kinematics by applying kinematic diagrams in combination with the Denavit Hartenberg Convention. Chapter Chapter 4 presents an approach to determine the dynamic equations of motion based on the Euler-Lagrange method. Then in chapter 5, a new framework for modelling hydrodynamic effects onto a robotic manipulator is proposed, and how this may be added to the original dynamical model. The combination of kinematics and dynamics qualifies the contingency to control the manipulator motion. Chapter 6 discusses possible control algorithms for position control to follow, or track, a desired trajectory. Also included in chapter 6, is a presentation of possible trajectory planning methods.

Part II
Part II involves a specific focus on the Reach Bravo 7 Manipulator by employing the theoretical denotations from part I. Chapter 7 contains the mathematical modeling of the Bravo 7 including its system description, forward and inverse kinematics solutions, and land-based and underwater dynamics model. Based on the mathematical model, a simulation setup is formed in Chapter 8 involving the use of Matlab and Simulink and a description of the setup of two case studies relevant for smolt production.

Part III
Part III presents and discusses the results from the simulations created in Part II. At the very end, a conclusion is put forth linked to the research objectives in Section 1.3

Part I

Theory

Chapter 2

How to Mathematically Describe Robot Manipulators

This chapter[1] covers the system setup and symbolic representation related to development of the robot manipulator mathematical models. Section 2.3 is especially important because it compiles and explains how robot manipulators will be described in the remaining chapters based on a set of coordinate frames.

2.1 Mechanical Structure of a Robot Manipulator

A robot manipulator is essentially a mechanical arm operating under computer control. The arm is composed of a sequence of rigid bodies (links) interconnected by joints to form a chain of links. The joints establish conjunctions in the chain that allows for relative motion of neighboring links. Generally, these joints are fitted with position sensors granting the possibility to measure the relative position of the adjacent links. In the case of rotary or *revolute joints*, these displacements are called joint angles. For sliding or prismatic joints, the relative displacements are referred to as *joint offsets*. More complicated joints exist, but they are typically represented as a combination of revolute and prismatic joints.

Figure 2.1 depicts how revolute and translation joints are illustrated in [57]. The links represent the spatial relationship between two joints and are displayed as simple lines between the joints. Although the chain of links makes up the majority of the robot arm, it is the *end-effector* that manipulates the surroundings and performs the desired tasks. The space in which the end-effector can affect its environment depends on the position of the manipulator *base*. Considering how the base can be moving, this brings an increased level of complexity when modelling the motion of the manipulator. Figure 2.2a shows an example of a manipulator consisting of a base, two revolute joints, one prismatic joint, and an end effector.

2.2 Symbolic Representation

Figure 2.2a shows a typical three degree of freedom or degrees of freedom (DoF) kinematic arrangement of a robot arm. This manipulator is defined as a spherical geometric type arm [57] due to its explicit combination of joint types. Figure 2.2b is a slightly altered version of 2.2a with a non-straight link referred to as

[1]This chapter is based on a similar chapter in the specialization project thesis. Chapter 2.4 is new.

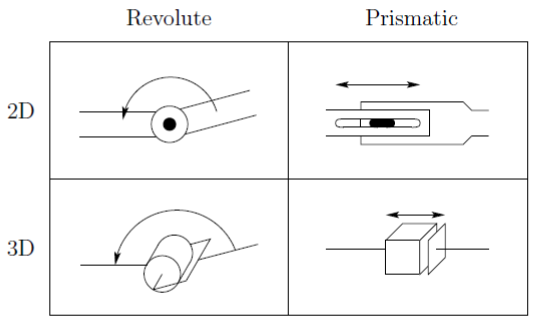
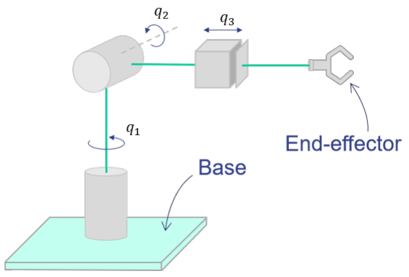
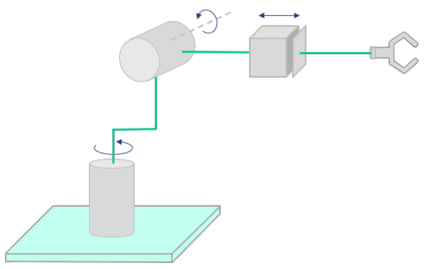


Figure 2.1: Symbolic illustrations of the different types of joints for a robot manipulator [57].



(a) Symbolic representation of spherical robot arm.



(b) Symbolic representation of a robot arm with a non-straight link.

Figure 2.2: Symbolic representation of two different spherical robot arms.

a shoulder offset. Having a shoulder offset tend to make the modelling slightly more complicated. Independent of its mechanical structure, the robot arm always have a base and an end-effector. A robot arm is always fixed to a base, but the base itself may be mobile. The end-effector is a gripper or a tool that may be used to influence the environment.

2.3 Coordinate Frames

To mathematically describe the configuration space and work space of the robot manipulator, coordinate frames are typically attached at the end of each link. The position of the coordinate frames may then be described in reference to a defined inertial frame (also called world frame). This way all objects, including the manipulator, may be referenced in relation to the inertial frame. By assuming the individual manipulator links are rigid and the base is fixed, it is possible to decide the position of any of the coordinate frames as a function of the joint variables. In the case of Figure 2.3 the inertial frame is established at the base of the robot arm and its origin is symbolically given by 2.1.

$$\{O_0\} = (x_o, y_o, z_o) \tag{2.1}$$

The position of the end-effector frame $\{O_3\}$ with respect to the inertial coordinate system is represented as $\boldsymbol{o}_3^0 = \begin{bmatrix} x_3^0 & y_3^0 & z_3^0 \end{bmatrix}^T$. Hence, for any manipulator with n joints, the position of the end-effector is given by 2.2.

$$\boldsymbol{o}_n^0 = \begin{bmatrix} x_n^0 & y_n^0 & z_n^0 \end{bmatrix}^T \tag{2.2}$$

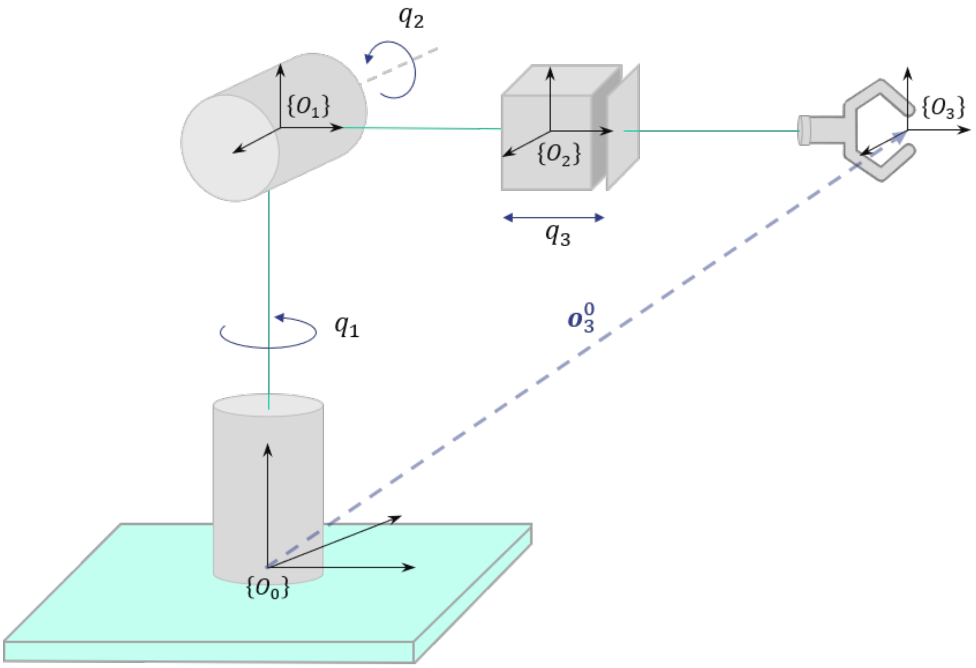


Figure 2.3: Symbolic representation of spherical robot arm with added coordinate frames.

2.4 Model Components

As previously described, the robot manipulator consists of multiple segments that need to be coordinated in terms of position and orientation and therefore requires advanced methods to form any type of analytical description. Consequently, this section includes a set of useful definitions and a description of the primary components that will be used to in the coming chapters develop the robot manipulator mathematical model.

2.4.1 Definitions

Joint variable:
Represents the relative displacement between adjacent links either in form of linear movement (prismatic join) or an angular rotation (revolute joint). The joint variable is denoted as $q_1 \cdots q_n$ for a n degrees of freedom robot manipulator.

Joint configuration:
A combination of a set of joint variable values resulting in a specific "shape" of the robot manipulator.

Configuration space:
The set of all possible joint configurations for a robot manipulator.

Workspace:
A set of points that can be reached by the end-effector. The workspace is constrained by the geometry of the manipulator as well as by the mechanical limitations of the joints.

2.4.2 Primary Components

Kinematics:
a geometrical description of the relationship between the joint position (workspace) and the joint configuration (configuration space).

Dynamics:
relationships between the torques applied to the *joints* and the consequent movements of the *links*

Control:
computation of the joint torques (control actions) necessary to execute a desired motion.

Trajectory Planning:
planning the desired movements of the manipulator.

Chapter 3

Kinematics

[1]Manipulator kinematics is the study of how joints connected to a chain of rigid bodies are transformed into viable kinematic motion without taking forces into account. The relationship is strictly geometric, and consequently, the forces and torques are ignored. The kinematic analysis aims to mathematically describe the bond between the joint variables and the position and orientation of different parts of the robot structure. This requires a systematic approach, especially when the number of joints and links increases. The systematic approach that describes the joints' cumulative effect throughout the kinematic chain is the foundation for solving the forward and inverse kinematics problems.

Section 3.1 in this chapter describes how to solve the forward kinematics using the Denavit-Hartenberg convention. Then Section 3.2 possible analytical and numerical solutions to the inverse kinematics problem, including a closer look at the spherical wrist manipulator configuration.

3.1 Forward Kinematics

The problem of forward kinematics for a robot manipulator seeks to mathematically formulate the position and orientation (pose) of the end-effector as a function of the joint variables. The method of solving forward kinematics is concise, flexible, and applicable regardless of the type and the number of joints in the robot arm. Calculating the forward kinematics is vital when getting familiar with a new robot manipulator. The solution can be found by following a systematic approach.

3.1.1 Kinematic Diagram

A good first step of any kinematic analysis is to draw a kinematic diagram similar to the symbolic representation introduced in Section 2.2. This type of schematic is helpful to aid the understanding of the robot manipulator system and to describe it mathematically. To avoid misconceptions about the labeling scheme, when explaining any of the approaches used here and in later chapters, a set of preliminary rules are defined below.

Preliminary Rules:

[1]The content in this chapter is a modified and more detailed version of the same chapter in the specialization project thesis.

For a manipulator with n joints numbered from 1 to n, there are $n+1$ links, where the links are numbered from 0 to n. The preliminary rules are demonstrated in Figure 3.1.

- Link 0 is equal to the base of the manipulator.
- Joint i connects link $i-1$ to i and moves them relative to each other.
- Link i extends joint i to joint $i+1$.
- There are at least one more frame than there are joints - one frame must be on the end-effector.
- All coordinate axes are drawn either up, down, right, left or in the first and third quadrant.

3.1.2 The Denavit-Hartenberg Convention

When the number of joints grows, the kinematic analysis becomes increasingly complex. The introduction of the Denavit Hartenberg (DH) convention simplifies the kinematic analysis by systematically describing the geometrical structure of a kinematic chain of links and joints. Central to the convention is the attachment of the coordinate frames. The definition of the frames requires consistency, but the frame placements are not unique [57][53]. Mathematically speaking, as long as frame i is rigidly attached to link i, there is considerable freedom when choosing the placement of the origin and the coordinate axes of the frame. The flexibility in terms of the coordinate axes has resulted in various variations of the notation. In this thesis the *distal variant* DH convention, described in [35], is practiced. By being consistent in regards to the defined variations, any of the DH conventions lead to simplifications of the kinematic modeling method.

Assigning the Coordinate Frames

There are rules that needs to be followed when assigning the frames based on the DH convention [57]:

- The Z axis must be the axis of rotation (revolute joint) or the direction of motion (prismatic joint).
- The X axis must be perpendicular to the Z axis of the frame before it.
- The X axis must intersect the Z axis of the frame before it.
- The Y axis must be drawn such that the frame follows the right-hand rule. Assigning the y-axis is rarely required when using the DH-convention.

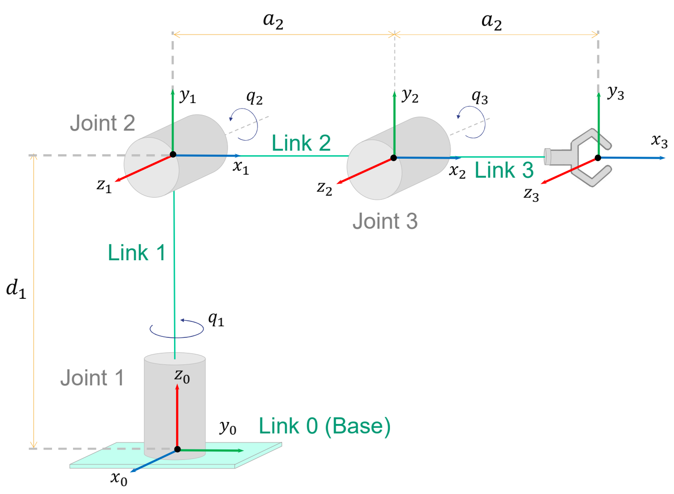


Figure 3.1: Kinematic diagram of an Elbow Manipulator with added coordinate frames according to the DH convention.

The DH convention and the preliminary rules are exemplified in Figure 3.1. Every link has its own coordinate frame which is connected to the far end of the link. There is always the same amount of frames as the number of links. The base is referred to as link 0.

Denavit-Hartenberg Parameters

Following the DH rules for assigning the coordinate frames, the description of the kinematic structure is broken down into four parameters as defined in Table 3.1. How the parameters relate to the kinematic diagram is depicted in Figure 3.1. Note that the definitions may be different for other variations of the DH convention.

Symbol	Definition	Type of value
θ_i	The angle between x_{i-1} and x_i measured about z_{i-1}.	Joint variable(q_i) if joint is revolute.
d_i	Distance along z_{i-1} from $\{O_{i-1}\}$ to the intersection of the x_i and z_{i-1} axes.	Joint variable(q_i) if joint is prismatic.
a_i	Distance along x_i from $\{O_i\}$ to the intersection of the x_i and z_{i-1} axes.	Constant
α_i	The angle between z_{i-1} and z_i measured about x_i .	Constant

Table 3.1: Explanation of the Denavit Hartenberg parameters.

Denavit-Hartenberg Table

The DH table is a summary of the DH parameter values identified for link 1 to n and is the result of the DH convention. The table is a compact overview of the manipulator's geometrical description and its possible movements. It is not uncommon to include the joint variables' physical constraints in the DH table.

The DH table for the Elbow Manipulator from Figure 3.1 is presented in Table 3.2. The table shows the manipulator configuration for each of the three links. An example of how the parameters are decided for each link is described below. The descriptions are very much conditioned on the definitions in Table 3.1.

Link 1:
Link 1 is affected by the rotary motion from joint 1, represented by the joint variable q_1. Joint 1 is not a prismatic joint, and therefore d_1 is a *constant* equal to the distance from O_0 to the intersection of the z_0 and x_0 axes in the direction of z_0. Since the intersection of the z_0 and x_1 axes is identical to the center of the second coordinate frame ($\{O_1\}$), a_1 is equal to 0. Lastly, to be able to rotate the first frame in such that z_0 becomes parallel to the z_1 axis, it has to be rotated $\frac{\pi}{2}$ radians about x_1. Thus, the value of α_1 is $\frac{\pi}{2}$.

Link 2/3:
Link 2 and Link 3 have the same relative position and orientation, which means they have a similar method of derivation of the DH parameter values. The links are rotated by joint 2 and joint 3, respectively, with joint variables q_2 and q_3. The coordinate frames $\{O_1\}$,$\{O_2\}$ and $\{O_2\}$ have all parallel axes, and therefore d_2,d_3,α_2 α_3 become 0. Unlike Link 1, the constant values of a_2 and a_3 are not 0 . This is because each frame is shifted along the x-axes of the frame that follows.

Link	d_i	θ_i	a_i	α_i
1	d_1	q_1	0	$\pi/2$
2	0	q_2	a_2	0
3	0	q_3	a_3	0

Table 3.2: The DH table for the Elbow Manipulator from Figure 3.1.

3.1.3 Solution to the Forward Kinematics Problem

The transformation matrix is a vital part of the calculation and presentation of the forward kinematics solution. In three-dimensional space, the transformation of a rigid object from one coordinate frame to another is described in regards to the frame pose. The transformation matrix is a mathematical representation of this transformation. For a robot manipulator, the transformation matrix is used to relate the linear and rotational movements of the joint to its neighboring links.

These link transformations can be combined of all the links between frame 0 and frame n. The resulting transformation matrix $\boldsymbol{T_n^0}$ represents the positional and rotational "awareness" of the origin of frame n relative to frame 0.

$$\boldsymbol{T}_n^0 = \begin{bmatrix} \boldsymbol{R}_n^0 & \boldsymbol{o}_n^0 \\ \boldsymbol{0}_{1\times 3} & 1 \end{bmatrix} \tag{3.1}$$

$\boldsymbol{T}_n^0 \in \mathcal{R}^{4\times 4}$ is the transformation matrix, $\boldsymbol{R}_n^0$ is a 3×3 rotation matrix which expresses the orientation of frame n relative to frame 0 and $\boldsymbol{o}_n^0$ is the position in relation to frame 0. A more general depiction for arbitrary frames k and j is given in (3.2)

$$\boldsymbol{T}_n^k = \begin{cases} \boldsymbol{A}_{k+1}\boldsymbol{A}_{k+2}\dots\boldsymbol{A}_{j-1}\boldsymbol{A}_j & \text{if } k < j \\ \boldsymbol{I} & \text{if } k = j \end{cases} \tag{3.2}$$

where $\boldsymbol{A}_i = \boldsymbol{A}(q_i) \in \mathcal{R}^{4\times 4}$ is the homogeneous transformation matrix expressing the pose of frame i with respect to $i-1$. Hence, the pose is a function of the joint variable.

$$\boldsymbol{A}_i = \begin{bmatrix} \boldsymbol{R}_1^{i-1} & \boldsymbol{o}_i^{i-1} \\ \boldsymbol{0} & 1 \end{bmatrix} \tag{3.3}$$

The way to solve the forward kinematics problem is to decide the transformation matrix from the base frame to the manipulator end-effector. First, the parameters and variables from the DH table are used to decide each homogeneous transformation matrix $\boldsymbol{A}_1$, $\boldsymbol{A}_2$, $\cdots$, $\boldsymbol{A}_e$ for each link where e denotes the end-effector.

$$\boldsymbol{A}_i(q_i) = \begin{bmatrix} c_{\theta_i} & -s_{\theta_i}c_{\alpha_i} & s_{\theta_i}s_{\alpha_i} & a_i c_{\theta_i} \\ s_{\theta_i} & c_{\theta_i}c_{\alpha_i} & -c_{\theta_i}s_{\alpha_i} & a_i s_{\theta_i} \\ 0 & s_{\alpha_i} & c_{\alpha_i} & d_i \\ 0 & 0 & 0 & 1 \end{bmatrix} \quad c_{\theta_i} = \cos(\theta_i)\ ,\ s_{\theta_i} = \sin(\theta_i) \tag{3.4}$$

Calculating this matrix is fairly straight forward since the parameter values α_i and a_i is taken directly from the DH-table, involving d_i and θ_i as the possible unknowns. The resulting transformation matrix $\boldsymbol{T}_e^0$ from (3.2) describes the transformation from link-frame 0 to the end-effector e . In other words, the solution to the forward kinematics problem.

3.2 Inverse Kinematics

Inverse kinematics is the reverse process of forward kinematics as it seeks to find what joint angles or positions (joint variables) are required to achieve a specific end-effector pose. Since the joint variables are normally governed by actuator motors, inverse kinematics can be used for robot control purposes. The usefulness of inverse kinematics becomes apparent when the goal of the end-effector is to interact with an object within its workspace.

Interaction with any sort of object requires knowledge of the pose of the object for the end-effector to match this pose. Hence, the transformation matrix of the object is assumed to be known. With the known pose, the inverse kinematics solution can be used to match the pose of the end-effector to the pose of the object. The solution is usually not unique, and there may be several combinations of joint values corresponding to the particular robot end-effector pose. Furthermore, for higher DoF robot manipulators, there are even more sets of possible joint variable solutions to keep track of. This makes inverse kinematics a difficult problem to solve for complex robot structures.

There are both analytical and numerical approaches to the inverse kinematics problem, and the most practical approach depends on the robot manipulator and its level of complexity. The problem becomes increasingly difficult as the number of joints in the robot increases. For typical robot configurations or particular classes of robots, solutions are often available in published literature [7]. On the other hand, some robot configurations might not have proper inverse kinematics solutions, implying a significant probability of singularities or solutions too complicated to be solved with a particular approach. For manipulators with six or more DoF, the possibility of finding a solution to the problem is very much dependent on a particular configuration called a *spherical wrist*, as it allows three joints to be modelled as one.

Spherical Wrist

Today, almost all industrial robots have a setup involving a spherical wrist with the purpose of simplifying the coordination between the motion of the arm and that of the end-effector [53]. The spherical wrist is a special type of manipulator configuration consisting of three revolute joints where all joint axes intersect at a common point called the *wrist center* [57]. The spherical wrist is pictured as a kinematic diagram in Figure 3.2. The intersecting axes become a spherical coordinate system, and the three joint variables can then be described as Roll, Pitch and Yaw motions for a single frame. Usually, the spherical wrist is added to an articulated part of a 6 DoF manipulator to orient the end-effector in space [57].

The spherical wrist greatly simplifies the calculation of the kinematics problems in general, but is especially important for the inverse kinematics solution. Although the spherical wrist consists of three joints, these three joints act as one contribution to the transformation and rotational movements of the end-effector. Therefore, the inverse kinematics problem of a 6 DoF manipulator may be partitioned into two simpler problems. First, the position of the wrist center is found based on the articulated part. Then, by only addressing the spherical wrist, the orientation of the wrist is found. Ultimately, the position and orientation of the end-effector is found separately. For a manipulator without a spherical wrist the position and orientation are coupled, and this complicates the coordination between the motion of the arm and that of the wrist [53].In conclusion, while the forward kinematics can be determined using standardized methods, the solving of the inverse kinematics is an often complex problem that is dependent with the manipulator design. Particularly, the complexity of the problem will often increase with more DoF and with a lack of a spherical wrist.

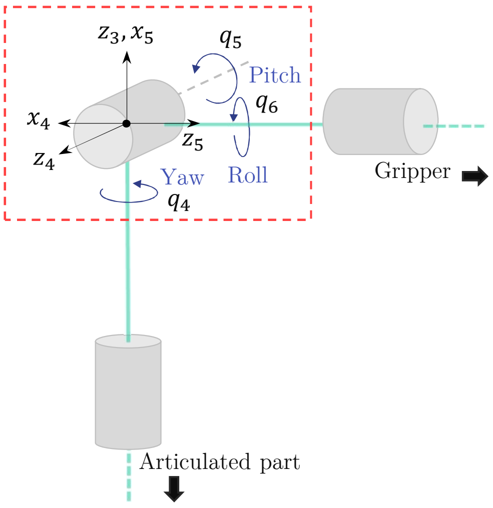


Figure 3.2: Spherical wrist with three joint axes intersecting at a common point.

3.2.1 Analytical Approaches

There are two analytic approaches to decide the closed-form solutions of the inverse kinematics problem, one of them is the geometrical approach [57][53][7]. The geometric solution is preferred for simple robots, e.g. a two link robot arm. This solution requires trigonometric intuition to find a solution. The main advantage is that it gives an insight into the multiple joint configuration possibilities resulting in the same end-effector pose. When the number of joints grow beyond two, the geometric solution tend to become too difficult to solve. Even when it is possible to find a geometric solution, there is typically a low level of generalization within the solution method. Hence, the chance of finding a solution is dependant on previous experience and expertise within the subject.

A more generalized approach to the analytical solution, is the algebraic approach. Referencing (3.5), the transformation matrix is given and therefore the equations for $r_{i,j}$ and (x_e^0, y_e^0, z_e^0) may be found by the method of forward kinematics. These equations can then be used to decide the unknown joint variables

$\boldsymbol{q} = [q_1, q_2, \dots q_e]^T$. Similar to the geometric approach, there is no general method for finding the closed-form solution, which makes the algebraic approach challenging for complex robot manipulators.

$$\boldsymbol{T}_e^0(\boldsymbol{q}) = \begin{bmatrix} r_{11} & r_{12} & r_{13} & x_e^0 \\ r_{21} & r_{22} & r_{23} & y_e^0 \\ r_{31} & r_{32} & r_{33} & z_e^0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{3.5}$$

There are publications with the conclusion that any six DoF robot with a spherical wrist have an analytical solution [51]. However, the criterion that a spherical wrist confirms an analytical solution, called the *Pieper Criterion*, has also been shown to be questionable in terms of its completeness. Deciding if a robot arm has an analytical solution is therefore challenging. If the Pieper Criterion was assumed to be true, both the analytical approaches are still time consuming and error prone when used for complex robot manipulators.

3.2.2 Numerical Approaches

An alternative to the analytical solution is the numerical solution which is applicable when the analytical solution does not exist, or it is just too hard to figure out [57][53][7]. A general solution for a six joint robot manipulator with no spherical wrist usually calls for a numerical solution. The numerical approach relies on the fact that it is always possible to determine the forward kinematics of the robot manipulator. One numerical approach is formulated as an optimization problem followed by an iterative solution. There are other approaches, but they will not be described in detail in this thesis. The optimization solution is based on the DH-convention.

The setup of the numerical optimization approach can be described in the following manner. First, to reach the desired pose, the joint variables $\boldsymbol{q}$ need to be adjusted until they match the desired joint values $\boldsymbol{q^*}$. This can be setup as a mathematical optimization problem by inserting the adjusted joint values for each step until the forward kinematics matches the desired pose [7]. A formal mathematical formulation of the optimization problem is to minimize the error between the forward kinematics solution and the desired pose $\boldsymbol{T}^*(\boldsymbol{q^*})$

$$\text{Minimize } \boldsymbol{E(q)} = \left\| [\boldsymbol{T(q)}]_t - \boldsymbol{T}^*(\boldsymbol{q^*}) \right\| \tag{3.6}$$

subjected to workspace and geometrical constraints. These constraints are usually designed with the aim of avoiding singular poses. However, because the possibility for singular poses is always present, the stability of the numerical solution and its convergence rate cannot be guaranteed for any of the numerical approaches [51].

The numerical solution has three main drawbacks [7]:

1. Slow for practical applications.
2. Not always able to find all the possible solutions, hence the identified pose may be sub-optimal.
3. Might be unstable due to singularities. Singularities occur when the robot is outside its workspace or in an impossible configuration.

Since the analytical approach has poor scalability and the numerical solution is unusable for most real-time problems, there is currently research invested in finding better and more generalized solutions. The Jacobian-based inverse kinematics solver and techniques described by [41] allow for a numerical solution as an efficient real-time solver. Another interesting approach for robots satisfying the Pieper-criterion is presented in [36]. Following the rapid development of artificial intelligence, this technology is key for further developments within robotics research. One example of inverse kinematics solvers based on artificial intelligence is [15] that proposes a solution algorithm based on improved back propagation neural network.. Another example is [65] which is able to mimic human robotics experts to form a solution, using a Behaviour Tree. In the end, based on the available research today, if an analytical solution can be found, it is always preferable to a numerical solution for the execution speed and the ability to pick the desired joint configuration among multiple solutions.

Chapter 4

Dynamics

[1]While the problem of kinematics is about describing the motion of the robot manipulator while disregarding the acting forces, the dynamics problem seeks to find the explicit relation between forces and robot motion. This relation appears in the form of the equation of motion, also called the dynamic model of the robot manipulator. The dynamic model is valuable for computations used for simulation, analysis of manipulator structures and design of control algorithms [53]. The most important concepts required for the derivation of the equation of motion are mentioned in Spong [57] and described in more detail in Egeland [10].

Section 4.1 presents the equations of motion and two different approaches to determine the dynamic model. The Euler-Lagrange and Newton-Euler approaches are introduced and a comparison between the methods is described briefly. Section 4.2 involves the derivation of the equation of motion using the Euler-Lagrange method for a n-DOF robot arm. The derivation using Newton-Euler is not presented any further in this project thesis. A more general and compact derivation of the inertial matrix for robot manipulators is also added at the end of section 4.2. The last section of the chapter (Section 4.3) explains the difference between the forward and inverse dynamics based on the setup of the equations of motion.

4.1 Equations of Motion

Dynamics is the analysis of motion caused by forces. This requires parameters like mass and inertia to calculate the acceleration of the bodies. The goal is to create a mathematical model describing the motion of a structure in terms of the forces and torques acting on it. The equation of motion is a compact version of this mathematical model typically called the dynamic model.

The motion of a robot manipulator is affected by internal and external forces acting on its rigid links. In this chapter, a land-based manipulator is assumed without any form of external forces, except gravity. Such forces tend to be situational and are therefore usually omitted for any sort of general-purpose dynamics model. The remaining forces are the internal mechanical forces of the manipulator. The mechanical connection between the links comes in the form of force and torque exerted by neighboring links where each link is supported

[1]Most of this chapter is based on a chapter in the specialization project thesis, but is slightly modified. Specifically, Section 4.1 involves some extra details regarding the method of choice.

by the preceding links [7]. Joint torques are applied from the actuators in each joint and might be controlled to achieve a particular manipulator state. Figure 4.1 demonstrates a simple two-dimensional robot arm and its corresponding free-body diagram. The diagram displays the forces between the links (F), the torques (τ) provided by the actuators, and the gravitational weight (G) acting on the center of mass (CoM). The type and values of all forces acting on the robot manipulator depend on the type of robot and its interactions with the working environment.

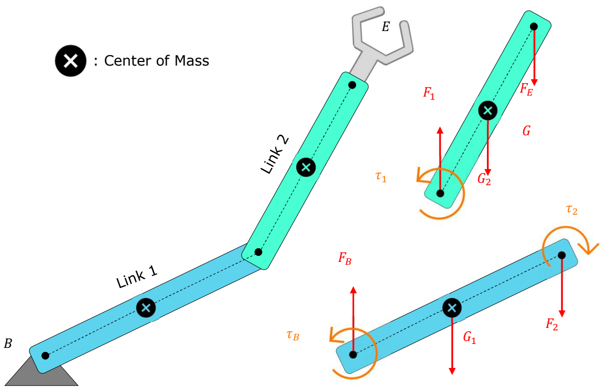


Figure 4.1: Simple 2-DOF 2D robot arm with corresponding free body diagram.

When it comes to describing the dynamics there are different approaches. Two common approaches, often described in robotics literature, are the Euler-Lagrange and Newton-Euler approaches [57][53]. The two different formulations lead to the same matrix form of the equations of motion (4.1), but the route to get there and the structure of the equations within the matrices are different.

$$\boldsymbol{M}(\boldsymbol{q})\ddot{\boldsymbol{q}} + \boldsymbol{C}(\boldsymbol{q},\dot{\boldsymbol{q}})\dot{\boldsymbol{q}} + \boldsymbol{G}(\boldsymbol{q}) = \boldsymbol{\tau} \tag{4.1}$$

where

$\boldsymbol{q}$	=	vector of joint variables
$\boldsymbol{\tau}$	=	vector of joint torques
$\boldsymbol{M}$	=	inertia matrix
$\boldsymbol{C}$	=	centrifugal and Coriolis terms
$\boldsymbol{G}$	=	gravity vector.

Table 4.1 presents a compact summary of some of the comparable factors regarding the equations of motion. Contrasting conceptual approaches and

different methods of derivation lead to distinct method properties. The Euler-Lagrange approach necessitates the use of coordinate transformations. The result is a highly structured but computationally inefficient formulation. The Newton Euler formulation gives efficient recursive equations but is also more challenging to use when deriving advanced control laws [8]. Therefore, for real-time control and simulation, the equations of Newton-Euler are often favored; however, if working with Lyapunov designs or passivity, the energy-based Euler-Lagrange equations might be more suitable [10]. Ultimately, the two methods have their own merits and demerits. Therefore, the choice of method may conclusively be based on a combination of personal preference and problem practicality.

Another method to be mentioned is Kane's method, which in some sense is a combination of the Euler-Lagrange and Newton-Euler method [58]. Kane's method provides a straight forward approach for incorporating environmental external forces in to the dynamic model. Similarly to the Newton-Euler it is less computationally demanding than the Euler-Lagrange approach, while also eliminating some of the unnecessary force calculations of the Newton-Euler approach. The method has also been proven to result in a more accurate approximation than the Euler-Lagrange, when modeling a two-link manipulator [47].

In this thesis, the method of choice is Euler-Lagrange. Having a well-structured formulation is advantageous when presenting the dynamics of a complex robot manipulator, making it easy to presents the procedure and the results in an orderly manner as a foundation for continued work. Additionally, the Euler-Lagrange is the method most familiar to the author.

	Euler-Lagrange Method	**Newton-Euler Method**
Conceptual approach:	Energy based.	Newton's second law (force balance).
Derivation:	Each link of the manipulator is treated in turn.	Manipulator treated as one system.
Equations of motion:	Structured form.	Less structured form.
Computations:	Less efficient.	Efficient.

Table 4.1: Comparing the Euler-Lagrange and Newton-Euler approach.

4.2 Euler-Lagrange Method

The Euler-Lagrange method is based on the basics of Lagrange mechanics. Lagrange mechanics is a tool to build mathematical models for complex mechanical systems. The mechanical system is described in terms of energy. The energy based description requires the computation of the kinetic ($\mathcal{K}$) and potential ($\mathcal{P}$) energy. In terms of these energy equations, the Lagrange equation ($\mathcal{L}$) is computed as

$$\mathcal{L} = \mathcal{K} - \mathcal{P}. \tag{4.2}$$

For a robot manipulator, building the Lagrange equation can be done in a systematic manner. The Lagrangian framework is convenient for a system based on a configuration relative to a reference frame. The configuration is described in terms of a vector of generalized coordinates. When working with a robot arm, the generalized coordinates are equal to its joint variables $\boldsymbol{q}(t) \in \mathbb{R}^n$, where n is the number of joints. One basic assumption about the manipulator which is required before calculating the Lagrange equation, is that the n links in motion are considered as rigid bodies [53].

4.2.1 Kinetic Energy

One way to calculate the general kinetic energy for a n-link robot is by employing the linear and angular velocity Jacobians to decide the inertia matrix together with the derivative of the joint variables [57]. The linear and angular velocities are expressed in terms of the Jacobian matrices in the form of (4.3) and (4.4) respectively.

$$\boldsymbol{J}_v = [\boldsymbol{J}_{v_1} ... \boldsymbol{J}_{v_n}] \tag{4.3}$$

$$\boldsymbol{J}_\omega = [\boldsymbol{J}_{\omega_1} ... \boldsymbol{J}_{\omega_n}] \tag{4.4}$$

The value of the Jacobians depends on the type of joint, and is summarized as

$$\boldsymbol{J}_{v_i} = \begin{cases} \boldsymbol{z}_{i-1} \times (\boldsymbol{o}_n - \boldsymbol{o}_{i-1}) & \text{for revolute joint } i \\ \boldsymbol{z}_{i-1} & \text{for prismatic joint } i \end{cases} \tag{4.5}$$

$$\boldsymbol{J}_{\omega_i} = \begin{cases} \boldsymbol{z}_{i-1} & \text{for revolute joint } i \\ \boldsymbol{0} & \text{for prismatic joint } i \end{cases} \tag{4.6}$$

where z_i and o_n can be found directly from the transformation matrices found in the forward kinematics problem, see (4.7).

$$\begin{cases} \boldsymbol{z}_i & \text{given by the first three elements in the third column of } \boldsymbol{T}_i^0 \\ \boldsymbol{o}_i & \text{given by the first three elements in the fourth column of } \boldsymbol{T}_i^0 \end{cases} \tag{4.7}$$

Hence, only the third and fourth columns of the transformation matrices are required to evaluate the the velocity Jacobians.

If the mass of link i is denoted as m_i, it is possible to calculate the inertia matrix of link i ($\boldsymbol{M}_i$) by using the velocity Jacobians, the inertia tensor ($\boldsymbol{I}_i$) and the rotation matrix ($\boldsymbol{R_i}$) of the link. In this thesis the gear ratio, denoted as ($\boldsymbol{J}_m$) in Spong [57], is assumed to be negligible. Thus, the inertia matrix is calculated as

$$\boldsymbol{M}_i(\boldsymbol{q}) = m_i \boldsymbol{J}_{v_i}(\boldsymbol{q})^T \boldsymbol{J}_{v_i}(\boldsymbol{q}) + \boldsymbol{J}_{\omega_i}(q)^T \boldsymbol{R}_i(\boldsymbol{q}) \boldsymbol{I}_i \boldsymbol{R}_i(\boldsymbol{q})^T \boldsymbol{J}_{\omega_i}(\boldsymbol{q}) \tag{4.8}$$

where $\boldsymbol{I}_i$ is the inertia tensor in the body attached frame of link i, but evaluated around the link center of mass (see figure 4.1). The matrix values are only dependent on the geometrical configuration of the rigid body [57]. Consequently, the inertia matrix given by (4.9) is constant and independent of any motion. Furthermore, the value of the inertia matrix is typically included in the robot documentation.

$$\boldsymbol{I}_i = \left.\begin{bmatrix} I_{xx} & I_{xy} & I_{xz} \\ I_{yx} & I_{yy} & I_{yz} \\ I_{zx} & I_{zy} & I_{zz} \end{bmatrix}\right|_i \tag{4.9}$$

The inertial matrix for an n-Link robot is found as a sum of the inertia matrix of all the links.

$$\boldsymbol{M}(\boldsymbol{q}) = \sum_{i=1}^{n} \boldsymbol{M}_i(\boldsymbol{q}) \tag{4.10}$$

In the end, the kinetic energy for a n-Link robot becomes

$$\mathcal{K} = \frac{1}{2} \dot{\boldsymbol{q}}^T \boldsymbol{M}(\boldsymbol{q}) \dot{\boldsymbol{q}} \tag{4.11}$$

where

$$\boldsymbol{q}^T = [q_1, q_2, \cdots, q_n]. \tag{4.12}$$

and

$$\dot{\boldsymbol{q}}^T = [\dot{q}_1, \dot{q}_2, \cdots, \dot{q}_n]. \tag{4.13}$$

4.2.2 Potential Energy

In the case of rigid dynamics of a robot manipulator, gravity is typically the only source of potential energy. The potential energy of the i-th link is computed while assuming the mass of the entire link is concentrated at its center of mass as illustrated in figure 4.1.

$$\boldsymbol{\mathcal{P}}_i = \boldsymbol{g}_0^T \boldsymbol{r}_{ci} m_i \tag{4.14}$$

The sum of the contributions from each link, is the total amount of potential energy stored the n-link robot and is given by

$$\boldsymbol{\mathcal{P}} = \sum_{i=1}^{n} \boldsymbol{g}_0^T \boldsymbol{r}_{ci} m_i \tag{4.15}$$

where $\boldsymbol{g}_0$ is the gravity acceleration vector and $\boldsymbol{r}_{ci}$ is the coordinate vector of the center of mass of link i. Both $\boldsymbol{g}_0$ and $\boldsymbol{r}_{ci}$ are given relative to the base frame [57]. By denoting the vectors as

$$\boldsymbol{r}_{ci} = [r_{cx},\ r_{cy},\ r_{cz}]^T \tag{4.16}$$

and

$$\boldsymbol{g}_0 = [g_x,\ g_y,\ g_z]^T\ . \tag{4.17}$$

4.2.3 Euler-Lagrange Equation

Based on the Lagrangian equation (4.2), the Euler-Lagrange equation for an n-link manipulator is calculated as

$$\frac{d}{dt}\frac{\partial \boldsymbol{\mathcal{L}}}{\partial \dot{\boldsymbol{q}}} - \frac{\partial \boldsymbol{\mathcal{L}}}{\partial \boldsymbol{q}} = \boldsymbol{\tau}. \tag{4.18}$$

.

Here, $\boldsymbol{\tau}$ is a vector (4.19) which represents the input generalized forces equal to the motor torques at the joints [10]. The torque τ_k is associated with the generalized coordinate q_k [53].

$$\boldsymbol{\tau} = \begin{bmatrix} \tau_1 \\ \tau_2 \\ \vdots \\ \tau_n \end{bmatrix} \tag{4.19}$$

From the calculation of the Euler-Lagrange equation [57], it is possible to rewrite (4.18) as

$$\sum_{i=1}^{n} M_{kj}(\boldsymbol{q})\ddot{q}_j \; + \sum_{i,j=1}^{n} c_{ijk}(\boldsymbol{q})\dot{q}_i\dot{q}_j + \phi_k(\boldsymbol{q}) \; = \tau_k,\; k = 1, ..., n \tag{4.20}$$

where ϕ_k and D_{kj} may be calculated individually, and C_{ijk} (known as Christoffel symbols [57]) can be found from D_{kj}, see (4.21 - 4.24) .

$$\phi_k(\boldsymbol{q}) = \frac{\partial \mathcal{P}}{\partial q_k} \tag{4.21}$$

$$M_{kj}(\boldsymbol{q}) = \frac{\partial(\frac{\partial \mathcal{K}}{\partial \dot{q}_j})}{\partial \dot{q}_k} \tag{4.22}$$

$$c_{ijk}(\boldsymbol{q}) = \frac{1}{2}\Big\{\frac{\partial M_{kj}}{\partial q_i} + \frac{\partial M_{ki}}{\partial q_j} - \frac{\partial M_{ij}}{\partial q_k}\Big\} \tag{4.23}$$

$$C_{kj} = \sum_{i=1}^{n} c_{ijk}(q)\dot{q}_i \; = \frac{1}{2}\sum_{i=1}^{n}\Big\{\frac{\partial M_{kj}}{\partial q_i} + \frac{\partial M_{ki}}{\partial q_j} - \frac{\partial M_{ij}}{\partial q_k}\Big\}\dot{q}_i \tag{4.24}$$

In the case of manipulators of higher complexity, which implies higher degrees of freedom, the Euler-Lagrange equation for an n-link manipulator is normally written in a matrix form, equivalent to how the equation of motion (4.1) is presented in section 4.1.

The correlation between (4.1) and (4.20) implies the following characteristics:

ϕ_k	$=$	the k-th element of the gravity vector $\boldsymbol{G(q)} \in \mathbb{R}^n$
M_{kj}	$=$	the k, j-th element of the manipulator inertia matrix $\boldsymbol{M}(\boldsymbol{q}) \in \mathbb{R}^{n\times n}$
C_{kj}	$=$	the k, j-th element of the centrifugal and Coriolis terms matrix $\boldsymbol{C(q, \dot{q})} \in R^{n\times n}$.

As a result of the characteristics above, the equation of motion for an n-link robot manipulator includes matrices on the form of 4.25.

$$\begin{bmatrix}\tau_1\\ \tau_2\\ \vdots\\ \tau_n\end{bmatrix} = \underbrace{\begin{bmatrix}M_{11} & M_{12} & \dots & M_{1n}\\ M_{21} & M_{22} & \dots & M_{2n}\\ \vdots & \ddots & \ddots & \vdots\\ M_{n1} & M_{n2} & \dots & M_{nn}\end{bmatrix}}_{\boldsymbol{M(q)}} \begin{bmatrix}\ddot{q}_1\\ \ddot{q}_2\\ \vdots\\ \ddot{q}_n\end{bmatrix} + \underbrace{\begin{bmatrix}C_{11} & C_{12} & \dots & C_{1n}\\ C_{21} & C_{22} & \dots & C_{2n}\\ \vdots & \ddots & \ddots & \vdots\\ C_{n1} & C_{n2} & \dots & C_{nn}\end{bmatrix}}_{\boldsymbol{C(q,\dot{q})}} \begin{bmatrix}\dot{q}_1\\ \dot{q}_2\\ \vdots\\ \dot{q}_n\end{bmatrix} + \underbrace{\begin{bmatrix}\phi_1\\ \phi_2\\ \vdots\\ \phi_n\end{bmatrix}}_{\boldsymbol{G(q)}} \tag{4.25}$$

$\boldsymbol{M(q)}$ can be computed either as done in (4.10) or (4.22).

4.3 Forward Dynamics

The matrix form of the Euler-Lagrange equation (4.1) is referred to as inverse dynamics because the equation setup maps motion to torque (4.26). This is useful for determining control laws [57].

$$(\boldsymbol{q}, \dot{\boldsymbol{q}}, \ddot{\boldsymbol{q}}) \to \boldsymbol{\tau} \tag{4.26}$$

Forward dynamics is the other way around, where the joint acceleration $\ddot{\boldsymbol{q}}$ is determined as a function of the joint torque. Solving the forward dynamics is useful for simulations of the manipulator because it maps torque to motion (4.27).

$$\boldsymbol{\tau} \to (\boldsymbol{q}, \dot{\boldsymbol{q}}, \ddot{\boldsymbol{q}}) \tag{4.27}$$

To be able to define the motion, based on the torque values, the Euler-Lagrange equation needs to be altered. The joint accelerations (4.28) are found based on the inverse of the inertia matrix [57].

$$\ddot{\boldsymbol{q}} = \boldsymbol{M}^{-1}(\boldsymbol{q})\big[\boldsymbol{\tau} - \boldsymbol{C}(\boldsymbol{q}, \dot{\boldsymbol{q}})\dot{\boldsymbol{q}} - \boldsymbol{g}(\boldsymbol{q})\big] \tag{4.28}$$

The form of (4.28) is the forward dynamics model in its explicit form where $\boldsymbol{M}$ is invertible[57]. $\boldsymbol{M(q)}$ might be a big and complex matrix, hence the inverse can become exceedingly complex [10]. In this case, having the model in its implicit form (4.29) or working with $\boldsymbol{M(q)}^{-1}$ numerically is sometimes required.

$$\boldsymbol{M}(\boldsymbol{q})\ddot{\boldsymbol{q}} = [\boldsymbol{\tau} - \boldsymbol{C}(\boldsymbol{q}, \dot{\boldsymbol{q}})\dot{\boldsymbol{q}} - \boldsymbol{G}(\boldsymbol{q})\big] \tag{4.29}$$

Chapter 5

Dynamics for Underwater Manipulators

In order to accurately describe the dynamics of a robot manipulator submerged in water, the model must take into account hydrodynamic and hydrostatic added effects. Land-based manipulators, as was modeled in Chapter 4, operate in air which is much lighter relative to the manipulator. Hence, environmental contributions were ignored. In underwater applications, the impact of the environment cannot be neglected [26]. The joint torques have to overcome the added weight along with the hydrodynamic load induced by the relative motion of the arm and the water. The expanded dynamics model for a multibody system is usually decoupled into independently computed inertia, drag force, fluid accelerations, and buoyancy, although these phenomena are strongly coupled [26][64][40]. It is important to note that the theory of fluid dynamics is complex, and it is challenging to develop a reliable model for most of the hydrodynamic effects [1].

This chapter[1] proposes a generalized method, by combining and modifying methods from different studies and literature, for adding underwater effects to the equations of motion, specifically directed at smolt production fish tanks. At first, in Section 5.1 a couple of preliminary notations and a set of assumptions for the computations will be presented. Section 5.2 explains the updated equations of motion with added underwater effects. Lastly, Section 5.3, 5.4 and 5.5 presents the suggested methods for calculating the added mass, drag forces and buoyancy forces respectively.

5.1 Preliminary Notations and Assumptions

To create a hydrodynamics model of a system with multiple rigid bodies, it is practical to focus on one link at a time. Individual focus on the links allows for utilization of some of the general methods for single-bodied systems presented in literature [19][1][33]. The body-fixed reference frames have axes x, y and z in line with the DH-notation, with the corresponding angular movement q around the z-axis. A reoccurring challenge is related to the combination of the DH-convention with the underwater effects. Due to the highly three dimensional nature of these effects, the use of DH-notations, with a main focus on the relative motion of the

[1]This chapter is completely new, and not based on specific chapters in the specialization project thesis.

links alone, will either require necessary alterations to the computations, or act as an argument for simplifications.

Due to the complexity of fluid dynamics, a number of key assumptions are to be made:

1. The robot manipulator is considered to be moving from a fixed based; totally submerged in water at all times.
2. The links are approximated as circular cylinders with link specific mass, radius and length values.
3. The current in the tank is assumed constant and irrotational in the base frame of the manipulator [1].
4. The skin friction is assumed to be zero.
5. The water density is assumed to be $\rho = 1000\ kg/m^3$

Assumption 1 was based on the fact that calculating the hydrodynamic effects on partially submerged bodies escalates the complexity within the dynamic model [33]. Hence, to limit the scope, the thesis only considers fully submerged bodies. In terms of assumption 2, it is common to model the links as cylinders when working with hydrodynamics of underwater arms, because it represents a manageable problem and it is close to the real world [38]. For assumption 5, the water density will vary based on the water's salinity and temperature. However, as these variations are small, the water density is assumed to be constant. This is a common assumption to make [1].

5.2 Dynamics for Underwater Rigid Bodies

For an underwater robot manipulator there are two additional sources creating forces on rigid bodies. In the previously calculated land-based model, the only forces included in the model were the internal mechanical forces together with gravity. For the underwater environment, there are now added external forces. This means the sum of generalized forces on the rigid links of the manipulator becomes

$$\boldsymbol{\tau} = \underbrace{\boldsymbol{\tau}_{hd} + \boldsymbol{\tau}_{hs}}_{\text{External}} + \boldsymbol{\tau}_m \tag{5.1}$$

where $\boldsymbol{\tau}_{hd}$ are the hydrodynamic forces, $\boldsymbol{\tau}_{hs}$ are the hydrostatic forces and $\boldsymbol{\tau}_m$ are the forces from the manipulator alone, as found in (4.1). The term *hydrostatic* implies the force is not a function of the relative movement between body and water [1]. Buoyancy is the only hydrostatic effect. The *hydrodynamic* forces are due to the water motion, and there are multiple hydrodynamic effects to take into consideration. The resulting equations of motion with the added hydrodynamic and hydrostatic forces, inspired by the works of [19], [1] and [33], is given by

$$\boldsymbol{M_m}(\boldsymbol{q})\ddot{\boldsymbol{q}}+\boldsymbol{C_m}(\boldsymbol{q},\dot{\boldsymbol{q}})\dot{\boldsymbol{q}}+\underbrace{\boldsymbol{M_A}\ddot{\boldsymbol{q}}+\boldsymbol{C_A}(\boldsymbol{q},\dot{\boldsymbol{q}})+\boldsymbol{N_D}(\boldsymbol{q},\dot{\boldsymbol{q}})}_{\text{hydrodynamic forces}}+\boldsymbol{G_m}(\boldsymbol{q})+\underbrace{\boldsymbol{G_B}(\boldsymbol{q})}_{\text{hydrostatic forces}}=\boldsymbol{\tau} \tag{5.2}$$

where the matrices $\boldsymbol{M}_A \in \mathbb{R}^{6\times6}$, $\boldsymbol{C}_A \in \mathbb{R}^{6\times6}$ and $\boldsymbol{N}_D \in \mathbb{R}^{6\times1}$ represent the hydrodynamic added mass inertia matrix, added mass Coriolis and centripetal terms, and the drag forces respectively. $\boldsymbol{G}_B \in \mathbb{R}^{1\times6}$ is the buoyancy forces. $\boldsymbol{M_m}$, $\boldsymbol{C_m}$ and $\boldsymbol{G_m}$ are the same as in (4.1).

5.3 Added Mass Effects

When the manipulator components are accelerated in an underwater environment, they create an additional hydrodynamic reaction force affecting the robot arm's movements. This effect is described by an added mass contribution. Besides, since the added mass is proportional to the vehicle acceleration [42], it is sometimes denoted as a *virtual mass*. The reaction force opposes the motion of the manipulator. Therefore, the applied force from the actuators must overcome the added inertia and the manipulators' own inertia. The added mass effect on the inertia is neglected for land-based robots due to the low density of air compared to water [1]. Ultimately, the added mass will be a function of the fluid density and the body's surface geometry. The properties of the inertia and Coriolis and centrifugal matrices will change accordingly.

Parts of the hydrodynamic effect on the robot manipulator's inertia is compiled in a matrix called the *added mass matrix* $\boldsymbol{M}_A \in \mathbb{R}^{6\times6}$. For an ideal fluid with low manipulator velocity, no currents or waves, and if assuming frequency independent matrices [1][43], the Added Mass matrix is symmetric and positive definite:

$$\boldsymbol{M}_A = \boldsymbol{M}_A^T > \boldsymbol{0} \tag{5.3}$$

Water is often assumed to have the properties of an ideal fluid. By approximating the manipulator as a a chain of cylindrical shaped links moving in slow speeds, the added mass contribution of link i can be approximated by [19]

$$\boldsymbol{M}_{A_i} = -diag\{X_{\dot{u}}, Y_{\dot{v}}, Z_{\dot{w}}, K_{\dot{\psi}}, M_{\dot{\phi}}, N_{\dot{q}}\}|_i \tag{5.4}$$

where

$$\begin{array}{rcl} X_{\dot{u}} & = & -0.1m_i \\ Y_{\dot{v}} & = & -\pi\rho r_i^2 L_i \\ Z_{\dot{w}} & = & -\pi\rho r_i^2 L_i \\ K_{\dot{\psi}} & = & 0 \\ M_{\dot{\phi}} & = & -\frac{1}{12}\pi\rho r_i^2 L_i^3 \\ N_{\dot{q}} & = & -\frac{1}{12}\pi\rho r_i^2 L_i^3. \end{array}$$

L_i, m_i and r_i are the length, mass and circular section radius of link i respectively. The mass coefficients mentioned above are calculated by applying strip theory to the cylindrical links [18]. Deriving the hydrodynamic effects, using strip theory, benefit from the simplified geometric assumption on the links being cylinders.

As a result of the added mass having a restrictive effect on a rigid body's operational and rotational motion, it contributes to the Coriolis and centripetal terms as well. Adopting the same approach as in Chapter 4, the Coriolis and centripetal terms are found with the Christoffel symbols on the added mass inertia matrix $\boldsymbol{M_A}$. However, since (5.4) is independent of the joint angles, the added mass effect to the Coriolis and centripetal matrix would be theoretically nonexistent. According to [19], a simplified hydrodynamic Coriolis and centrifugal matrix for a single rigid body can be calculated based on the constant terms of 5.4, the linear velocities, and the Euler-angle velocities. To correlate these parameters in terms of the DH-convention makes for complex computations with little benefit due to low manipulator velocity. Furthermore, literature on underwater dynamics for manipulators tend to ignore this effect [40][34][49][52][63]. In conclusion, the added mass effect on the Coriolis and centripetal terms $\boldsymbol{C_A}$ are set to zero.

5.4 Drag Forces

The drag force is a force on a body opposing its motion. When a manipulator moves through water, the viscosity of the fluid causes damping effects in the form of drag and lift forces. The drag force is a sum of two resistive fluid forces: friction drag and pressure drag [62]. The direction of friction drag is collinear to the body surface while pressure drag is parallel with the flow velocity. The friction drag us usually neglected because its values tend to be small.

When calculating the pressure drag in water, a general formulation is given by [43]

$$D = \frac{1}{2}C_D\rho A V^2 \tag{5.5}$$

which involves the drag coefficient C_D, the water density ρ, the projected area A perpendicular to the flow direction, and the body velocity relative to the fluid V. The velocity squared dependency is the dominant feature. The water density is normally set to $1000kg/m^3$. The drag coefficient is a value that summarizes the complex dependencies of shape and flow conditions related to drag. The value of the drag coefficient is sometimes assumed to be constant [33], other times it is set to vary with the development of the flow conditions [38].

5.4.1 Previous Work

In scientific studies, the determination of the functional drag force on underwater manipulators has been done using different methods, involving varying levels of complexity in regards to the drag coefficient. Originally, studies seeking to model the drag force on manipulators assumed a constant drag coefficient [40][33], or a drag coefficient varying based on the angle of attack and the Reynolds Number [49]. However, none of these papers were validated experimentally. McLain [37] then proved experimentally that using constant coefficient values did not result in an accurate model of the drag on a manipulator for a three dimensional flow. The inaccuracy was mainly a result of how the fluid velocities form varying pressure gradients along the length of each link.

Building on the work of McLain, [39] performed more detailed experiments validating the use of a variable drag coefficient on a single link. The conclusion was that the the drag coefficient of a cylinder is a function explicitly of how far the the cylinder has traveled.In [31] the same model formulation based on distance traveled was used, but with a two-linked manipulator, further emphasizing the flow characteristics effect on neighboring links. The two-linked manipulator showed that the drag is also a function of the angle between the two links. To account for the flow in 3D, a standard strip theory approach was applied in all the studies mentioned.

5.4.2 Computation of the Drag

Given the complexity of fluid flow and the difficulty of performing reliable simulations, the drag computations must be consistently evaluated in terms of possible approximations. Therefore the process towards tolerating the physical inaccuracies of the drag coefficient by cause of computational complexity is described initially. In the end, a proposed solution, based on the fluid flow angle of attack, is presented.

Variable Drag Coefficient

Determining the drag force often involve approximations regarding the fluid behaviour which means the level of abstraction in the model may vary. One of the decisions to be made in terms of complexity, is whether to use a constant or variable drag coefficient. By assuming the drag coefficient of a manipulator link is not constant, but a function of how far the link has traveled, [37] proposes the following drag force for a cylindrical link in 2D:

$$F_D = C_d(s/D)\frac{1}{2}\rho D U^2 \tag{5.6}$$

where ρ is the water density, C_D is the drag coefficient, s is the link's displacement from its initial resting position, D is the diameter of the link and U is the velocity of the link. The ratio s/D represents the distance travelled based on the link diameter.

Using strip theory, the analysis can be extended to three dimensions. The strip theory approach is depicted in Figure 5.1 a model that accurately describes the drag force on a segmented part of the link is represented on the form:

$$dF_{D_i} = C_{d_i}(l_i\theta/D)\frac{1}{2}\rho D|U|U dl \tag{5.7}$$

where dl is the length of the link segment. The distance traveled of the i^{th} segment is due solely to rotation about the current link-frame: $s_i = l_i\theta$.

Adding the contributions from each of the segmented parts of the link results in the total drag force:

$$F_d = \sum_{i=1}^{n} dF_{d_i} \tag{5.8}$$

where n is the number of segments used in the model.

The progressive development to model the drag force, even with simplifications, results in a relatively high level of complexity for a single link system. This is exemplified in [38], where the variable drag coefficients are simplified even further to what is labeled as *average coefficient functions.* Still a function of the distance travelled by the link. The level of complexity is elevated all the more for a manipulator with multiple DoF. A legitimate assumption to lower the computational demand, for complicated manipulators in particular, is to use a constant drag coefficient.

Proposed Solution Based on Angle of attack

The drag force is proportional to the squared relative velocity, and the velocity value is therefore a decisive component when calculating the drag. In general, an accurate account for the relative velocity is based on the angle of attack α_d of the flow velocity $\boldsymbol{U_F}$ on the body (see Figure 5.2a) [11]. By rewriting the general formula (5.5) to

$$\boldsymbol{F_D} = \frac{1}{2}C_N\rho\boldsymbol{U^2}lD, \tag{5.9}$$

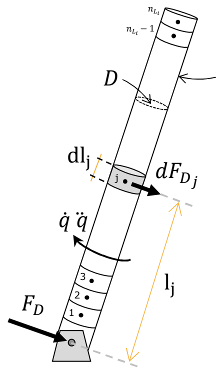


Figure 5.1: Diagram of Strip-Theory Implementation

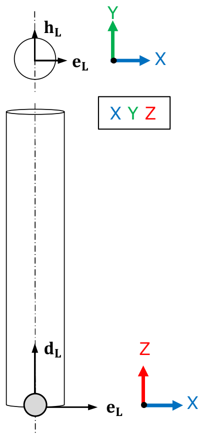
where D is the link diameter and l is the link length, it is possible to decompose the relative velocity $\boldsymbol{U}$ to the normal and tangential directions (see 5.2b). C_N is a notation used to illustrate that the drag will generate a force normal to the link's axis. Since the current is assumed to be constant and irrotational, it is possible to decompose the relative velocity V the relative velocity to the normal and tangential directions: $\boldsymbol{U} = [U_x \quad U_y \quad U_z]^T$. Applying the cross-flow principle [33] the value of the coefficient becomes

$$C_N = C_D \sin^2 \alpha_d. \tag{5.10}$$

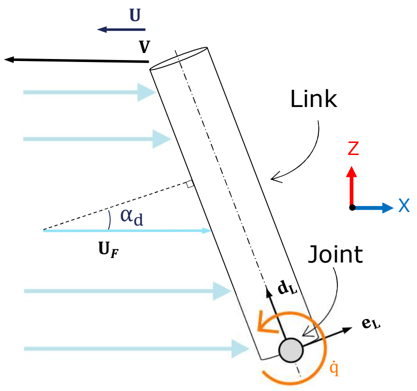
For simplicity, the function arguments of the drag coefficient C_D are omitted.

The calculation the total drag force on each link, is based on the algorithm proposed by Lévesque [33]. A modified version of the algorithm is presented in the Appendix A.1. The computations involve a secondary local system of orthonormal unit vectors $\boldsymbol{d_L}$, $\boldsymbol{e_L}$ and $\boldsymbol{h_L}$ (see Figure 5.2a). The main purpose of the secondary system is to standardize the geometric definitions of every link due to the fact that the current DH representation of the link motions is directly associated with the specific joints, and so are the coordinate frames. Thus, the benefit of the unit vectors is to be used as reference axes when determining the direction of the resulting drag force on the link (perpendicular to the links longitudinal axis $\boldsymbol{d_L}$). Figure 5.2b includes the X, Y and Z axes corresponding to the stationary base frame of the manipulator as a reference.

An important part of the algorithm is to determine $\boldsymbol{\alpha_d}$ for each link, which is computed with regard to the direction of the flow with respect to the link (the direction of $\boldsymbol{U}$). This requires the unit vector $\boldsymbol{d_{L_i}}$ along the link's longitudinal axis, in the local coordinate frame, as depicted in Figure 5.2a. Most robot manipulators have a rotational joint rotating around the vertical direction. Thus,



(a) Secondary system of orthonormal unit vectors.



(b) Angle of attack of the water flow on a link.

the longitudinal unit vector of the first link becomes $\boldsymbol{d_{L_1}} = \boldsymbol{e_z} = \begin{bmatrix} 0 & 0 & 1 \end{bmatrix}^T$. Building from the first link, it is then possible to determine $\boldsymbol{d_L}$ for the remaining links according to the following deduction

$$\begin{aligned} {}^{i}\boldsymbol{d_{L_i}} &= \boldsymbol{R_0^i} \, {}^{\mathbf{0}}\boldsymbol{d_{L_i}} \\ &= \boldsymbol{R_i^{0T}} \, {}^{\mathbf{0}}\boldsymbol{d_{L_i}} \\ &:\Leftrightarrow \boldsymbol{R_i^T} \, {}^{\boldsymbol{i-1}}\boldsymbol{d_{L_i}} \end{aligned} \tag{5.11}$$

where the rotation matrix property of $\boldsymbol{R_i^{-1}} = \boldsymbol{R_i^T}$ has been applied and $\boldsymbol{R_i}$ is the the description of the orientation in $\boldsymbol{A_i}$ (see (3.3)). The leading subscript is used to underline the local coordinate system of the variable.

The angle of attack α_d can then be computed based on the relation:

$$\cos \alpha_d = |\boldsymbol{u} \cdot \boldsymbol{d_L}| \tag{5.12}$$

where α_d is the acute angle between the axis of the link and the direction of the flow relative to this link, given by the unit vector $\boldsymbol{u} = \boldsymbol{U}/\sqrt{U^2}$.

Referencing the strip theory depicted in Figure 5.1 the drag force can be found for each strip, similar to (5.7). For strip j on link i, the following force calculations are then performed:

$$\boldsymbol{dF_{ij}} = \frac{1}{2}(\rho C_{N_{ij}} U_{ij}^2 D_i dl_i)\boldsymbol{c_{ij}} \tag{5.13}$$

where $\boldsymbol{c_{ij}} = \begin{bmatrix} c_h & c_e & c_d \end{bmatrix}^T$ represents the direction of the resulting drag force on the link (perpendicular to the link's longitudinal axis):

$$c_h = (\boldsymbol{u} \cdot \boldsymbol{h_L})/c, \tag{5.14}$$

$$c_e = (\boldsymbol{u} \cdot \boldsymbol{e_L})/c, \tag{5.15}$$

$$c_d = 0, \tag{5.16}$$

$$c = \arccos \frac{\pi}{2} - \alpha_d \ . \tag{5.17}$$

The resulting torque force on strip j is given by

$$\boldsymbol{dN_{ij}} = \boldsymbol{r_{ij}} \times \boldsymbol{dF_{ij}} \tag{5.18}$$

where $\boldsymbol{r_{ij}}$ is the length between strip j and the center of mass of link i.

In the end, the total force torque over the link length is then found by adding the contributions of all the strips such that

$$\boldsymbol{N_{D_i}} = \sum_{j=1}^{n_{L_i}} \boldsymbol{dN_{ij}} \ . \tag{5.19}$$

where n_{L_i} is the chosen number of strips.

Current Effects

Any current affecting the movements of a robot manipulator can be included in the equations of motion through the relative velocity vector $\boldsymbol{U}$. For an underwater vehicle the effect of a small current has to be considered also in structured environments such as a pool [2]. The current is an external disturbance and can be set to affect all the terms in the dynamics of an underwater vehicle [18], but here it will be assumed to only affect the drag forces on the manipulator. This is essentially due to the low manipulator velocities combined with the directly coupled effect on the drag and its velocity squared dependency. For a robot manipulator, its motion velocities are usually small such that that drag values are almost negligible. However, as the drag forces are dependant on the relative velocity of body and fluid, with current velocities of significant value, the hydrodynamic effects might have a much larger impact on the dynamics of the manipulator system.

Smolt tanks are by design induced with a current in order to create an optimal environment for the fish. An optimal flow domain in culture tanks is vital for fish growth and welfare [60]. With fish present, different recirculation aquaculture systems (RAS) tank was proven to have water rotational velocities of maximum 40 cm/s and minimum of 25-26 cm/s [22]. The tanks also exhibited a relatively uniform water velocity field in the vertical water column. For the computations it is helpful to simplify the current as constant and irrotational. No real flow is irrotational. It is essentially a way to simplify the flow fields and make the calculation of fluid dynamics solvable [19].

Neighboring Links

Even though not taken into consideration in this thesis, it is worth to briefly mention how neighboring links affect the drag on each other. For a robot arm with multiple links, the joint torque values from the drag force on one link is affected by the flow characteristics of all the links that follow up to and including the end-effector. Therefore, the joint angles of the neighboring links will have an effect on the drag force of all the links that are not the last link [31].

5.5 Buoyancy

Buoyancy is the resultant force caused by the hydrostatic pressure acting on a submerged robot arm. When submerged, the links of the arm is affected by hydrostatic forces acting on all surfaces in contact with the water. The forces acting on the sides tend to cancel each other out as they are equal and opposite. However, in the vertical direction, the magnitude of the pressure increases with depth. Thus, due to higher hydrostatic forces pushing from below there is a resultant force in the upward direction called buoyancy. Buoyancy forces work against gravity and are directed upwards. This is the only hydrostatic effect on an immersed manipulator [1].

The value of the buoyancy force is proportional to the volume of the water displaced by the link. For a single link of the manipulator, this means the buoyancy is highly dependant on the volume of its fully or partially immersed body. The buoyancy of a fully immersed link is given by

$$B = \rho g V \tag{5.20}$$

where ρ is the water density, g is the gravitational acceleration and V is the link volume. Similar to how the gravity force acts on the center of mass, the buoyancy force acts in the center of buoyancy.

Calculation the buoyancy force and the equivalent joint torque is done relative to the base frame, with $\boldsymbol{R_i^0}$ describing the orientation [13]. The z-axis of the base frame is assumed stationary, and always has a z-axis in the upward direction. Thus, the buoyancy is of positive value, opposite of gravity found in (4.17). If the center of buoyancy for link i is denoted as $\boldsymbol{r_{b_i}} = \begin{bmatrix} x_b & y_b & z_b \end{bmatrix}^T$ the buoyancy force is given by

$$\boldsymbol{F_{B_i}} = \rho g V_i \begin{bmatrix} \boldsymbol{R_i^0 e_z} \\ \boldsymbol{r_{b_i} R_i^0 e_z} \end{bmatrix} \tag{5.21}$$

where $\boldsymbol{e_z} = \begin{bmatrix} 0 & 0 & 1 \end{bmatrix}^T$ and $\boldsymbol{r_{B_i}}$ is the position vector of the center of buoyancy with respect to the origin of frame i. The total effect of the buoyancy on link i is then found as

$$\boldsymbol{G_{B_i}} = \boldsymbol{J_i^T(q) A_i^0(q) F_{B_i}(q)} \tag{5.22}$$

where $\boldsymbol{J} = \begin{bmatrix} \boldsymbol{J_v} & \boldsymbol{J_\omega} \end{bmatrix}^T$ is the manipulator Jacobian. It is worth mentioning that if the center of buoyancy does not coincide with the center of gravity, the buoyancy forces may add a rotational motion depending on the relative size of the two forces.

Chapter 6

Positional Control of Robot Manipulators

With the ensured derivation of the underwater equations of motion and kinematics for the robot manipulator, designing a motion control algorithm is a legitimate next step. A robot manipulator is essentially a positioning device. The process of positioning the end-effector is decided by the actuator forces in each joint. The end-effector position is therefore a multi-variable problem based on the actuator forces. Coupled with the manipulator control system, is usually the trajectory planner with the ability to design trajectories that generates the reference points for the control system to follow.

The rest of this chapter[1] is presented in a sequential order, opening with the establishment of the input reference values from the trajectory planner (Section 6.1) followed by the control system setup using, with a special focus on Inverse Dynamics Control (Section 6.2). In the last section, a set of alternative control techniques are briefly discussed (Section 6.3).

6.1 Trajectory Planning

The main objective of the trajectory planner is to provide reference values for the position control system. The reference values are often either assigned for a point-to-point motion or a motion made by a sequence of points. If the goal is to go from an initial to a final point, how the trajectory planner gets there, is decided by the technique for generating the trajectory. The same concept applies when provided a sequence of points; the only difference is that multiple points result in shorter distances for each motion. Thus, when the circumstances demand more control of the motion, e.g., to avoid obstacles, a solution can be to add more waypoints to the path. This is obviously more computationally demanding. Moreover, the extra reference values inherently affect the computational demand in the control system as well. Note that trajectory planning is distinct from path planning in that it is parametrized by time.

Trajectory planning is not just about going from A to B, but to a greater extent about how to get there. One of the first design choices is regarding whether to generate a trajectory in the joint-space or configuration-space. The configuration-space suggests the waypoints are described by the pose of the manipulator. At

[1]This chapter is completely new, and not based on specific chapters in the specialization project thesis.

the same time, the joint-space configuration implies waypoints described directly by the joint positions (angles or displacements depending on the joint type). In terms of model complexity, compared to the joint-space, the configuration-space method is much more demanding due to the computation of the inverse kinematics for every change in the manipulator's joint positions, especially for optimization-based inverse kinematics solvers. The configuration-space solution tend to provide a smoother and more "natural" motion from A to B. [6].

A next step towards a constructing trajectories between two configurations is to specify the dynamic aspects of the motions. For a generated trajectory there are always constraints on the motion. The starting- and end-point are the most obvious constraints, but there are more. For example constraints on the initial and final velocities, and accelerations. For the end-point, the final velocity is typically set to zero. However, if a waypoint is the middle of a sequence of points, zero velocity is not necessarily needed. These constraints are usually dependant on the specific task given to the manipulator. Nevertheless, there are infinitely many trajectories that will satisfy a finite number of constraints on the waypoints.

There are various techniques to interpolate the joint configuration or pose over time. Due to the infinite number of possible, it is common practice to choose trajectories from a finitely parameterizabel family [57]. One such family is the polynomial approach where a polynomial of degree $n-1$ generates a smooth curve satisfying n constraints based on n independent coefficients that are to be chosen. One example is the *quintic* polynomial trajectory which require a fifth order polynomial to satisfy six constraints [57]. Other employed trajectory methods are:

The trapezoidal velocity method [57][6] (also called the LSPB method) based on linear segments with parabolic blends with a trapezoidal velocity profile.

The cubic splines method [46] based on optimized cubic spline interpolation.

The B-splines method based on fifth-order B-spline interpolation [16].

The remaining question is about which trajectory planner to choose. The answer to this is typically dependant on the type of task the robot manipulator is set to do. Still, a rather generalized comparison of three promising methods (quintic polynomials, cubic splines and fifth-order B-spline) was performed by [17]. The conclusion was that the fifth-order B-splines method yielded the optimal results with the least amount of jerk. The jerk value is defined as the derivation of acceleration. Lower jerk values implies smoother movements by the manipulator.

Whichever trajectory planer is chosen, its output is typically represented as $\boldsymbol{q}_d$, $\dot{\boldsymbol{q}}_d$ and $\ddot{\boldsymbol{q}}_d$ because they represent the *desired* joint variable. A more detailed description of how a control system may be designed to use this reference value, follows in the next section.

6.2 Control System

The control system for the joint space control problem allows for the actual manipulator motion $\boldsymbol{q}$ to track the reference input $\boldsymbol{q_d}$. Similar to the trajectory planner, the position control system can be based on either the joint space the the configuration space. In this section, the main focus will be on the joint space control. To successfully control a manipulator, the time history of the generalized forces $\boldsymbol{\tau}$ on the joint actuators has to be determined as to guarantee execution of the given task while satisfying transient and stead-state requirements [53]. The generalized forces are contingent on the manipulator's dynamic properties in order to know how much force has to be exerted for the desired position. Too little force and the manipulator will be slow and sluggish; too much force and the arm will crash into objects in its path and may end up oscillating about its desired position, or become unstable. In order to track a desired joint trajectory accurately, the controller has to be capable of removing the potential disturbances as well.

For a control scheme to be beneficial for an underwater manipulator it must be able to cope with the highly nonlinear, time varying and uncertain dynamics as presented in Chapter 4 and 5. Robot manipulators are inherently nonlinear systems. Hence, linear PID or PD control methods are not suitable mainly due to their lack of effectiveness and robustness to the uncertainties and disturbances to be accounted for. The result is poor dynamic accuracy when trajectory tracking comes into play and the dynamic performance of the manipulator varies according to its configuration [25]. To have any success with a PID controller, the system needs a decentralized control scheme. This type of manipulator control strategy is based on individual control of each joint axis. Another way to control an underwater manipulator is to use a nonlinear *centralized* control scheme, which takes advantage of the dynamics model by eliminating the nonlinearities [53]. One conventional method to overcome the nonlinearities of the system is the Inverse Dynamics Controller (IDC) method [57][53]. This method has also been specifically adopted for underwater robot manipulators [50].

6.2.1 Inverse Dynamics Control

IDC is a nonlinear control technique which provides trajectory tracking by calculating the required joint actuator torques to achieve a given trajectory [53][57] .The main idea is to linearize a nonlinear system using a nonlinear feedback based on the inverse dynamics. The result is an **exact** linearization of the robot dynamics [53]. This is possible due to how the manipulator dynamics are arranged. Because the inertia matrix $\boldsymbol{M}$ is of full rank and invertible for any manipulator configuration[57]. Considering the previously covered underwater dynamic equations of an n-link robot (5.2):

$$\boldsymbol{u} = \boldsymbol{M}(\boldsymbol{q})\ddot{\boldsymbol{q}} + \boldsymbol{n}(\boldsymbol{q}, \dot{\boldsymbol{q}}) \tag{6.1}$$

where $\boldsymbol{u} = \boldsymbol{\tau}$ is the control law for the actuators, $\boldsymbol{M}$ is the sum of the manipulator inertia $\boldsymbol{M_m}$ and the added-mass inertia $\boldsymbol{M_a}$, and $\boldsymbol{n(q, \dot{q})}$ is the remaining terms. To establish the linear closed loop system, the nonlinear control law is chosen as [57]

$$\boldsymbol{u} = \boldsymbol{M}(\boldsymbol{q})\boldsymbol{a_q} + \boldsymbol{n}(\boldsymbol{q}, \dot{\boldsymbol{q}}) \ . \tag{6.2}$$

A proposed controller design based on the IDC fundamentals, is depicted in Figure 6.1. This control architecture consists of of an inner and outer loop. The function of the inner loop is to obtain the linear and decoupled input/output relationship, whereas the outer loop is there to stabilize the overall system [53].

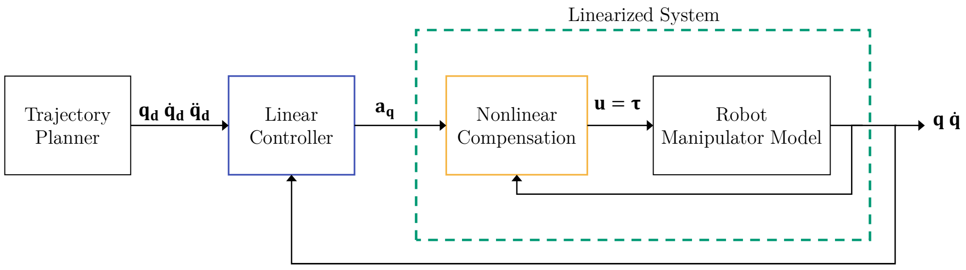


Figure 6.1: Basic inner/outer loop Inverse Dynamics Control architecture.

The linearized system is then described by the new input $\ddot{\boldsymbol{q}} = \boldsymbol{a_q}$ which is chosen to be

$$\boldsymbol{a_q}(t) = -\boldsymbol{K_0}\boldsymbol{q}(t) - \boldsymbol{K_1}\dot{\boldsymbol{q}}_d + \boldsymbol{r}(t) \ . \tag{6.3}$$

$\boldsymbol{r}(t)$ is the chosen reference input consisting of the desired joint angles $\boldsymbol{q}_d$, velocities $\dot{\boldsymbol{q}}_d$ and accelerations $\ddot{\boldsymbol{q}}_d$:

$$\boldsymbol{r}(t) = \ddot{\boldsymbol{q}}_d(t) + \boldsymbol{K_0}\boldsymbol{q}_d(t) + \boldsymbol{K_1}\dot{\boldsymbol{q}}_d \ . \tag{6.4}$$

Finally, with the proceeding tracking error $\boldsymbol{e}(t) = \boldsymbol{q} - \boldsymbol{q_d}$, the complete form of the input for the combined system is given by

$$\boldsymbol{a_q}(t) = \boldsymbol{K_0}(\boldsymbol{q}_d - \boldsymbol{q}) + \boldsymbol{K_1}(\dot{\boldsymbol{q}}_d - \boldsymbol{q}) + \ddot{\boldsymbol{q}}_d. \tag{6.5}$$

The resulting system is indeed linear, as well as decoupled [57]. A further detailed version controller design depicting the detailed computations, is shown in Figure 6.2.

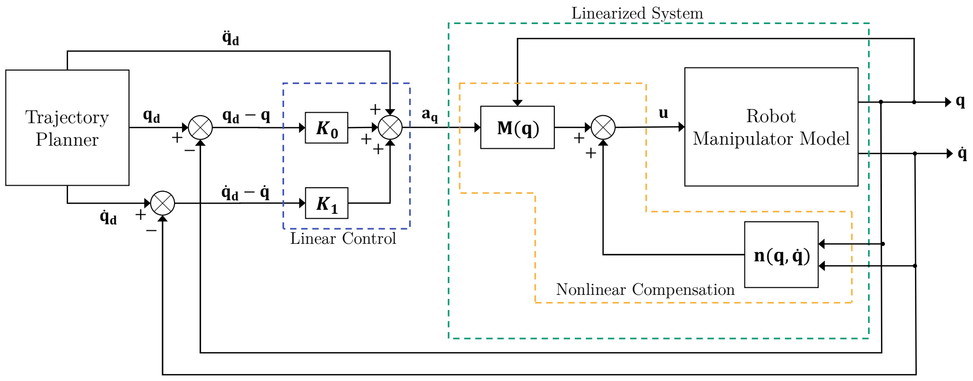


Figure 6.2: Expansive and detailed Inverse Dynamics Control architecture.

Performance

The inverse dynamics approach is a good foundation and beneficial for further work with position control on a manipulator, including more advanced methods. From a control viewpoint. the IDC approach is described as *attractive* [53], *extremely important* [57], and even the *perfect* [23] approach for robot applications. For a manipulator, the technique is attractive due to the resulting linearized and decoupled system which allows for well-known linear control methods. Additionally, from already determined dynamic equations, the process of setting up the IDC system is relatively straightforward even for a complex manipulator. Accordingly, for manipulator systems with detailed information about the model parameters involved, IDC is a functional robust controller for which variations in robot parameters do not affect performance [4]. In other words, it is capable of removing disturbances and other uncertainties for a well modeled system.

Even though the IDC system has multiple advantages, the IDC approach also has its limitations, particularly for unpredictable physical systems. The technique of nonlinear compensation and decoupling is based on the assumption of perfect cancellation of the dynamic terms. Perfect cancellation is rarely a luxury when talking about the relationship between the modelled and the real-world system. This is mainly due to mathematical suppositions and uncertainties. Therefore, the implementation of the IDC first and foremost requires that the parameters related to the system dynamics are accurately known. Additionally, a complete form of the equations of motion is to be known, considering any external disturbances of significant value. These conditions are difficult to achieve and thus raising questions regarding the robustness of the control system. Specifically, regarding imperfect knowledge of manipulator mechanical parameters or external disturbance parameters, and model dependence on end-effector payloads is not exactly known and thus not perfectly compensated. Apart from the model specific limitations, the IDC has two major faults in regard to the control system response. One is that the transient state response of the system is insufficient, and secondly the overall response of the system is very slow.

From an implementation viewpoint, as for any control technique, there is a list of advantages and disadvantages to be considered. In modern control theory, a nonlinear control system can be solved using several available control approaches. Adopting the IDC system is still a candid method to observe the manipulator system, and perhaps as a basis of choice for increasingly complicated control schemes.

6.3 Alternative Control Algorithms

The IDC control scheme lacks the adequate robustness because it is sensitive to time varying and uncertain model parameters and external disturbances. The parameters needs to be known exactly. There are however alternative control algorithms with focus on a robust and adaptive control to maintain performance despite parametric uncertainties [53]. A robust controller is a fixed controller, static or dynamic, designed to satisfy performance specifications over a given range of uncertainties whereas an adaptive controller incorporates a type of on-line parameter estimation [57]. These two types of controllers does well at counteracting different types of uncertainties, and therefore can be combined for improved results. A robust controller counteracts the uncertainties regarding the model approximations and an adaptive controller will try to adapt the model to the real underwater manipulator dynamics.

In literature there are various control algorithms proposed for underwater manipulators specifically. A robust controller is proposed by [32] combining a computed torque controller and a sliding mode controller (SMC) with a neural network controller acting as a compensator, maintaining the control performance when the initial uncertainty assumptions cease to be valid. Another example is [21] which involves a disturbance observer controller that simplifies the complex model into a system with disturbance error, and a non-regressor based adaptive controller designed according to the new model. In [55] there are summaries of even more examples of proposed control algorithms for underwater manipulators.

Part II

Bravo 7 Control Framework Derivation

Chapter 7

Mathematical Modelling

This chapter[1] involves the mathematical modelling derivation of the Bravo 7 based on the material in Part II. Section 7.1 describes important information details provided by Blueprint Lab. Then Section 7.2 and 7.3 presents the forward and inverse kinematics solutions. In the end Section 7.4 describes both the land-based and underwater dynamics models.

7.1 Bravo Reach 7

Blueprint Lab is the company behind the Reach Bravo 7, a robot arm released in 2020 [28] designed to conduct inspection, maintenance and repair tasks typically reserved for human divers in subsea operations [30]. This section involves a description of available information about the Bravo 7 used in this thesis as well as some schematics used in the mathematical modelling. The information available from Blueprint Lab is added to the delivery file of the this master thesis, see Appendix B for more information. The documentation was of particular interest as it included the kinematic, dynamic and hydrodynamic properties.

7.1.1 System Description

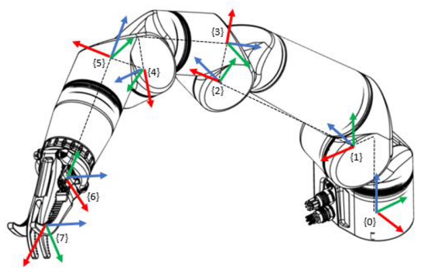
The Bravo 7 is described as a "*7-Function*" robot manipulator, which is another way of saying it has seven DoF. More specifically, the Bravo 7 has a total of six revolute joints making up the first six DoF. The seventh DoF makesup the end-effector and its ability to grasp objects or perform other tasks with different tools attached. The schematics in figure 5.1 shows the position of the six joints and their possible rotational movements. In figure 7.1a coordinate frames are added at each joint ($\{0\} - \{5\}$). The last two coordinate frames $\{6\}$ and $\{7\}$ describes the beginning and end of the end-effector tool. All the actuators includes electric motors.

Some important features from the datasheet related to operating in smolt tanks:

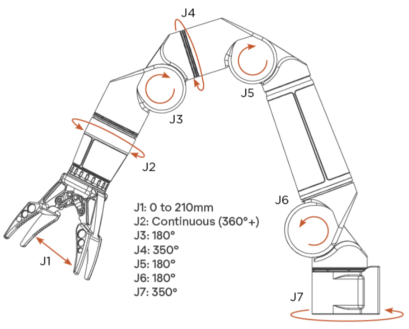
1. **All-electric motors, zero oil:** This is important to avoid the risk of possible oil contamination into the smolt tank.
2. **Task specific end-effectors:** Allows the ability to change end-effector based on the task at hand, e.g, for picking up fish or washing the tank walls.

[1]Sections 7.1, 7.2, 7.3 and 7.4.1 are based on the specialization project thesis, but the content is improved and elaborated. Section 7.4.2 is new.

3. **High mobility and compactness:** Is important for work in a dynamically changing environment such as smolt tanks. Additionally, a smaller size is advantageous to reduce the change of scaring the fish (as mentioned in the introduction).
4. **Specifically designed for UUVs:** Due to a reach of only 0.9 meters, attaching the Bravo 7 to an UUVs will be necessary to make it useful in a big smolt tank.



(a) Schematics of Bravo 7 with added coordinate frames (from Bravo 7 documentation, see Appendix B).



(b) Schematics of Bravo 7 showing possible joint movements (from Bravo 7 data sheet, see Appendix B))

Figure 7.1: Two different schematics of Reach Bravo 7 from Blueprint Lab.

7.1.2 Denavit Hartenberg Parameters

The documentation from Blueprint Lab provides a DH-table consisting of the DH-parameters. By examining how the the parameters are presented in the table, it may be confirmed that it is based on the distal variant version of the DH-convention. This is is an important observation as to avoid confusion when working with the forward kinematics. Another thing to take notice of, is the added constant values to the three first joint variables. The constants are added for the chosen zero-configuration, meaning the configuration where all joint variables are set equal to zero. The zero-configuration is visualized using the Peter Corke Robotics Toolbox for MATLAB [7] in Figure 7.2.

7.2 Forward Kinematics

The computation of the forward kinematics for the Bravo 7 is based on the material from Section 3.1.

7.2.1 Kinematic Diagram and Denavit Hartenberg Table

Following the preliminary rules and frame rules of the DH-convention in Section 3.1.2, a kinematic schematic of the Bravo 7 is depicted in Figure 7.3. The

Link	d (mm)	θ	a (mm)	α
0	107.4	$\theta_0 + \pi$	46.0	$\pi/2$
1	0.0	$\theta_1 - \pi/2 + \theta_a$	293.6	0.0
2	0.0	$\theta_2 - \pi/2 - \theta_a$	40.8	$-\pi/2$
3	-160.0	θ_3	40.8	$-\pi/2$
4	0.0	θ_4	40.8	$-\pi/2$
5	-223.5	θ_5	0.0	$\pi/2$
6	0.0	$-\pi/2$	120.0	0.0

Table 7.1: Distal variant DH-table for Bravo 7 where $\theta_a = tan^{-1}(\frac{5.2}{293.55})$ (from Bravo 7 documentation, see Appendix B)

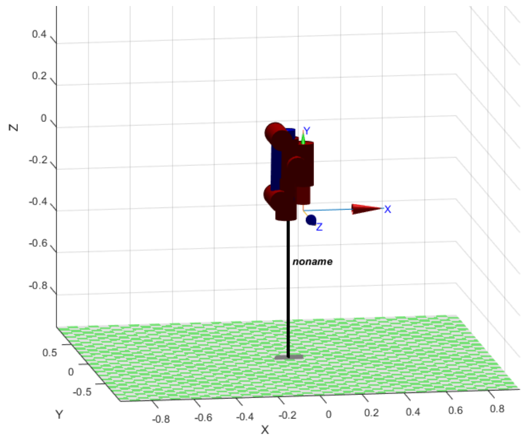
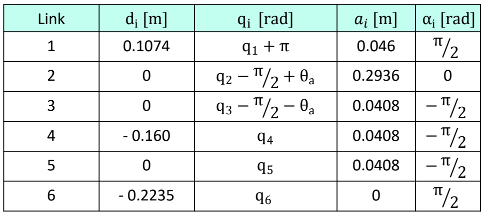


Figure 7.2: Bravo 7 zero-configuration visualization using Peter Corke Robotics Toolbox [7].

kinematic schematic was created based on the DH-table 7.2. The DH-table in table 7.2 is similar to table 7.1, but with a couple of alterations. The numbering of the links are not the same, i.e. link 0 is changed to link 1 and so on. This is according to personal preference and is the same convention as the one used in [57]. Another alteration is regarding the joint variables. The joint variables are set as q_i instead of θ_i .



Link	d_i [m]	q_i [rad]	a_i [m]	α_i [rad]
1	0.1074	$q_1 + \pi$	0.046	$\pi/2$
2	0	$q_2 - \pi/2 + \theta_a$	0.2936	0
3	0	$q_3 - \pi/2 - \theta_a$	0.0408	$-\pi/2$
4	- 0.160	q_4	0.0408	$-\pi/2$
5	0	q_5	0.0408	$-\pi/2$
6	- 0.2235	q_6	0	$\pi/2$

Table 7.2: Altered DH table directly based on figure 7.3, without tool length.

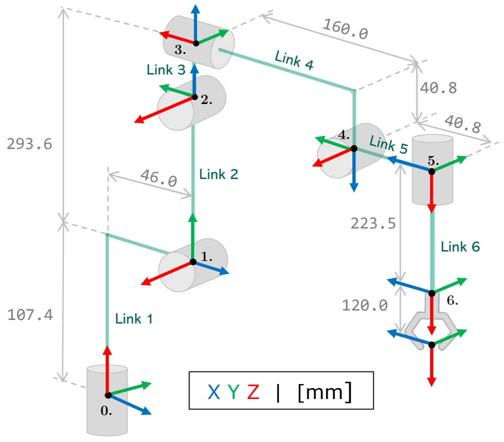


Figure 7.3: Kinematic schematic of Reach Bravo 7 in its zero-angle configuration (not made up to scale).

7.2.2 Forward Kinematics Solution

As mentioned in section 3.1, when the DH-table is complete it is possible to form the forward kinematics solution from the base frame to the far end of any of the links directly from the table. The solution comes in the form of a transformation matrix. Table 7.2 does not include the end-effector (of length 120mm) which means the forward kinematics solution from link 0 (the base) to link 6 becomes $\boldsymbol{T_6^0}(q)$. The joint variables are defined based on Table 7.2 to be $\boldsymbol{q} = [q_1, q_2, q_3, q_4, q_5, q_6]^T$.

$$\boldsymbol{T_6^0}(\boldsymbol{q}) = \boldsymbol{A}_1\boldsymbol{A}_2\boldsymbol{A}_3\boldsymbol{A}_4\boldsymbol{A}_5\boldsymbol{A}_6 = \begin{bmatrix} r_{11} & r_{12} & r_{13} & x_6^0 \\ r_{21} & r_{22} & r_{23} & y_6^0 \\ r_{31} & r_{32} & r_{33} & z_6^0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{7.1}$$

The transformation matrix in (7.1) is too big and complex to be added here, even in its simplest symbolic form. $\boldsymbol{A}_1 \cdots \boldsymbol{A}_6$ are the homogeneous transformation matrices expressing the pose of each frame with respect to its preceding frame. The matrices are in their simplest symbolic form. In the terms of table 7.2, the symbols are simplified as $c_i = cos(\theta_i)$ and $s_i = sin(\theta_i)$ and the variable values can be inserted directly from the table. See B for the forward kinematics Matlab function file.

7.3 Inverse Kinematics

Due to the structure of the Bravo 7 robot, the solution to the inverse kinematics problem is a complex and comprehensive affair. The geometrical structure of the

Bravo 7 is complicated compared to other robot arm configurations because of its non-straight links. Furthermore, it has no spherical wrist and . As mentioned in section 3.2, a spherical wrist simplifies the derivation of the inverse kinematics and there is a lower probability that an analytical solution exists.

Even though a numerical solution was most likely the only possibility for the Bravo 7 manipulator, a geometrical an algebraic solution were attempted. For the geometrical method, finding a solution for the three first joints was possible, but challenging due to the the non-straight form of link 1. Going further than three joints was deemed unattainable. An algebraic solution was attempted by using the IKBT (Inverse Kinematics Behavioural Tree) method, which is a framework created for automatic inverse kinematics solving [65]. However, a solution were only found for a maximum of four joints. The framework is also made for manipulator with a spherical wrist, thus making the validity of the solution questionable. In the end a numerical optimization approach was chosen which involved the same mathematical formulation as presented in Section 3.2.2. The Matlab function used the `fsolve` function from the *Optimization Toolbox* [45] with the *Levenberg-Marquardt* algorithm, a nonlinear least-squares algorithm solver [59].

7.4 Dynamics Model

This section explains the procedure that was used to develop the Matlab code for the dynamics model both with and without underwater effects. The base model is presented first, and then the underwater effects are added.

7.4.1 Land-based Dynamics

The Euler-Lagrange approach for determining the land-based dynamics model was derived in Chapter 4. Section 7.2 then showed the method for calculating the transformation matrix with forward kinematics for Bravo 7. The transformation matrix and some additional parameters from the Blueprint Lab documentation were used for finding a symbolic representation of the land-based dynamic model in this section. All the required parameter values was found in Table 7.3. Because the table is from the Blueprint Lab documentation, the link numbering is shifted compared to Figure 7.3. Thus link i in the table is equal to link $i+1$ in the computations.

Kinetic Energy

The kinetic energy of Bravo 7 was computed by first determining the inertia matrix using the velocity Jacobians, inertia tensor, rotation matrix, joint variable and mass of all the links. The mass and inertia tensors are constants found in Table 7.3. As the robot only consists of revolute joints, the velocity Jacobians were computed the same way for every joint, see equation (4.3) and (4.4). By

Link	Mass (kg)	COM (mm)	I ($kg.mm^2$)
0	1.25	$\begin{pmatrix} -18 & -4 & -1 \end{pmatrix}$	$\begin{pmatrix} 2108 & 182 & -15 \\ 182 & 2573 & -21 \\ -15 & -21 & 3483 \end{pmatrix}$
1	1.55	$\begin{pmatrix} 17 & -7 & 57 \end{pmatrix}$	$\begin{pmatrix} 11442 & -484 & 3405 \\ -484 & 12980 & -1265 \\ 3405 & -1265 & 3202 \end{pmatrix}$
2	1.98	$\begin{pmatrix} 117 & 15 & 6 \end{pmatrix}$	$\begin{pmatrix} 3960 & 4200 & 3204 \\ 4200 & 69099 & -24 \\ 3204 & -24 & 70450 \end{pmatrix}$
3	1.14	$\begin{pmatrix} 22 & -29 & 1 \end{pmatrix}$	$\begin{pmatrix} 3213 & -1548 & -31 \\ -1548 & 2327 & 6 \\ -31 & 6 & 4340 \end{pmatrix}$
4	1.14	$\begin{pmatrix} 18 & 6 & -117 \end{pmatrix}$	$\begin{pmatrix} 21232 & 330 & -3738 \\ 330 & 22252 & -1278 \\ -3738 & -1278 & 2054 \end{pmatrix}$
5	1.03	$\begin{pmatrix} 20 & -24 & 1 \end{pmatrix}$	$\begin{pmatrix} 2430 & -1144 & -40 \\ -1144 & 2026 & 11 \\ -40 & 11 & 3330 \end{pmatrix}$
6	1.04	$\begin{pmatrix} 0 & 0 & -128 \end{pmatrix}$	$\begin{pmatrix} 22359 & 1 & -19 \\ 1 & 22363 & 15 \\ -19 & 15 & 936 \end{pmatrix}$
7	0.47	$\begin{pmatrix} 28 & -1 & 0 \end{pmatrix}$	$\begin{pmatrix} 244 & -12 & 0 \\ -12 & 1130 & 1 \\ 0 & 1 & 1178 \end{pmatrix}$

Table 7.3: Inertial properties for Bravo 7 from the Blueprint Lab documentation (link 7 is for interlocking jaws in the closed position)

reference to (3.1), (3.2) and (4.5), the velocity Jacobian and rotation matrix were found from the transformation matrix for each link as presented below.

Link 1

The transformation matrix $\boldsymbol{T}_1^0$ is used to find:

$$\boldsymbol{J}_{v_1} = \boldsymbol{z}_0 \times (\boldsymbol{o}_6 - \boldsymbol{o}_0) \tag{7.2}$$

$$\boldsymbol{J}_{\omega_1} = \boldsymbol{z}_0 \tag{7.3}$$

$$\boldsymbol{R}_1 = \boldsymbol{R}_1^0 \tag{7.4}$$

$\boldsymbol{o}_0 = [0, 0, 0]^T$ and $\boldsymbol{z}_0 = [0, 0, 1]^T$.

Link 2

The transformation matrix $\boldsymbol{T}_2^0$ is used to find:

$$\boldsymbol{J}_{v_2} = \boldsymbol{z}_1 \times (\boldsymbol{o}_6 - \boldsymbol{o}_1) \tag{7.5}$$

$$\boldsymbol{J}_{\omega_2} = \boldsymbol{z}_1 \tag{7.6}$$

$$\boldsymbol{R}_2 = \boldsymbol{R}_2^0 \tag{7.7}$$

Link 3

The transformation matrix $\boldsymbol{T}_3^0$ is used to find:

$$\boldsymbol{J}_{v_3} = \boldsymbol{z}_2 \times (\boldsymbol{o}_6 - \boldsymbol{o}_2) \tag{7.8}$$

$$\boldsymbol{J}_{\omega_3} = \boldsymbol{z}_2 \tag{7.9}$$

$$\boldsymbol{R}_3 = \boldsymbol{R}_3^0 \tag{7.10}$$

Link 4

The transformation matrix $\boldsymbol{T}_4^0$ is used to find:

$$\boldsymbol{J}_{v_4} = \boldsymbol{z}_3 \times (\boldsymbol{o}_6 - \boldsymbol{o}_3) \tag{7.11}$$

$$\boldsymbol{J}_{\omega_4} = \boldsymbol{z}_3 \tag{7.12}$$

$$\boldsymbol{R}_4 = \boldsymbol{R}_4^0 \tag{7.13}$$

Link 5

The transformation matrix $\boldsymbol{T}_5^0$ is used to find:

$$\boldsymbol{J}_{v_5} = \boldsymbol{z}_4 \times (\boldsymbol{o}_6 - \boldsymbol{o}_4) \tag{7.14}$$

$$\boldsymbol{J}_{\omega_5} = \boldsymbol{z}_4 \tag{7.15}$$

$$\boldsymbol{R}_5 = \boldsymbol{R}_5^0 \tag{7.16}$$

Link 6

The transformation matrix $\boldsymbol{T}_6^0$ is used to find:

$$\boldsymbol{J}_{v_6} = \boldsymbol{z}_5 \times (\boldsymbol{o}_6 - \boldsymbol{o}_5) \tag{7.17}$$

$$\boldsymbol{J}_{\omega_6} = \boldsymbol{z}_5 \tag{7.18}$$

$$\boldsymbol{R}_6 = \boldsymbol{R}_6^0 \tag{7.19}$$

With the computed Jacobians ready, the inertia matrix for each link was computed as in (4.8), making it possible to find the kinetic energy by using (4.11)

Potential Energy

The computation of potential energy only depends on the mass, the center of mass and the gravitational acceleration. The coordinate vector for the center of mass for every link was found in Table 7.3. This coordinate vector is in reference to the link frame and not the reference frame. Thus, the coordinate vector $\boldsymbol{r}_{c_i}$ in 4.15 had to be transformation from the link frame to the base frame as done below.

$$\boldsymbol{r}_{c_1} = \boldsymbol{r}^0_{c_1} = \boldsymbol{R}^0_1 \boldsymbol{r}^1_{c_1} \tag{7.20}$$

$$\boldsymbol{r}_{c_2} = \boldsymbol{r}^0_{c_2} = \boldsymbol{R}^0_2 \boldsymbol{r}^2_{c_2} \tag{7.21}$$

$$\boldsymbol{r}_{c_3} = \boldsymbol{r}^0_{c_3} = \boldsymbol{R}^0_3 \boldsymbol{r}^3_{c_1} \tag{7.22}$$

$$\boldsymbol{r}_{c_4} = \boldsymbol{r}^0_{c_4} = \boldsymbol{R}^0_4 \boldsymbol{r}^1_{c_4} \tag{7.23}$$

$$\boldsymbol{r}_{c_5} = \boldsymbol{r}^0_{c_5} = \boldsymbol{R}^0_5 \boldsymbol{r}^5_{c_5} \tag{7.24}$$

$$\boldsymbol{r}_{c_6} = \boldsymbol{r}^0_{c_6} = \boldsymbol{R}^0_6 \boldsymbol{r}^1_{c_6} \tag{7.25}$$

The potential energy was then computed as in (4.15) where $\boldsymbol{g}_0 = [0, 0, -9.81]^T$.

Land-based Equations of Motion

The symbolic equation for the equations of motion (4.1) with $\boldsymbol{q}$, $\dot{\boldsymbol{q}}$ and $\ddot{\boldsymbol{q}}$ as input variables, required the inertia matrix $\boldsymbol{M}$, centrifugal and Coriolis terms matrix $\boldsymbol{C}$, and gravity vector $\boldsymbol{G}$ to be computed first. The inertia matrix was calculated as a result of the kinetic energy, which leaves $\boldsymbol{C}$ and $\boldsymbol{G}$.

Calculating the gravity vector is straight forward. Every vector input is a partial derivative of the equation for potential energy with respect to the general coordinates $\boldsymbol{q}$. For bravo 7 this means that the gravity vector becomes

$$\boldsymbol{G} = \begin{bmatrix} \frac{\partial \mathcal{P}}{\partial q_1} \\ \frac{\partial \mathcal{P}}{\partial q_2} \\ \frac{\partial \mathcal{P}}{\partial q_3} \\ \frac{\partial \mathcal{P}}{\partial q_4} \\ \frac{\partial \mathcal{P}}{\partial q_5} \\ \frac{\partial \mathcal{P}}{\partial q_6}. \end{bmatrix} \tag{7.26}$$

The centrifugal and Coriolis terms matrix is not as straight forward to calculate. It requires the Christoffel symbols to be found for all the links. The element of row k and column j of matrix $\boldsymbol{C}$ is found according to (4.24).

$$C_{kj} = \frac{1}{2}\sum_{i=1}^{6}\left\{\frac{\partial D_{kj}}{\partial q_i} + \frac{\partial D_{ki}}{\partial q_j} - \frac{\partial D_{ij}}{\partial q_k}\right\}\dot{q}_i \tag{7.27}$$

With $\boldsymbol{G}$ and $\boldsymbol{C}$ readily calculated, the left side of the equations of motion (4.1) is completed

7.4.2 Underwater Dynamics

The computation of the underwater dynamics for Bravo 7 was done based on the material introduced in Chapter 6 with the assumptions listed in Section 5.1.

Added Mass Effects

The added mass for link i was computed as in equation (5.4), with the length and mass for the approximated circular cylinders as the only link specific inputs. All links were assumed to have a diameter of $D = 80$ mm based on the information in the Bravo 7 data sheet. The water density was set to $\rho = 1000 kg/m^3$. Furthermore, the link lengths were based on the DH-table where $L_1 = 107.4$ mm, $L_2 = 293.6$ mm, $L_3 = 40.8$ mm, $L_4 = 160.0$ mm, $L_6 = 40.8$ mm and $L_6 = 223.5$ mm. At last, the mass of each link was found in Table 7.3. When the added mass matrix $\boldsymbol{M_{A_i}}$ was determined for each link, the total added mass effect on the inertia of the Bravo 7 was found as

$$\boldsymbol{M_A} = \sum_{i=1}^{6}\boldsymbol{M_{A_i}} \quad . \tag{7.28}$$

Drag Forces

The drag force computations were based on the presented material in Section 5.4. As previously indicated, the drag force is a complex variable to model, meaning the numerical computation time during simulation had to be taken into consideration. The computation time is very much dependent on the number of strips used as well [33]. The simulation time of the Bravo 7 were already computationally demanding, without any underwater effects considered. Therefore, the drag forces were modeled assuming a constant drag coefficient of $C_D = 1.1$ which is a typical value for cylinders [38], and with the number of strips set to $n_{L_i} = 10$. The drag force was then computed following the algorithm in Appendix A.1. The current velocity U_F was set as a variables that allowed for simulations of varying conditions. The output of the algorithm was

$$\boldsymbol{N_{D_i}} = \sum_{j=1}^{10}\boldsymbol{N_{D_i}} \tag{7.29}$$

where $\boldsymbol{N_{D_i}} = \begin{bmatrix} N_{D_x} & N_{D_y} & N_{D_z} \end{bmatrix}_i^T$ consists of its vector force components according to the local link frame. Due to the equations of motion in (5.2) being simplified as a total force on each joint, the resulting drag force was converted into its resultant force torque:

$$N_{D_i} = \sqrt{N_{D_x}^2 + N_{D_y}^2 + N_{D_z}^2} \tag{7.30}$$

Hence, the addition to the equations of motion for the Bravo 7 manipulator was set to

$$\boldsymbol{N_D} = \begin{bmatrix} N_{D_1} & N_{D_2} & N_{D_3} & N_{D_4} & N_{D_5} & N_{D_6} \end{bmatrix}^T \tag{7.31}$$

Buoyancy

The computation of the buoyancy forces were based on the presented material in Section 5.5. The general buoyancy force for the links of the fully immersed Bravo 7 was calculated as

$$B_i = \rho g_z V_i \ . \tag{7.32}$$

The water density ρ was the same as used for the added mass and the gravitational acceleration was set to $g_z = 9.81 m/s^2$. The volume was the only link specific value, and was provided by the Bravo 7 documentation in Table 7.4.

Link	**Volume (L)**	**COB (mm)**
0	0.72	$\begin{pmatrix} -18 & -3 & -3 \end{pmatrix}$
1	0.60	$\begin{pmatrix} 27 & -11 & 92 \end{pmatrix}$
2	1.94	$\begin{pmatrix} 145 & 35 & -1 \end{pmatrix}$
3	0.47	$\begin{pmatrix} 33 & -43 & -7 \end{pmatrix}$
4	0.51	$\begin{pmatrix} 20 & 12 & -140 \end{pmatrix}$
5	0.43	$\begin{pmatrix} 33 & -38 & -8 \end{pmatrix}$
6	0.48	$\begin{pmatrix} 0 & 0 & -152 \end{pmatrix}$
7	0.16	$\begin{pmatrix} 28 & 1 & 0 \end{pmatrix}$

Table 7.4: Hydrodynamic properties for Bravo 7 from the Blueprint Lab documentation.

To calculate the specific buoyancy force relative to the base frame, in the z-direction, both the rotation matrix and the center of buoyancy of each link was required. The rotation of each link was previously found in the forward kinematics solution (Section 7.2.2), where the rotation matrix for link i $\boldsymbol{R_i^0}$ is from $\boldsymbol{T_i^0}$. The center of buoyancy $\boldsymbol{r_b}$ is given by the Bravo 7 documentation, see Table 7.4. The calculation of the buoyancy force for each link was then computed as according to (5.21) and the equivalent joint force torque according to (5.22). Similar to the drag forces, the resulting buoyancy forces for each link was set to the resultant force torque value:

$$G_{B_i} = \sqrt{G_{B_x}^2 + G_{B_y}^2 + G_{B_z}^2} \tag{7.33}$$

At last, the addition to the equations of motion for the Bravo 7 manipulator was set to

$$\boldsymbol{G_B} = \begin{bmatrix} G_{B_1} & G_{B_2} & G_{B_3} & G_{B_4} & G_{B_5} & G_{B_6} \end{bmatrix}^T \tag{7.34}$$

Equations of Motion with Added Underwater Effects

Setting up the equations of motion with added underwater effects was done according to (5.2). Due to the added mass Coriolis and centripetal terms being neglected, only the added mass inertia matrix, the drag forces and the buoyancy forces were added to land-based model for form the underwater dynamics model. All the mentioned matrices were computed already, thus resulting in a complete dynamics model on the following form:

$$\boldsymbol{M}(\boldsymbol{q})\ddot{\boldsymbol{q}} + \boldsymbol{C}(\boldsymbol{q}, \dot{\boldsymbol{q}})\dot{\boldsymbol{q}} + \boldsymbol{N}(\boldsymbol{q}, \dot{\boldsymbol{q}}) + \boldsymbol{G}(\boldsymbol{q}) = \boldsymbol{\tau} \tag{7.35}$$

where

$$\begin{aligned} \boldsymbol{M} &= \boldsymbol{M_m} + \boldsymbol{M_A} \\ \boldsymbol{C} &= \boldsymbol{C_m} \\ \boldsymbol{G} &= \boldsymbol{G_m} + \boldsymbol{G_B} \\ \boldsymbol{N} &= \boldsymbol{N_D} \end{aligned}$$

The Matlab script for setting up the symbolic expression is added among the delivery files, see Appendix B.

Chapter 8

Simulation Setup

8.1 Inverse Dynamics Control in Simulink

To perform the positional control of the Bravo 7, the IDC was chosen to track the desired trajectories. Simulations for testing the system performance of the IDC system with and without added underwater effects on Bravo 7 was performed in Simulink [9]. An image of the Simulink setup is provided in Figure 8.1 and the Simulink file is among the delivery files (see Appendix B). The setup is very similar to the one presented in Figure 6.2 with similar color codes to enhance the system setup correlation to the theoretical model.

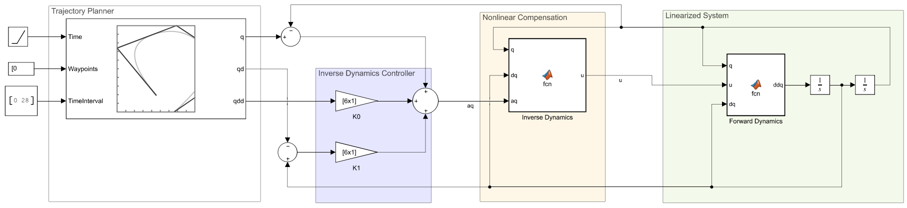


Figure 8.1: Inverse Dynamics Control setup in Simulink.

8.1.1 Trajectory Planner

For the trajectory planner, various techniques were tested. This was made possible due to the Robotics System Toolbox for Simulink [9] with added trajectory planner blocks. The possible trajectory planners were based on the trapezoidal velocity, cubic splines and B-splines methods. Ultimately, the B-splines method was chosen because it provided the smoothest version of the desired trajectory with the least amount of waypoints. In terms of how the B-spline method performs, there is little documentation available, but some observations were made during testing. The method clearly prioritized the first and the last waypoints. However, the waypoints in between the start- and endpoint were typically only used as constraints. In other words, these B-spline trajectory planner would hit the first and last waypoints while staying within the waypoints on the way. How close the desired trajectory were to the set waypoints were dependent on the distance between each waypoint.

The input to the trajectory planner was set in joint space, and consequently the waypoints had to be converted from the configuration space (Cartesian space) to the joint space using inverse kinematics. Converting to the joint space from the configuration space was necessary to control the end-effector and make sense of its position in the XYZ-plane. The inverse kinematics conversion was done by performing the optimization method described in Section 7.3. Due to the optimization method possibly having multiple joint configuration for a given end-effector waypoint, each waypoint were found with the previous waypoint as input.

8.1.2 Dynamics Model Matlab Functions

The Simulink blocks in Figure 8.1 denoted `Inverse Dynamics` and `Forward Dynamics`, are Simulink function blocks. The function blocks consists of the forward and inverse dynamics forms of the equations of motion as described in Section 4.3. The functions are Matlab functions that were used to first linearize the system (inverse dynamics) and then calculate the joint accelerations (forward dynamics) as output to the feedback loop of the IDC. The inertia matrix for the Bravo 7 is big and complex and the simulation of the dynamics model was therefore solved numerically in a Simulink environment by having the joint values as input in every time step.

The Matlab functions for the Simulink function blocks were generated from symbolic expressions. All the required computations for the matrices required in the equations of motion were first written symbolically by using the computational approaches presented in Chapter 7. Then the symbolic expressions were converted to a Matlab function by using the built-in function of `matlabFunction`. All the matrix equations were then chosen as outputs, and set to form either an inverse or forward dynamics form of the equations of motion. Additionally, the variable inputs were decided as the joint position, joint velocity and the current velocity.

8.2 Case Studies

In relation to the smolt production, there were two operations simulated for the case studies:

1. **Dead Fish Pickup:**

   The robot manipulator was controlled to follow the desired trajectory and reach the pickup point. The desired trajectory was implemented as a descending motion where the end-effector was lowered towards an end-position for the pickup.

2. **Tank Cleaning:**

   The robot manipulator was controlled to follow the desired trajectory to cover a area of interest. The desired trajectory was implemented as a circular motion of the end-effector.

8.2.1 Performing the Simulations

The orientation of the end-effector was constant for the entire duration of the simulations. Additionally, the Bravo 7 was initialized in its zero-configuration before it started tracking the desired trajectory. All the simulations were performed with an artificial error of 10 g added to the weight to the mass of each link. The intention was to slightly unsettle the perfect cancellation between the nonlinear dynamics system and the the nonlinear compensation.

Below is a step-by-step presentation of how the simulation of the case studies were performed:

1. Pick all the waypoints in the configuration space (as few as possible to complete the desired trajectory).
2. Convert the waypoints to the joint space by using inverse kinematics with the previous waypoint as input.
3. Choose a suitable time frame for the simulation.
4. Run numeric simulation in Simulink using the `ODE23s` solver (solver was picked based on trial-and-error).
5. Create desired plots in Matlab.

8.2.2 Dead Fish Pickup Inputs

The trajectory planner had four waypoint inputs in the configuration space, given by the following XYZ-coordinates (in order): $(-0.25, 0, 0.07)$, $(0, 0, 0)$ $(0.425, 0, 0)$, $(0.65, 0.04, 0)$. This means the desired trajectory of the end-effector is set to descend to 0.65 m below its base frame origin and pickup a dead fish. Based on these configuration space waypoints, the inverse kinematics gives the following joint configuration waypoints:

$$\boldsymbol{Q_w} = \begin{bmatrix} 0 & 0 & 0 & 0 \\ 1.0594 & 0.1992 & -0.4219 & -1.3937 \\ -2.7557 & -1.3608 & 0.7762 & 2.6877 \\ 0 & 0 & 0 & 0 \\ -0.1255 & 0.4092 & 1.9251 & 2.7939 \\ 0 & 0 & 0 & 0 \end{bmatrix} \tag{8.1}$$

where each column represents the joint variables $\boldsymbol{q}$.

8.2.3 Tank Cleaning Inputs

The trajectory planner had seven waypoint inputs in the configuration space, given by the following XYZ-coordinates (in order): $(0, 0, 0)$, $(0.5, 0, 0)$, $(0, 0.5, 0)$, $(-0.5, 0, 0)$, $(0, -0.5, 0)$, $(0.25, -0.5, 0)$, $(0.5, 0, 0)$. This means the desired trajectory of the end-effector is set to follow a circle of radius 0.5 m around its base frame origin.

The joint configuration inverse kinematics gives the following joint configuration waypoints:

$$\boldsymbol{Q_w} = \begin{bmatrix} 0 & 0 & 2.1340 & 3.1416 & 4.3901 & 5.3086 & 6.2832 \\ 0.1992 & -0.6287 & -1.4202 & -1.5337 & -1.4017 & -1.4251 & -0.8779 \\ -1.3608 & 1.1882 & 2.9776 & 2.6877 & 2.8054 & 2.9089 & 1.5527 \\ 0 & 0 & 1.5535 & 0 & 1.5621 & 1.6822 & 3.1416 \\ 0.4092 & 2.1303 & 1.1005 & -0.3049 & -1.3946 & -2.3249 & -2.2457 \\ 0 & 0 & -1.5677 & -3.1416 & -1.7353 & -1.7086 & -3.1416 \end{bmatrix} \tag{8.2}$$

where each column represents the joint variables $\boldsymbol{q}$.

Part III

Results, Discussion and Conclusive Remarks

Chapter 9

Results

This chapter involves results related to the implemented control system and trajectory planner. The results are all plots from the Matlab Simulink simulation of two relevant case study operations related to real-life smolt production. The intention is to present the general system performance as well as showcasing potential manipulator tasks.

9.1 Case Studies with Trajectory Visualization

The case studies are demonstrated by various plots simulated in Matlab, using the Simulink model presented in Section 8.1, with input waypoints from the trajectory planner setup described in Section 8.1.1. The dead fish pick up and the tank cleaning operations were the two case studies simulated The simulations were performed with and without the hydrodynamic and hydrostatic effects included in the model according to the step-by-step procedure presented in Section 8.2.1.

9.1.1 Dead Fish Pickup

The fish pickup operation was simulated with the desired trajectory following the waypoint inputs from Section 8.2.2 resulting in a descending motion towards the bottom of the smolt tank. The arm is assumed to be attached to a fully stationary vehicle with the resulting z-axis pointing towards the wall of the tank and the x-axis pointing towards the bottom of the tank.

Without Underwater Effects

First, a plot showing how each joint position is able to reach its desired value and then track this value over time is presented in Figure 9.1. Then a position plot in the zx-plane is presented in 9.2. The zx-plane is interesting for this operation, since the end-effector is lowered in the x-direction. The starting position (the green circle) is decided by the zero-configuration of the manipulator. Initially, the manipulator has to move from its zero-configuration before reaching the desired trajectory.The last trajectory plot in Figure 9.3 is a visualization of the end-effector positioning in three dimensions. Note the positive x-direction is the downward motion of the end-effector.

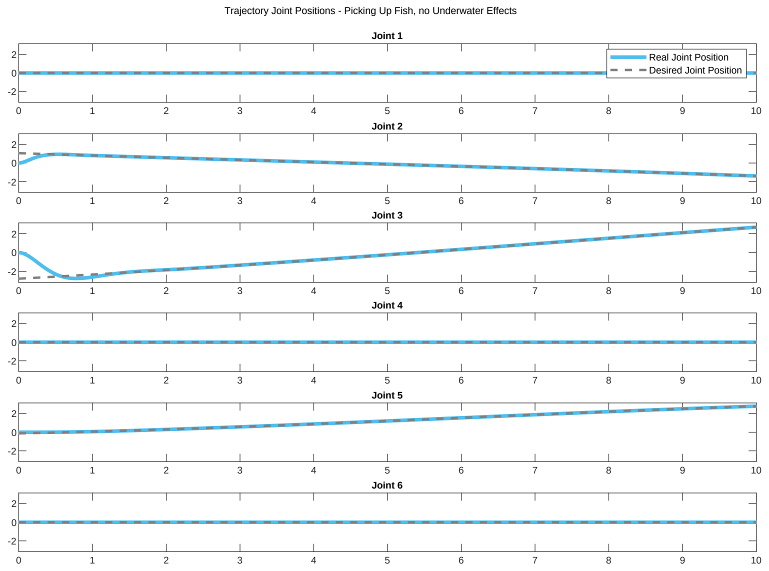


Figure 9.1: Simulation plot of real vs. desired joint positions for the dead fish pickup operation, without added underwater effects.

With Added Underwater Effects

The same three plots, as for the case with no underwater effects, are presented in Figure 9.4, 9.5 and 9.6. An additional plot of the error between the desired and real trajectory positions is included in Figure 9.7.

9.1.2 Tank Cleaning

The tank cleaning operation is showcased by circular motion while imagining a rotating brush at the end-effector. The input waypoints are presented in Section 8.2.3 Similar to the fish pickup operation, the arm is assumed to be attached to a fully stationary underwater vehicle with the resulting z-axis pointing towards the wall of the tank and the x-axis pointing towards the bottom of the tank.

Without Underwater Effects

The first plot in Figure 9.8 demonstrating the control systems ability to track the desired joint positions. The second plot in Figure 9.9 is performed in the xy-plane. This is due to the fact that the wall of the tank is parallel to the xy-plane and is where the cleaning operation was to be performed. The last plot is once more a three dimensional plot of the end-effector trajectory.

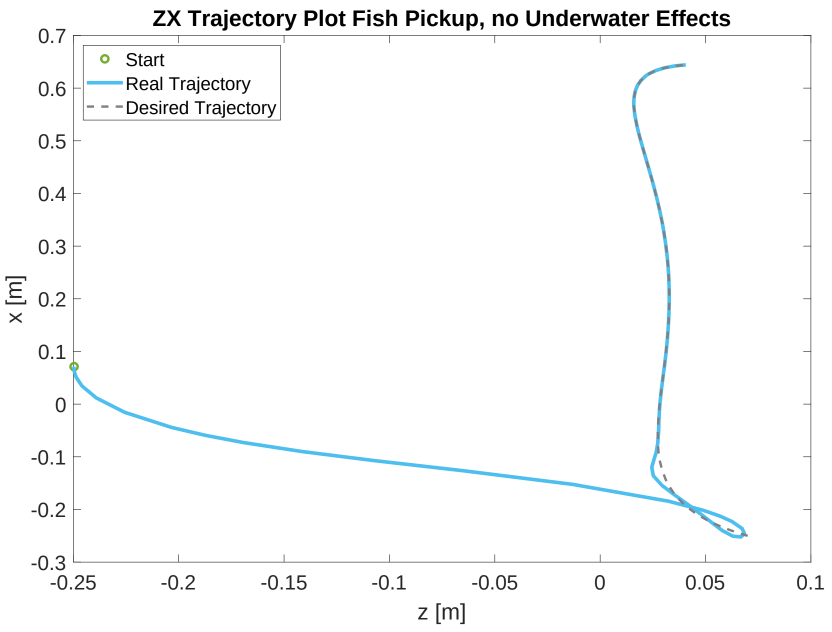


Figure 9.2: Trajectory plot of the real vs. desired trajectory for the tank cleaning operation in the zx-plane, without added underwater effects.

With Added Underwater Effects

The same three plots, as for the case with no underwater effects, are presented in Figure 9.11, 9.12 and 9.13. An additional plot of the error between the desired and real trajectory positions is included in figure 9.14.

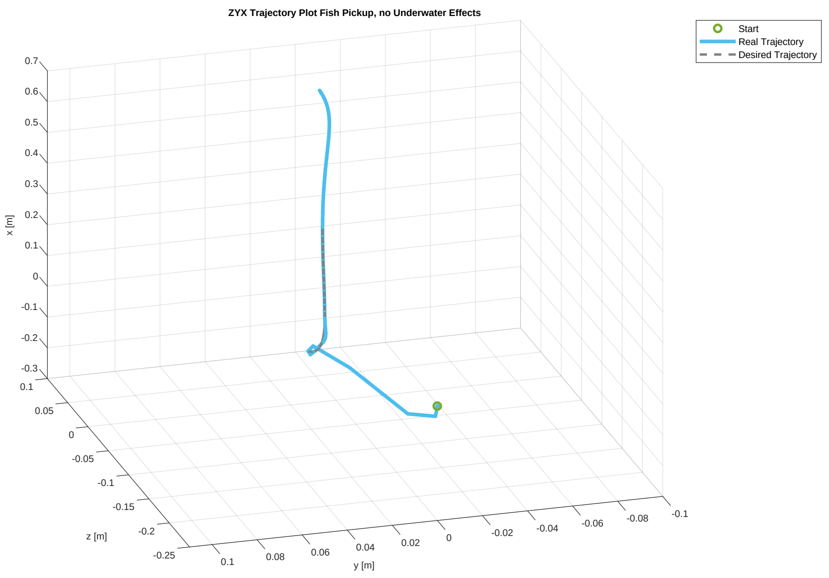


Figure 9.3: 3D trajectory plot of the real vs. desired trajectory for the dead fish pickup operation in the zyx-space, without added underwater effects.

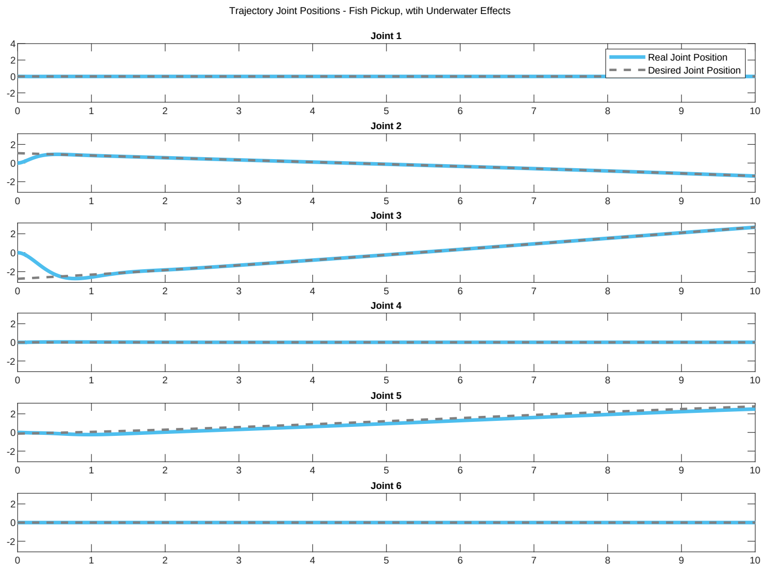


Figure 9.4: Simulation plot of real vs. desired joint positions for the dead fish pickup operation, with added underwater effects

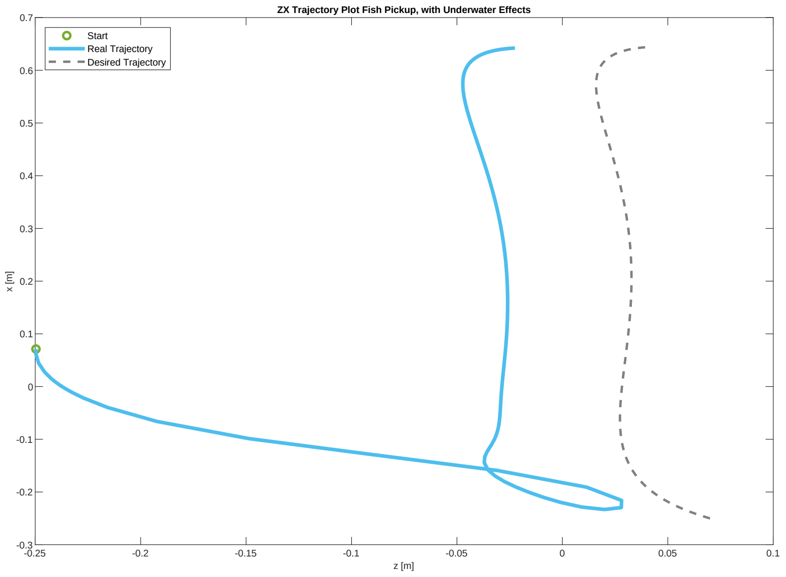


Figure 9.5: Trajectory plot of the real vs. desired trajectory for the dead fish pickup operation in the zx-plane, with added underwater effects.

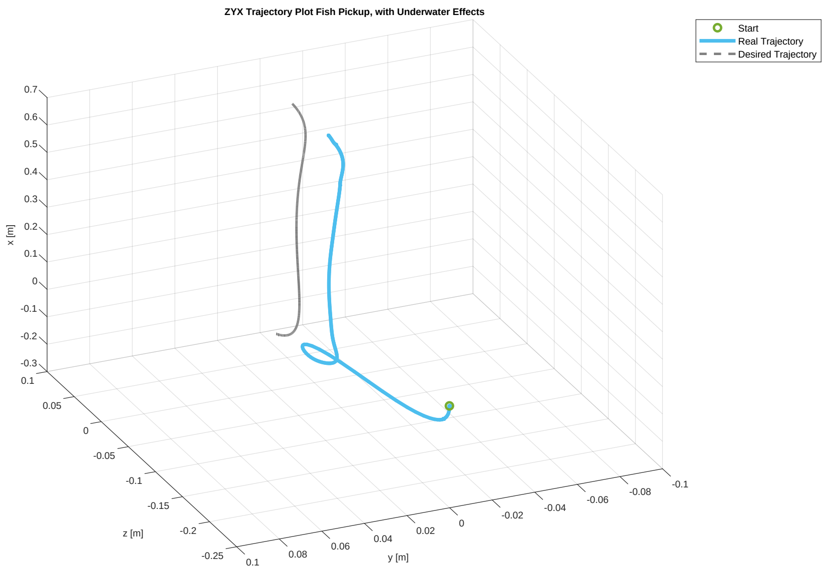


Figure 9.6: 3D trajectory plot of the real vs. desired trajectory for the dead fish pickup operation in the zyx-space, with added underwater effects.

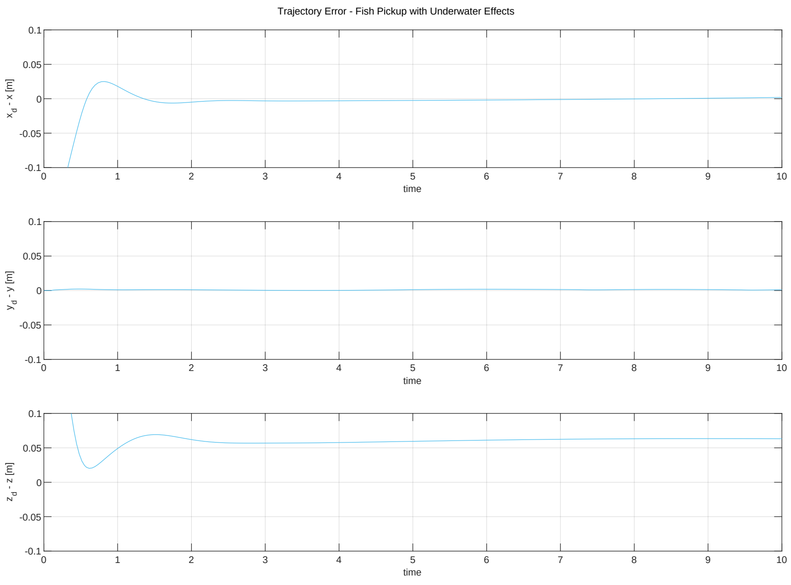


Figure 9.7: Error between the desired and real joint values for the dead fish pickup operation, with added underwater effects.

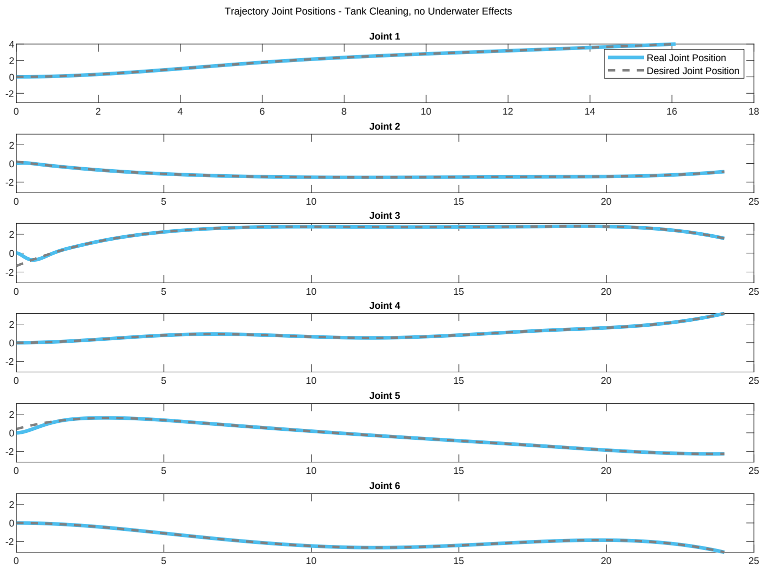


Figure 9.8: Simulation plot of real vs. desired joint positions for the tank cleaning operation, without added underwater effects.

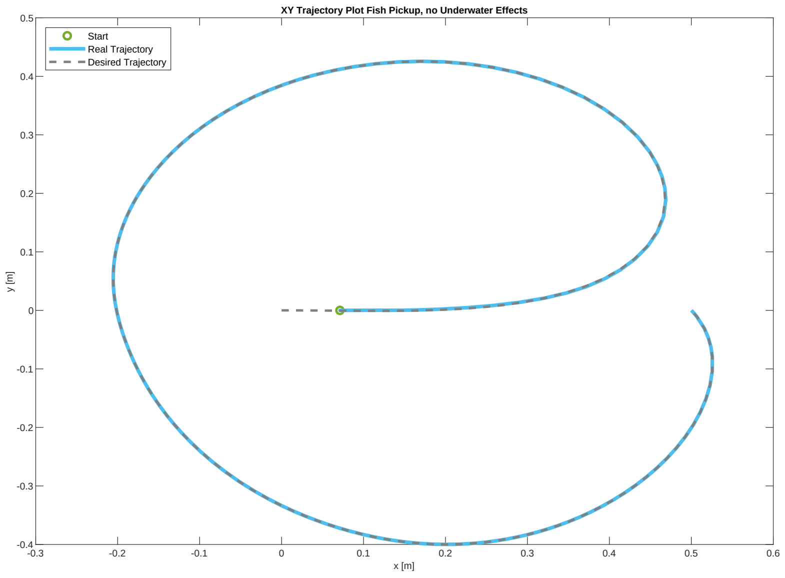


Figure 9.9: Trajectory plot of the real vs. desired trajectory for the tank cleaning operation in the xy-plane, without added underwater effects.

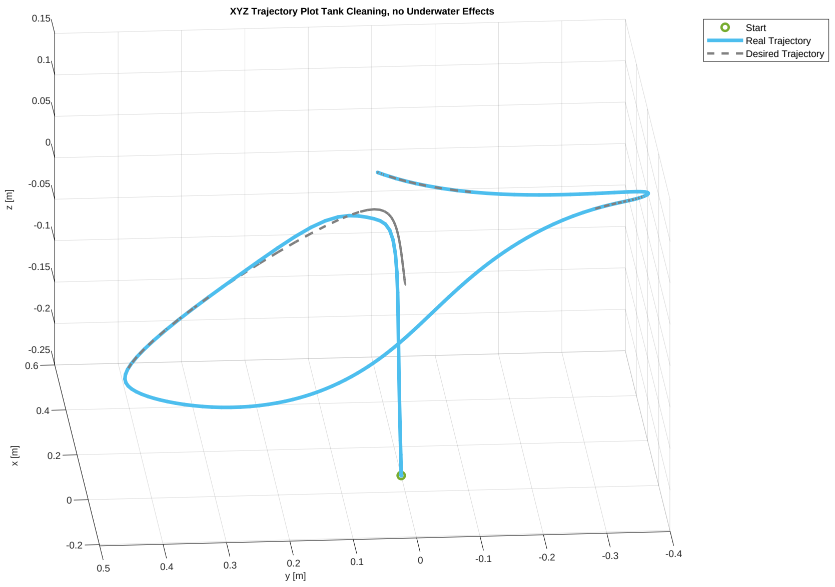


Figure 9.10: 3D trajectory plot of the real vs. desired trajectory for the tank cleaning operation in the xyz-space, without added underwater effects.

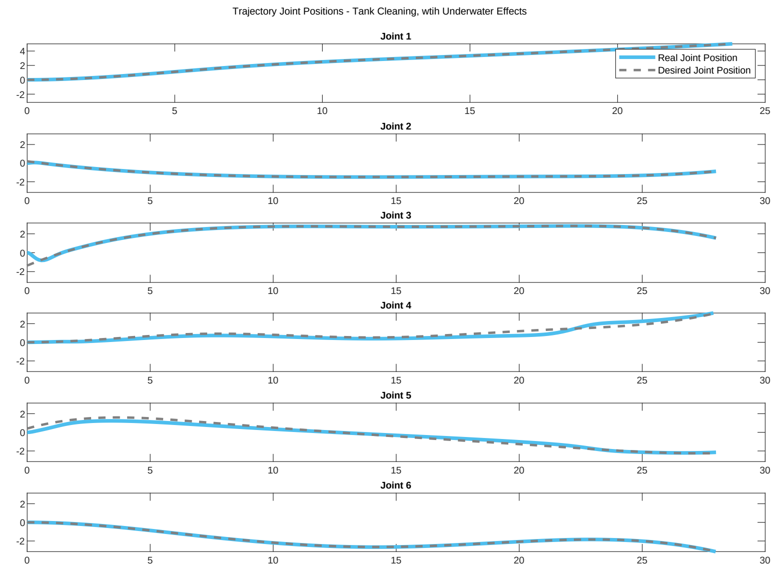


Figure 9.11: Simulation plot of real vs. desired joint positions for the tank cleaning operation, with added underwater effects.

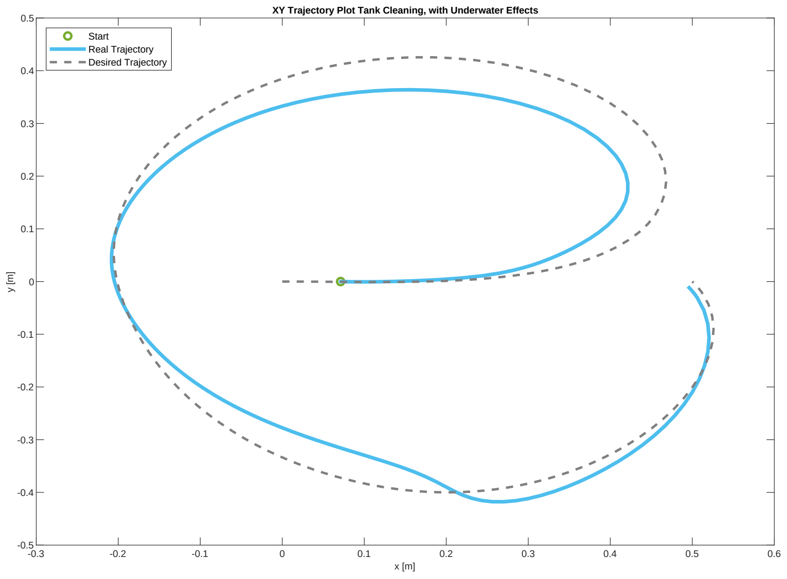


Figure 9.12: Trajectory plot of the real vs. desired trajectory for the tank cleaning operation in the xy-plane, with added underwater effects.

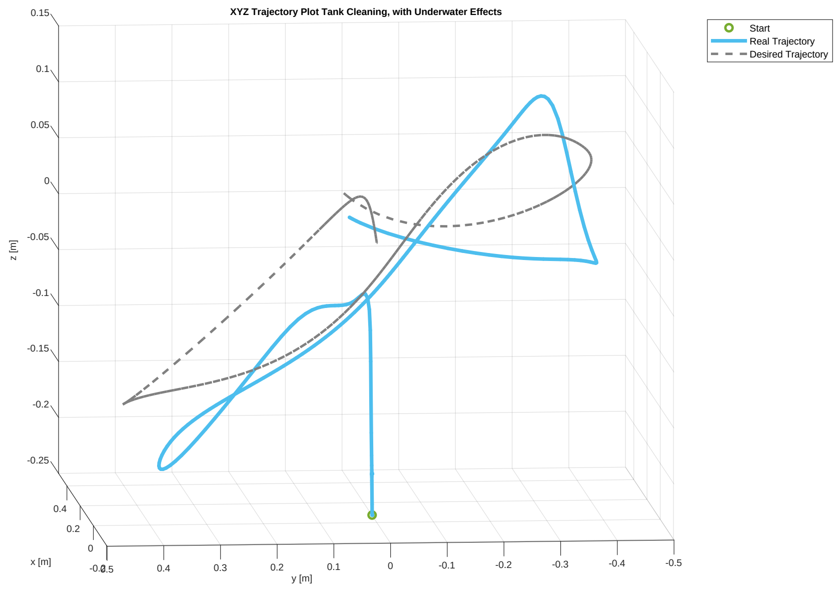


Figure 9.13: 3D trajectory plot of the real vs. desired trajectory for the tank cleaning operation in the xyz-space, with added underwater effects.

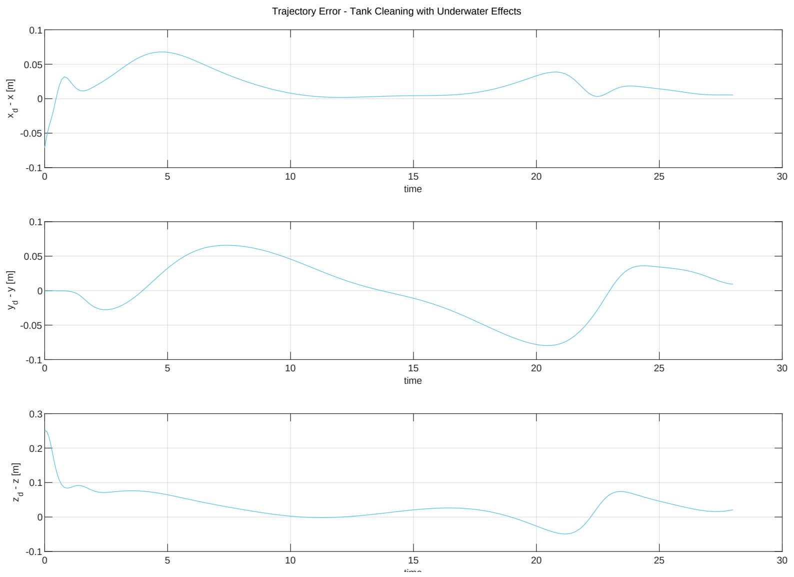


Figure 9.14: Error between the desired and real joint values for the tank cleaning operation, with added underwater effects.

Chapter 10

Discussion

This chapter contains the discussion regarding the performance of the system based on the background material from Part I and the results in Chapter 9. At first, in Section 10.1 some of the expected model and control system limitations are discussed, before moving on to discussing the results from Section 10.2. The last part of the chapter, in Section 10.2.6, presents a list containing suggestions for possible future work based on the findings in this master thesis.

10.1 Model and Control System Limitations

As for most research related to mathematical modeling of physical systems, the research in this thesis involves certain limitations. The setup of the manipulator and fluid dynamics are highly complex systems. Thus, combining the two systems demands an appropriate amount of simplifications compared to the real world to avoid too slow and demanding computations. Simplifications, although necessary, can become prominent sources of error.

Regarding the robot manipulator, the model limitations are, in fact, not directly coupled to the mathematical modeling but rather its role in the control system. The model derivation, based on the DH-convention, is based on established robot manipulator research and has been proven to create precise models. However, the physical inputs need to form perfect cancellation of the nonlinearities. Normally, the physical input parameters are not perfect, making the IDC somewhat limited.

The introduction of the underwater effects amplifies the limitations of the IDC while also involving very hard-to-model physical effects. The simplifications to the hydrodynamic effects on the manipulator might be acceptable for certain conditions, but a smolt tank involves currents and added turbulence from the fish. Furthermore, the simplified added mass effect assumes no currents. Currently, the model in this thesis assumes only the drag force to be affected by the relative velocity. Each of the mentioned components, factors into external disturbances of possible significant value. Therefore, limitations of the current control system approach are to be expected if used to control the Bravo 7 in a real-world smolt tank. Still, the simulations displayed below, do involve perfect cancellation of the physical parameters and should perceived perfectible with this in mind.

10.2 Simulations and Results

The IDC system performance is the main interest regarding the system performance, but in virtue of the results in the previous chapter, the performance of trajectory planner also becomes a noteworthy part of the discussion. To make the link to the result section unambiguous, the two case studies are discussed in a separate manner, with the similar captions as in the result section. In general, the trajectory tracking ability of the IDC system was expected to perform quite well because of the perfect nonlinear compensation of the dynamics model. However, the perfect cancellation also emphasize the limitations of the simulations relative to the real-world scenario of a fish-tank.

10.2.1 Simulink Model Performance

The general performance of the Simulink model setup with Matlab-function blocks were acceptable for the case studies, but proved to have restrictions regarding the added underwater effects and certain other configurations. As soon as the Simulink setup had compiled the setup with a given set of waypoints, the simulation speed of the performed case studies were acceptable. However, if too many waypoints or for some specific waypoint combinations, the simulations became extremely slow or resulted in error messages. Usually, the error messages were fixed by lowering the relative tolerance value or by changing the numerical simulation method, which again resulted in slow simulations. A more specific limitation of the Simulink model setup, was related to the simulations with added underwater effects. These simulations were only allowed to set the current speed as low as 1 m/s. For lower speeds, a diffuse error message was presented which, based on some investigation, were confirmed to be specific for Simulink. Because the error was related to the current speed, the cause is very likely related to the drag force model. In conclusion, the Simulink setup with the Matlab-function blocks worked well for the case studies, but proved to have limitations beyond this scope.

10.2.2 Dead Fish Pickup without Underwater Effects

The plot results of the simulated dead fish pickup operation with no underwater effects were generally as expected. The joint positions in Figure 9.1 showed excellent tracking of the desired joint positions, with little sign to the slow transient response mentioned in Chapter 6. The trajectory plot in the zx-plane displayed the systems' promising ability to perform the motion for the pickup. The three dimensional trajectory plot confirmed the notion of a well behaved system as can be seen in Figure 9.3.

10.2.3 Dead Fish Pickup with Underwater Effects

The resulting plots from the tank cleaning operations with added underwater effects were interesting, but not unexpected. From Figure 9.4 it is at first hard to see much difference compared to the same simulation without underwater effects. Still, joint 5 shows a resulting steady state error when tracking its desired position. The steady state error proves the effect on the position tracking from the hydrodynamic and hydrostatic forces. In Figure 9.5 and 9.6 the resulting trajectory tracking with the present steady state error shows up as a shifted real trajectory unable to track the desired positions. Thus, the endpoint of the fish pickup is missed by about 2 cm. An error of this magnitude is relatively small, especially since the current speed of 1 m/s is way above the typical values for smolt tanks. Furthermore, the IDC was never tuned for the added underwater effects. One conclusion may be that the underwater effects have little impact on the control of the manipulator when under water. However, the unrealistically perfect cancellation of the nonlinearities for this simulation still makes a difficult case in terms of supporting this conclusion without any further research.

10.2.4 Tank Cleaning without Underwater Effects

The resulting plots from the tank cleaning operations with no water effects showed particularly promising results in regards to the trajectory tracking. For the joint positions and xy-plane trajectory plots, much of the same as for dead fish picking operation could be repeated. In particular, the result in Figure 9.9 displayed perfect trajectory tracking of the desired outputs from the trajectory planner. Still, the trajectory planner is the weak link in this scenario. Even though the waypoints are set for a z-value constantly equal to zero, the trajectory planner has difficulties keeping the desired trajectory constant. For the manipulator to clean a flat surface, this generated trajectory lowers the control and quality of the performed operation. A possible solution could be to involve a lot more waypoints, but one major implication of more waypoints is the slow simulations.

10.2.5 Tank Cleaning with Underwater Effects

The resulting plots from the tank cleaning operations with added underwater effects displays a control system struggling to track the desired trajectory. The results for the joint positions in Figure 9.11 presents difficulties controlling joint 4 and joint 5, but without steady state errors. In other words, a more unpredictable error compared to the dead fish pickup operation. Due to the unpredictable behaviour, the Bravo 7 manipulator ends up both over-reaching and under-reaching relative to the desired trajectory, as can be seen in Figure 9.12 and 9.13. By disregarding the lackluster attempt from the trajectory planner to follow the waypoints, the error between the desired and controlled trajectory is sometimes as big as 5 cm in various directions. Even though the end-effector seems to hit the end-points, the cleaning operation should follow the desired trajectory through the whole time frame. In the end, it its challenging to make predictions about how the IDC system would perform with lower current velocities, but unpredictable behaviour due to the underwater effects is a definitely a possibility to be taken into consideration.

10.2.6 Future Work

Discussing what the results above implies for the use of the Bravo 7 in smolt tanks is for the most part related to future work possibilities. The mathematical models and the subsequent IDC system created here makes for a good foundation for further work. In reality, the properties of the IDC clearly suggests that a different, more robust control system, is required to handle the added underwater effects when controlling the Bravo 7 manipulator. Due to the manipulator complexity, working towards a more efficient models for simulation is also important. Other possibilities are to improve the dynamics modelling for the underwater effects, and to look into more suitable techniques for trajectory planning.

Based on the discussion and the current implementation, there are several possibilities to use this thesis as basis for future work. Some suggestions are listed below:

1. **Implement more robust control algorithms:**

   There are various control methods for underwater manipulators better suited than the IDC system for the parameter uncertainties in the model and for handling the nonlinearities. A summary of some of these methods is provided in [55].

2. **Improve the underwater dynamics model:** Use Kane's Method for developing the dynamic equations is should give more efficient computations as well as providing a straightforward approach for incorporating external hydrodynamic forces into the model. [58].

3. **Implement own Trajectory Planner:** There is little documentation regarding the trajectory planner from the Robotics System Toolbox in Simulink. A specifically implemented trajectory planner for the Bravo 7 robot arm for use in smolt operations would provide more control of the generated trajectories. Especially important for including the physical constraints of the Bravo 7.

4. **Improve the setup for the Trajectory Planner input:** An improved inverse kinematics method for deciding the end-effector waypoints in the Cartesian space.

Chapter 11

Conclusion

This thesis has first and foremost presented a thorough presentation to solving the research objectives introduced in Chapter 1. The conclusive remarks and findings, directly linked to the research question sub-tasks, are stated below.

1. **Perform a literature study on manipulator fluid dynamics modelling.**

   The literature study established a set of relevant underwater effects for a manipulator system. The added effects to the dynamics model included added mass effects, drag forces (involving current effects), and buoyancy. How to calculate these effects and the necessary simplifications for a robot manipulator were also part of the study. Additionally, a review of the feasibility and possibility of a variable drag coefficient was performed. One observation from the literature study was the lack of updated work involving a specific focus on complex underwater manipulators, especially tailored with the DH-convention in mind. Still, a decent amount of studies on underwater vehicle-manipulator-systems were found to be helpful.

2. **Create a mathematical model describing the underwater dynamics.**

   Based on the literature study a generalized method was proposed for calculating the added underwater effects to the land-based manipulator dynamics model. The symbolic expression for the mathematical model was developed in Matlab. The symbolic expression was then converted to a function file by using the built-in `matlabFunction`. This was considered to be an orderly and transparent method for developing the complex mathematical models involved, but the conversion to the function file was terribly time-consuming.

3. **Suggest and implement a suitable control system.**

   In order to control the manipulator end-effector position in an underwater environment, an Inverse Dynamics Controller was chosen due to being an established control method for robot manipulators and thus forming a valuable base for future work. The control system was implemented in Matlab Simulink by including the symbolic model expressions using Matlab-function blocks. Setting up the Simulink model for an Inverse Dynamic control system formed a precise connection to the theoretical model; nevertheless, it caused some debugging issues and unpredictabilities as a simulation setup. More specifically, the simulations were sometimes slow, and there was a chance of obscure error messages appearing for selective configurations.

4. **Suggest and implement methods for trajectory planning.**

   A trajectory planner was implemented as a Simulink block from the Robotic Systems Toolbox, allowing for quick testing of different trajectory planning techniques. A trapezoidal velocity-, polynomial- and B-spline trajectory planner were tested. Based on literature evidence and through testing, the B-spline trajectory planner was preferred. The main problem with the use of this trajectory planner was the lack of documentation. Therefore, it was challenging to figure out what was causing unwanted system behaviors.

5. **Simulation experiments.**

   Simulations were performed in Matlab Simulink, and the trajectories were plotted in the Cartesian space, conduction two smolt production case studies: pickup of dead fish and a tank cleaning operation. Both operations were simulated considering two model setups: with and without the added underwater effects included in the dynamics model. A comparison of the IDC system performance based on the plotted results was then discussed for the two setups.

   As expected, the control system performed well for the land-based dynamics model. The only system flaw was the unanticipated poor execution of the trajectory planner when creating the desired trajectory for the circular cleaning motion. Otherwise, the position tracking was performed with little sign of errors. When the underwater effects were added, the position control was affected, causing a steady-state error for the dead fish pickup and more unpredictable errors for the cleaning trajectory. Still, the error values were relatively small. Thus, considering the simulations were performed with no further tuning of the control gains and with current speeds more than doubling what would be expected, the end-effector position tracking performance was good for all the case studies. Conclusively, the IDC system performed well. However, the simulated perfect cancellation of the physical system is too unrealistic, making for a very low probability of acceptable performance, regarding the manipulator position control, in a real-world smolt tank.

Regarding the more generalized research question from the introduction, a mathematical model of an underwater manipulator has been created and used to control the end-effector position of the Bravo 7 manipulator performing typical day-to-day tasks for smolt production. Although adopting an unrealistically clever IDC system for an already complex nonlinear manipulator system, with the addition of hard-to-model manipulator fluid dynamics, the work in this thesis should be a good foundation for further development and research within smolt production automation.

Appendices

Appendix A

Algorithms

A.1 Drag Force Algorithm

All variables used in the algorithm exists for link $i = 1$ to n with the corresponding strip $j = 1$ to n_{L_i}.

Inputs:

U_F	$=$	flow/current velocity
$\boldsymbol{V_{p_i}}$	$=$	velocity realtive to the base frame of the origin of the local coordinate frame
L_i	$=$	length of link i
n_{L_i}	$=$	number of integration strips i
$\boldsymbol{Up_i}$	$=$	the relative velocity of the point corresponding to the origin of coordinate frame i
$\boldsymbol{l_{i,j}}$	$=$	the position vector from the local frame origin to strip j
$\boldsymbol{\omega_i}$	$=$	angular velocity of link i where $\boldsymbol{\omega_i} = [0 \quad 0 \quad \dot{q}_i]^T$
C_D	$=$	drag coefficient

Steps to compute the drag on link i:

```
Step 1.  $\boldsymbol{u_{F_i}} = \boldsymbol{R_i^T} \boldsymbol{u_{F_{i-1}}}$ where $\boldsymbol{u_{F_i}} = [\cos q_i \quad \sin q_i \quad 0]^T$
Step 2.  $\boldsymbol{U_{p_i}} = U_F \boldsymbol{u_{F_i}} - \boldsymbol{V_{p_i}}$
Step 3.  $\boldsymbol{d_{L_i}} = \boldsymbol{R_i^T}\ \boldsymbol{d_{L_{i-1}}}$ where $\boldsymbol{d_{L_1}} = [0 \quad 0 \quad 1]^T$
Step 4.  $dl_i = \frac{L_i}{n_{L_i}}$
Step 5.  $\boldsymbol{dl_i} = dl_i \boldsymbol{d_{L_i}}$
Step 6.  for $j = 1$ to $n_{L_i}$
             a. $\boldsymbol{l_{i,j}} = \boldsymbol{l_{i,j-1}} + \boldsymbol{dl_i}$ where $\boldsymbol{l_{i,0}} = [0, 0, 0]^T$
             b. $\boldsymbol{U_{ij}} = \boldsymbol{U_{p_i}} - \boldsymbol{\omega_i} \times \boldsymbol{l_{ij}}$
             c. $U_{ij}^2 = \boldsymbol{U_{ij}} \cdot \boldsymbol{U_{ij}}$
             d. $\boldsymbol{u_{ij}} = \frac{\boldsymbol{U_{ij}}}{\sqrt{U_{ij}^2}}$
             e. $\alpha_{d_{i,j}} = \arccos |\boldsymbol{u_{i,j}} \cdot \boldsymbol{d_{L_i}}|$ where $\boldsymbol{d_{L_i}} = [0 \quad 0 \quad 1]^T$
             f. $C_{N_{ij}} = [C_D^2 + (\sin \alpha_{d_{i,j}})^4]^{1/2}$
             g. $\boldsymbol{dF_{ij}} = \frac{1}{2}(\rho C_{N_{ij}} U_{ij}^2 D_i dl_i) \boldsymbol{c_{ij}}$
```

h. $\boldsymbol{dN_{ij}} = \boldsymbol{r_{ij}} \times \boldsymbol{dF_{ij}}$

Step 7. $\boldsymbol{N_{D_i}} = \sum_{j=1}^{n_{L_i}} \boldsymbol{dN_{ij}}$

Appendix B

Contents of Delivered ZIP-File

PDF Files

- *Project Thesis:* The specialization project thesis.
- *Master Thesis:* The master thesis.
- *Project Task Description:* The given project description with background information for the master thesis.
- *Bravo Reach7 Documentation:* Documentation of the Bravo Reach 7 robot from Blueprint Lab. Contains important parameter values for the mathematical model.
- *Bravo Reach7 Datasheet:* Datasheet of the Bravo Reach 7 robot from Blueprint Lab. Contains a general introduction of the manipulator as well as physical specifications.

Matlab Files

- *getDCGfunc_v4.m:* Matlab function to set up the matrices for the dynamics model in symbolic form..
- *bravo_fk.m:* Matlab function to compute the forward kinematics.
- *bravo_ik.m:* Matlab script to compute optimization problem of the inverse kinematics.

Simulink Files

- *inverse_dynamics_control.m:* Simulink diagram of the inverse dynamics controller.

Bibliography

[1] Antonelli, G. *Underwater Robots.* 3rd. Springer, 2014.

[2] Antonelli, G. et al. *Adaptive Control of an Autonomous Underwater Vehicle. Experimental Results on ODIN.* IEEE Transactions on Control Systems, 1999.

[3] Bank, T. W. *FISH TO 2030 - Prospects for Fisheries and Aquaculture.* 2013.

[4] Barhaghtalab, M. H. et al. *Dynamic Analysis, Simulation, and Control of a 6-DOF IRB-120 Robot Manipulator using Sliding Mode Control and Boundary layer method.* Journal of Central South University, 2018.

[5] Bjørndal, T. and Tusvik, A. *Economic analysis of on-growing of salmon post-smolts.* Taylor and Francis Online, 2020.

[6] Castro, S. *Knuth: Computers and Typesetting.* URL: https://blogs.mathworks.com/racing-lounge/2019/11/06/robot-manipulator-trajectory/.

[7] Corke, P. *Robotics, Vision and Control.* Springer International Publishing AG, 2017.

[8] Deshpande, V. A. *Dynamics of Robot Manipulators: A review.* International Journal of Engineering Research and Technology, 2010.

[9] Documentation, S. *Simulation and Model-Based Design.* 2020. URL: https://www.mathworks.com/products/simulink.html.

[10] Egeland, O. and Gravdahl, J. T. *Modelling and Simulation for Automatic Control.* Norwegian University of Science and Technology, 2002.

[11] Ersdal, S. *An Experimental Study of Hydrodynamic Forces on Cylinders and Cables in Near Axial Flow.* Norwegian University of Science and Technology, 2004.

[12] Food and United Nations, A. O. of the. *Aquaculture production: Quantities 1950-2014.* URL: http://www.fao.org/fishery/%20aquaculture/en.

[13] From, P. J., Gravdahl, J. T., and Pettersen, K. Y. *Vehicle-Manipulator Systems: Modeling for Simulation, Analysis, and Control.* Springer-Verlag London, 2014.

[14] Føre, M. et al. *Precision fish farming: A new framework to improve production in aquaculture.* Elsevier Ltd., 2017.

[15] Gao, R. *Inverse kinematics solution of Robotics based on neural network algorithms.* Journal of Ambient Intelligence and Humanized Computing, 2020.

[16] Gasparetto, A. and Zanotto, V. *A New Method for Smooth Trajectory Planning of Robot Manipulators.* Mechanism and Machine Theory, 2007.

[17] Haselirad, A. and Neubert, J. *A Comparison of Three Trajectory Planning Methods for Smooth Motion in 5-DOF Manipulators.* IEEE international conference on Mechatronics and Automation, 2014.

[18] I.Fossen, T. *Guidance and Control of Ocean Vehicles.* 1st. John Wiley and Sons, 1994.

[19] I.Fossen, T. *Handbook of Marine Craft Hydrodynamics and Motion Control.* 1st. John Wiley and Sons, 2011.

[20] Iversen, M. H. *Stress and its impact on animal welfare during commercial production of Atlantic salmon (Salmo salar L.)* Nord University, PHD, 2013.

[21] J. Yuh, S. Z. and Lee, P.-M. *Application of adaptive disturbance observer control to an underwater manipulator.* IEEE International Conference on Robotics and Automation, 2001.

[22] Jagan, G. et al. *Water velocity in commercial RAS culture tanks for Atlantic salmon smolt production.* Elsevier B.V., 2018.

[23] Jankowski, K. P. *Inverse Dynamics Control in Robotics Applications.* Trafford Publishing, 2004.

[24] Kelasidi, E. *Using robotics to drive up efficiency and minimise risks.* URL: https://blog.sintef.com/sintefocean/using-robotics-to-drive-up-efficiency-and-minimise-risks/.

[25] Khalil, W. and Dombre, E. *Modeling, Identification and Control of Robots.* Elsevier Ltd., 2004.

[26] Kołodziejczyk, W. *Some Considerations on an Underwater Robotic Manipulator Subjected to the Environmental Disturbances caused by Water Current.* Acta Mechanica et Automatica, 2016.

[27] Kruusmaa, M. et al. *Salmon behavioural response to robots in an aquaculture sea cage.* The Royal Society Publishing, 2020.

[28] Lab, B. *About Blueprint Lab.* URL: https://blueprintlab.com/about-us/.

[29] Lab, B. *Reach Bravo.* URL: https://blueprintlab.com/products/reach-bravo/.

[30] Lab, B. *Reach Bravo.* URL: https://blueprintlab.com/products/reach-bravo/.

[31] Leabourne, K. N. and Rock, S. M. *Model Development Of An Underwater Manipulator For Coordinated Arm-Vehicle Control.* Stanford University Aerospace Robotics Laboratory, 1998.

[32] Lee, M. and Choi., H.-S. *A robust neural controller for underwater robot manipulators.* IEEE Transactions on Neural Networks, 2000.

[33] Lévesque, B. and Richard, M. J. *Dynamic Analysis of a Manipulator in a Fluid Environment.* The International Journal of Robotics Research, 1994.

[34] Li, R. et al. *Dynamic Modeling of Underwater Manipulator and its Simulation.* International Journal of Mechanical and Mechatronics Engineering, 2012.

[35] Lipkin, H. *A Note On Denavit-Hartenberg Notation In Robotics.* American Society of Mechanical Engineers, 2005.

[36] Liu, H., Zhang, Y., and Zhu, S. *Novel Inverse Kinematic Approaches for Robot Manipulators with Pieper-criterion based Geometry.* International Journal of Control, Automation, and Systems, 2015.

[37] McLain, T. W. *Experiments in the coordinated control of an underwater arm/vehicle system.* Kluwer Academic Publishers, 1996.

[38] McLain, T. W. and Rock, S. M. *Development and Experimental Validation of an Underwater Manipulator Hydrodynamic Model.* The International Journal of Robotics Research, 1998.

[39] McLain, T. W. and Rocky, S. M. *Development and Experimental Validation of an Underwater Manipulator Hydrodynamic Model.* The International Journal of Robotics Research, 1998.

[40] McMillan, S., Orin, D. E., and McGhee, R. B. *Efficient Dynamic Simulation of an Underwater Vehicle with a Robotic Manipulator.* Proceedings of the International Conference on Robotics and Automation, 1994.

[41] Meredith, M. and Maddock, S. *Real-Time Inverse Kinematics: The Return of the Jacobian.* University of Sheffield, 2004.

[42] Mishra, V., Vengadesan, S., and Bhattacharyya, S. *Translational Added Mass of Axisymmetric Underwater Vehicles with Forward Speed Using Computational Fluid Dynamics.* Journal of Ship Research, 2011.

[43] Newman, J. N. *Marine Hydrodynamics.* 40th anniversary edition. Massachusetts Institute of Technology, 2017.

[44] Nofima. *Potensial for høy overlevelse i fremtidens oppdrett.* URL: https://nofima.no/nyhet/2014/10/potensiale-for-hoy-overlevelse-i-fremtidens-oppdrett/.

[45] *Optimization Toolbox.* 2020. URL: https://www.mathworks.com/help/optim/.

[46] Piazzi, A. and Visioli, A. *Global Minimum-jerk Trajectory Planning of Robot Manipulators.* IEEE Transactions on Industrial Electronics, 2000.

[47] Rambely, A. S., Halim, N. A., and Ahmad, R. R. *A Numerical Comparison of Lagrange and Kane's Methods of an Arm Segment.* World Scientific Publishing Company, 2012.

[48] Romano Capocci, e. a. *Inspection-Class Remotely Operated Vehicles—A Review.* Journal of Marine Science and Engineering, 2017.

[49] Schjølberg, I. *Modeling and Control of Underwater Robotic Systems.* Norwegian University of Science and Technology, 1996.

[50] Schjølberg, I. and I.Fossen, T. *Modeling and Control of Underwater Vehicle-Manipulator Systems.* Norwegian University of Science and Technology, 1994.

[51] Shanda, W. et al. *Existence Conditions and General Solutions of Closed-form Inverse Kinematics for Revolute Serial Robots.* MDPI, 2019.

[52] Sharma, A. K. and Saha, S. K. *Simplified Drag Modeling for the Dynamics of an Underwater Manipulator.* IEEE Journal of Oceanic Engineering, 2019.

[53] Siciliano, B. et al. *Robotics: Modelling, Planning and Control.* 3rd. Springer, 2010.

[54] Sintef. *Autosmolt2025.* URL: https://www.sintef.no/en/projects/autosmolt2025/.

[55] Sivcev, S. et al. *Underwater manipulators: A review.* Elsevier Ltd, 2018.

[56] Skeide, P. H. *Automating Tank Operations in Smolt Production – A Concept Study for Automating Tank Cleaning using Robotic Arms.* Norwegian University of Science and Technology, 2020.

[57] Spong, M. W., Hutchinson, S., and Vidyasagar, M. *Robot Modeling and Control.* 1st. John Wiley and Sons, 2005.

[58] Tarn, T., Shoults, G., and Yang, S. *A Dynamic Model of an Underwater Vehicle with Robotic Manipulator using Kane's Method.* Kluwer Academic Publishers, 1996.

[59] The MathWorks, I. *Optimization Toolbox: Least-Squares (Model Fitting) Algorithms.* 2020. URL: https://www.mathworks.com/help/optim/ug/least-squares-model-fitting-algorithms.html.

[60] Timmerhaus, G. et al. *The optimum velocity for Atlantic salmon post-smolts in RAS is a compromise between muscle growth and fish welfare.* Elsevier B.V., 2020.

[61] Torrissen, O. et al. *Salmon Lice - Impact on wild salmonids and salmon aquaculture.* Journal of Fish Diseases, 2013.

[62] W.Fox, R., T.McDonald, A., and J.Pritchard, P. *Introduction to Fluid Mechanics.* John Wiley and Sons, 2004.

[63] Wuxiang Zhang, H. X. and Ding, X. *Design and Dynamic Analysis of an Underwater Manipulator.* Proceedings of the 2015 Chinese Intelligent Automation Conference, 2015.

[64] Yuh, J. *Modeling and Control of Underwater Robotic Vehicles.* Transactions on systems, man, and cybernetics, 1990.

[65] Zhang, D. and Hannaford, B. *IKBT: Solving Symbolic Inverse Kinematics with Behavior Tree.* Journal of Artificial Intelligence Research, 2019.