NTNU
Kunnskap for en bedre verden

DEPARTMENT OF ENGINEERING CYBERNETICS

SPECIALIZATION PROJECT

Kinematic and Dynamic Modelling of the Reach Bravo 7 Robot Manipulator

---

Author:
Oscar Nissen

January, 2021

Abstract

Kinematics and dynamics are two different ways of solving the problem of how to control a robot arm to achieve desired behaviour, which is required for the implementation of control algorithms. This thesis explains the methods of forward and inverse kinematics as well as the derivation of the dynamic model. A complete derivation of the the dynamic model, involving the forward kinematics is, performed by using of the Euler-Lagrange approach. Both analytical and numerical methods to solve the inverse kinematics are attempted. A numerical optimization problem solution is the only method to give any direct results. The results of the dynamic model are created as simulation plots, and the simulations prove that the system is unstable in an open loop configuration, the results are as expected.

Table of Contents

List of Figures

List of Tables

Part I

Introduction and Motivation

Chapter 1

Motivation

The goal of this chapter is to express the motivation behind the project thesis from a global and national perspective based on the predicted future of the aquaculture industry.

1.1 A Growing Population

The UN recognizes aquaculture as an area of opportunity in fulfilling the *Sustainable Development Goals* [3]. The human population is growing and a direct consequence of this is the increased toll on the world's food supplies. Projection studies presents aquaculture as part of the solution [4]. By developing new production techniques and exchanging technology, new countries can be introduced to the industry, which will also contribute to economic growth [5]. Furthermore, as an addition and supplement to livestock, fish farming transpires to a more sustainable food production [6].

1.2 Future Prospects

Fish to 2030 [4] is a study made by The World Bank attempting to predict the future global prospects for fisheries and aquaculture. The previous study, Fish to 2020, was replenished prematurely by the new study after underestimating the growth of the aquaculture industry. In the devised version a much faster aquaculture growth is predicted. The aquaculture production is projected to continue at a strong pace until it matches production of capture fisheries by 2030. Currently observed trends confirm these predictions, and an accelerating growth is the most plausible of all the scenarios suggested in the study. To be able to accommodate the growth in production and the emerging biological, economical and social challenges that follow, technological advancements are imperative. Scandinavia and the salmon aquaculture is mentioned as one of the most technologically advanced regions throughout the industry. Subsequently, the Norwegian Seafood Federation plans to see Norwegian aquaculture become an accelerator of technological development within the industry [5].

1.3 Norwegian Aquaculture Industry

Aquaculture is an essential part of the Norwegian ocean industry and is expected to become even more valuable in the future. Norway is the largest global supplier of Atlantic salmon and due to the the high price of salmon in recent years, aquaculture has become the biggest contributor to value creation in the nation's seafood industry[7]. Nonetheless, growth of the aquaculture industry cannot be determined solely by market demand; it has to occur within the limits which the environment can tolerate.

The Norwegian Seafood Federation has previously expressed an ambition to produce more seafood while having an emphasis on sustainable production [5]. If production is sustainable, then the environment, employees, communities, and organizations all benefit. However, to be able to maintain a sustainable production in a rapidly growing industry presents new challenges which affects fish health and welfare. Therefore, research into new technological applications to be able to control and monitor increasingly complex farming operations is essential to further development of the Norwegian aquaculture industry. Especially in the context of feeding a world with an increasing demand for sustainable food supplies.

Chapter 2

Project Introduction

This chapter is an introduction to the project and presents the description and scope which of what will follow. Section 2.1 explains the life cycle of an Atlantic salmon before the salmon Smolt phase is highlighted in section 2.2 including an introduction the project of Autosmolt 2025. Section 2.3 explains the inspiration of the project thesis and its scope.

2.1 Life cycle of Fish Farmed Atlantic Salmon

To be able to optimise and meet new challenges of modern fish farming, comprehensive knowledge about the entire production cycle of the Atlantic salmon is important. Modern demands of increased production and efficiency does not have a straight forward solution. Production growth is met by biological constraints. Higher levels of knowledge-driven approaches in terms of biological parameters are required [8]. This is related to the entire production cycle of the Atlantic salmon.

The production cycle of captive salmon is presented in Figure 2.1 and in Norway the cycle lasts approximately three years [9]. The freshwater phase requires 10-16 months and the seawater phase adds 14-22 months to this, making a sum of approximately 24-36 months. A more detailed description of the two phases follows.

Fresh Water Phase

The fresh water phase occurs in land-based facilities and involves the various stages of producing smolt, otherwise known as a young salmon. Similarly to how a wild salmon grows up in a river, the farmed salmon starts its production cycle in freshwater. The salmon eggs are initially in an incubator until they are ready to be hatched. The stage after hatching is the alevin stage where the fish is attached to a yolk sac. During the first weeks, the only variables affecting the growth and development is the consumption of the yolk sac and the water temperature [10]. Eventually the yolk sac is absorbed and the alevin develops more distinctive fish-like features. The fish, now called fry, is moved into a fish tank where the initial feeding begins. The

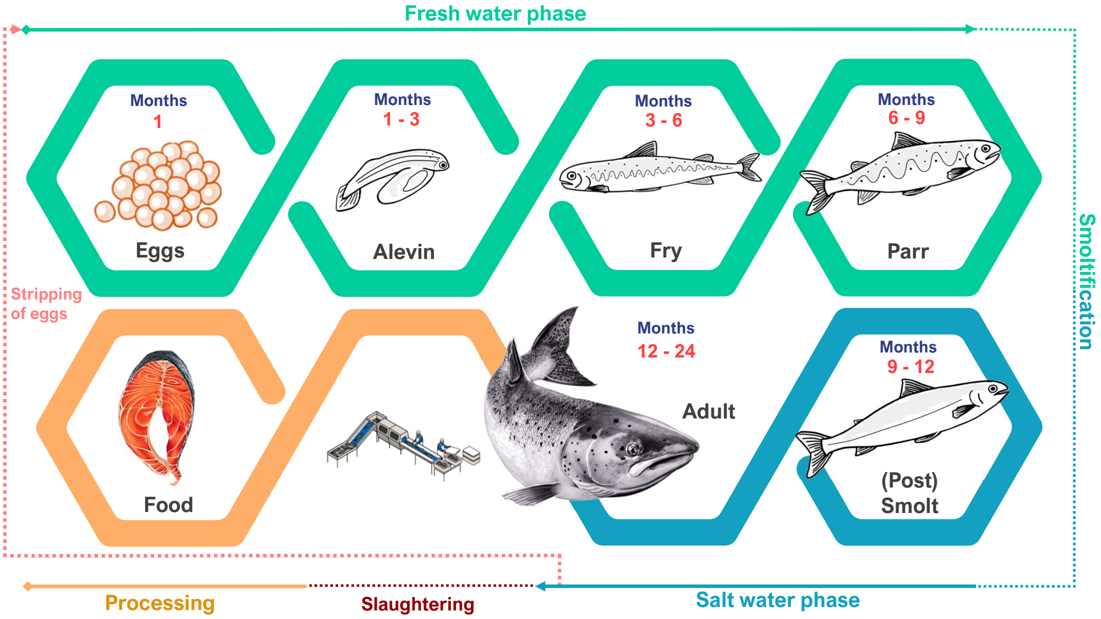


Figure 2.1: The life stages of the Atlantic salmon in captivity.

temperature and lighting conditions are closely monitored to maximize growth. Additionally, there is an ongoing sorting process where the bigger and faster growing fish are moved to bigger tanks. The sorting process is preformed to ensure uniform growth in each tank. After about six to nine months most of the fish are big enough to be referred to as parr. This is the stage right before the process of smoltification. This is a process in which the salmon changes appearance and adapts the ability to live in seawater.

Sea Water Phase

Following the smoltification process, the fish has grown to about 80 grams reaching the stage of post-smolt [11] . Post-smolt marks the initial growth stage of the seawater phase. A stage that has gained an increased amount of attention within the production cycle in recent years. Transporting the smolt directly to an open sea cage after its transition has been the normal convention. However, keeping the post-smolt salmon in a closed in-land facility to grow further, before the transportation, is regarded as the future norm. A salmon is considered as post-smolt until it reaches the weight of 1 kg, but how long it stays in a closed facility varies between the production sites. The common process of every production, regardless of time frame, is to gradually increase the salinity of the water such that the salmon gradually adapts to a life in the sea. When the salmon is deemed ready for a life in the sea, it is transported to a seawater cage. Here the salmon begins the growth spurt until reaching a weight of 4-5 kg [9]. This is the longest growth stage before the fish is harvested, slaughtered and processed. Some of the salmon, which are part of the broodstock, are stripped of eggs to breed future generations.

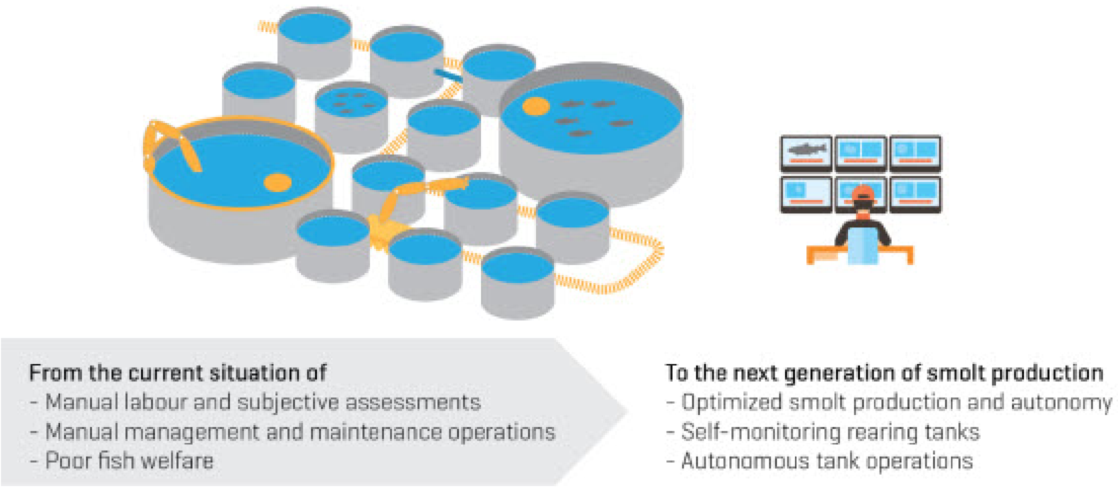
2.2 Emphasis on Smolt Production

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 [12]. 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 [11]. Although the longer land-based production results in increased production costs, based on current prices, it remains profitable in comparison to the risk of lower production volumes due to sea lice. As a result, many new post-smolt facilities are now planned for construction [13]. This will eventually increase the burden on land-based facilities to be able tackle the challenges of the up-scaling production.

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 [1] is an ongoing project where the objective is to bring smolt production closer to the realization within the framework of Industry 4.0.[14] by applying principles of Precision Fish Farming[8]. Precision fish farming is a concept developed to inspire goal-oriented research and innovative technological development by applying control-engineering principles to fish production while still contribute to the fish welfare. The goal is to go from an experienced-driven to a knowledge-driven fish production. The PFF concept is closely linked to the driving forces of Industry 4.0., which are about gathering and connecting information digitally. Industry 4.0 seeks to optimize and control production in terms of essential production parameters through integrated information systems. Based on the combination of PFF and Industry 4.0, Autosmolt 2025 aspire to create a foundation for future unmanned, self-rearing and cost-effective smolt production systems (Figure 2.2).

2.3 Scope of the Project Thesis

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 needs to be finished daily. To complete the tasks and to optimize production without human intervention, robots are going to be necessary. A robot arm is a type of robot manipulator connected to a base which typically performs repetitive tasks. The base may be connected to stationary or moving objects and the end-point can be fitted with any type of tools or grippers. This makes the robot arm suitable to perform a range of different tasks. One of



From the current situation of
- Manual labour and subjective assessments
- Manual management and maintenance operations
- Poor fish welfare

To the next generation of smolt production
- Optimized smolt production and autonomy
- Self-monitoring rearing tanks
- Autonomous tank operations

Figure 2.2: Figure from the presentation of the Autosmolt 2025 project by SINTEF [1].

the typical use cases for robot arms today are to perform manufacturing duties, but there are also robot arms on the Mars rover performing tasks with minimal human interaction. There are multiple use cases for robot arms in smolt production as well.

This project thesis will be a continuation of a master thesis[15] that explored the possibility for automated operations using a robot arm in smolt production facilities. Selected operations for automation were cleaning of tanks, controlled feeding and feeding patterns, and removal of dead fish and waste. To perform these operations, a robot arm produced by Blueprint Lab, called Reach Bravo 7, was chosen on the basis of a comparative study. Bravo 7 preferred because it has (relatively) decent reach, is lightweight, can be used underwater and has electric motors. Simulations of the robot arm were created using Robot Operating System(ROS), Computer Aided Drawing(CAD) and Unified Robot Description Format(URDF).

Rather than using existing tools, as was done in the earlier master thesis, the scope of this thesis will involve a more mathematical approach for calculation of the kinematic and dynamic models for Reach Bravo 7. In creating general mathematical models for the Bravo 7 robot arm, the aim is to map potential challenges as a basis for future work and to build a theoretical foundation for more advanced models.

Part II

Theory

Chapter 3

Abstract Modelling of Robot Manipulators

This chapter covers the setup and symbolic representation of how robot manipulators typically are presented for creating mathematical models. Section 3.3 is important because it summarize and exemplify the way robot manipulators will be described in terms of coordinate frames in the remaining chapters.

3.1 Robot Manipulator Set-up

Manipulators consist of nearly rigid links, which are connected by joints to form a chain of links. The joints establish conjunctions in the chain that allows for relative motion of neighboring links. The joints are normally fitted with position sensors that make it possible to measure the relative position of the adjacent links. In the case of rotary or revolute joints, these displacements are called joint angles. Some manipulators include sliding (or prismatic) joints in which a translation, also called offset, is the relative displacement between the links. Both types of joints are assumed to have one degree of freedom (DOF). There exists more complicated joints, but they are normally represented as a combination of revolute and prismatic joints. Figure 3.1 depicts how revolute and translation joints are typically illustrated. The links represents the spatial relationship between two joints and are illustrated as simple lines between the joints.

3.2 Symbolic Representation

Figure 3.2a shows a typical three DOF kinematic arrangement of a robot arm. This manipulator is defined as a spherical geometric type arm [2] because it consists of two revolute joints followed by a prismatic joint. Figure 3.2b is a slightly altered version of 3.2a with a non-straight link. Such links are atypical because they tend make the modelling of the robot manipulator more involved. Independent the setup, the robot arm always have a base and an end-effector. A robot arm is always fixed

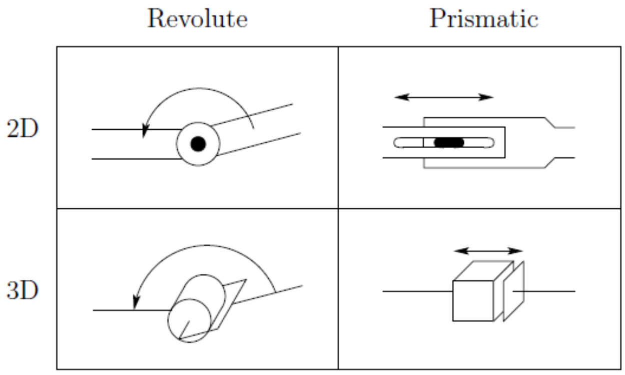
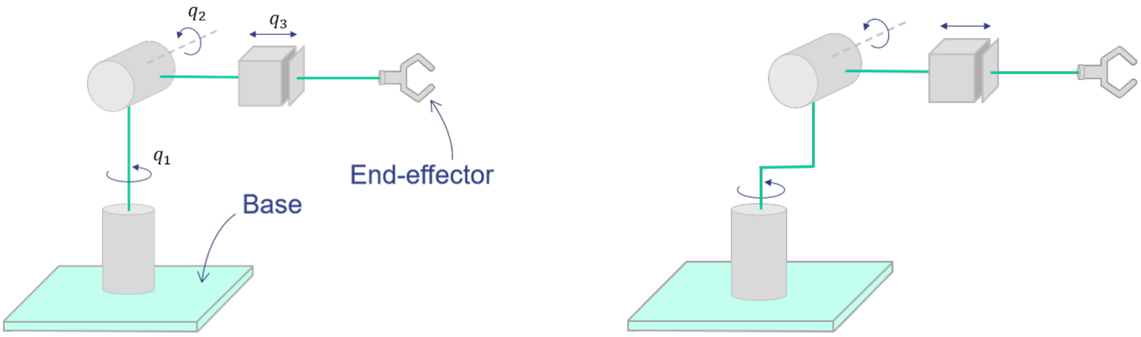


Figure 3.1: Symbolic illustrations of the different types of joints for a robot manipulator [2].



(a) Symbolic representation of spherical robot arm.

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

Figure 3.2: Symbolic representation of two different spherical robot arms.

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. Based on Figure 3.2b, two basic definitions within robot modelling are defined below.

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 varaible is denoted as $q_1 \cdots q_n$ for a n DOF robot manipulator.

Configuration space: The set of all possible combinations of joint variable values. for the robot manipulator.

Workspace: The space in which the end-effector can be positioned to perform a task. Constrained by the geometry of the manipulator as well as by the mechanical limitations of the joints.

3.3 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 3.3 the inertial frame is established at the base of the robot arm and its origin is symbolically given by 3.1.

$$\{O_0\} = (x_o, y_o, z_o) \tag{3.1}$$

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 manipualtor with n joints, the position of the end-effector is given by 3.2.

$$\boldsymbol{o}_n^0 = \begin{bmatrix} x_n^0 & y_n^0 & z_n^0 \end{bmatrix}^T \tag{3.2}$$

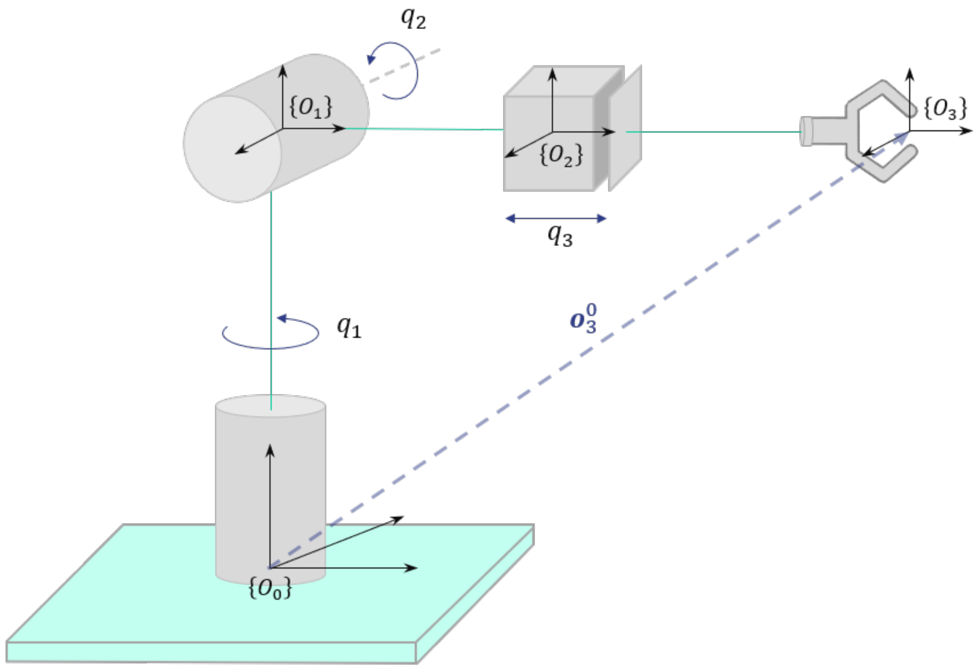


Figure 3.3: Symbolic representation of spherical robot arm with added coordinate frames.

Chapter 4

Kinematics

Kinematics within robotics is the study of the robot structure and its motion taking into account only geometric properties. The description is strictly geometric and consequently the forces and torques are ignored. The goal of the kinematic analysis is to form a relation between the joint variables and the position and orientation of different parts of the robot structure. Visualizing how one joint affects the position and orientation of connected joints and links is a convoluted effort, especially when the number of joints and links increases. Therefore, a systematic approach is required. The systematic approach which describes the cumulative effect of the joints throughout the kinematic chain is the foundation for solving the problems of forward and inverse kinematics.

The theory in this chapter primarily based on the book *Robot Modeling and Control* by Spong et al. [2].

4.1 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. When the number of joints increase, the kinematic analysis becomes more complex. The Denavit-Hartenberg convention was introduced to simplify the kinematic analysis

Denavit-Hartenberg Convention

The Denavit-Hartenberg convention is a systematic way of communicating the geometrical structure of a kinematic chain of links and joints. Central of the convention is the attachment of the coordinate frames. The convention requires consistency when defining the joint frames, but the frame placements are not unique. 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 [16], is practiced. Following any of the DH conventions lead to simplifications of the kinematic modelling.

Preliminary rules

To avoid misconceptions about the labeling scheme, a set of preliminary rules are defined. 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 [17].

- Link 0 is equal to the base of the manipulator.
- Joint i connects link $i - 1$ and moves them relative to each other.
- Link l connects joint l to joint $l + 1$.
- 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.

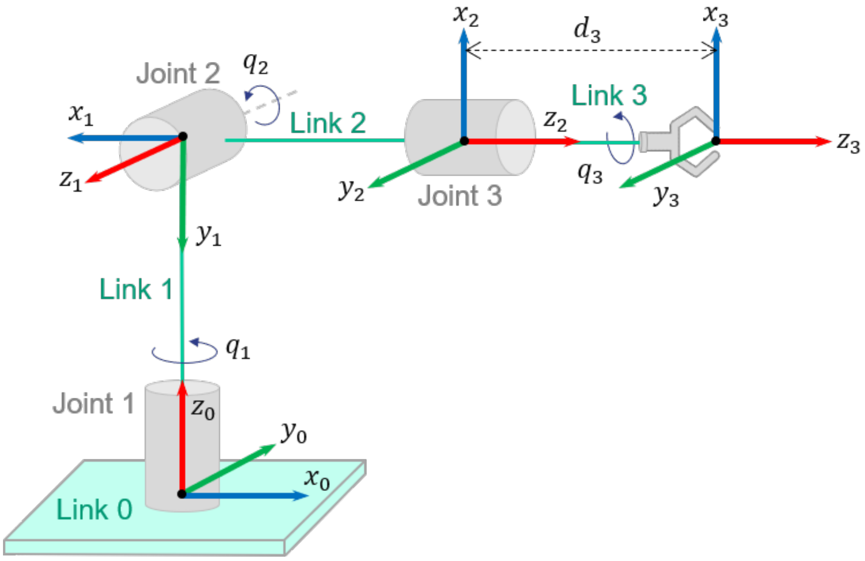


Figure 4.1: Symbolic representation of spherical wrist with added coordinate frames according to the DH-convention.

Assigning the coordinate frames

Frame rules based on the DH-convention in [2]:

- 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.

The DH-convention is exemplified in Figure 4.1 following the preliminary rules and the rules for frame placement. Every link has its own coordinate frame which is connected to the far end of the link.

Denavit-Hartenberg Parameters

Referencing Figure 4.1, the following DH-parameters are defined in terms of the distal variant 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 4.1: Explanation of the Denavit Hartenberg parameters.

Denavit-Hartenberg Table

A DH-table is a summary of the DH-parameter values defined in table 4.1. There are four parameters associated with every DH-table. The table communicates the individual geometrical description and the possible movements of the robot arm.

Link	d_i	θ_i	a_i	α_i
1	0	q_1	$-\pi/2$	0
2	0	q_2	$\pi/2$	0
3	d_3	q_3	0	0

Table 4.2: The DH table for the spherical wrist from Figure 4.1.

The DH table for the spherical wrist from Figure 4.1 is presented in Table 4.2.

Spherical Wrist

Today, a lot of the industrial robots have a setup including a spherical wrist. This is a special type of 6 DOF manipulator that is possible to divide into two parts. The first three joints are called the articulated part while the last three joints are called the spherical wrist. 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. Thus, instead of having to consider six joints, the model can focus on four.

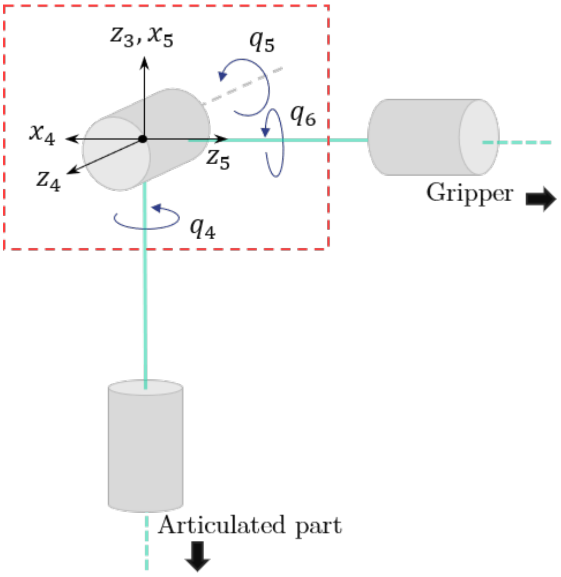


Figure 4.2: Spherical wrist, intersecting join axes in common point.

Transformation Matrix

A rigid object in a 3-dimensional space is described in terms of both position and orientation. The pose of the end-effector relative to the inertial frame is given by a transformation matrix. The transformation matrices are mathematical representations of the transformation from one coordinate frame to another. The transformation matrix of a robot manipulator relates linear and rotational movements of a joint, affecting the neighboring links. These link-transformations can be combined of all the links between frame 0 and frame n. Frame 0 cane be swapped out with any frame from 0 to $n-1$. The combined transformation-matrix describes the positional and rotational "awareness" of the origin of frame n relative to frame 0.

$$\boldsymbol{T}_n^0 = \begin{bmatrix} \boldsymbol{R}_n^0 & \boldsymbol{o}_n^0 \\ \mathbf{0}_{1\times 3} & 1 \end{bmatrix} \tag{4.1}$$

$\boldsymbol{T}_n^0 \in \mathcal{R}^{4\times 4}$ 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 (4.2)

$$\boldsymbol{T}_n^0 = \begin{cases} \boldsymbol{A}_{k+1}\boldsymbol{A}_{k+2}\dots\boldsymbol{A}_{j-1}\boldsymbol{A}_j & \text{if } k < j \\ \boldsymbol{I} & \text{if } k = j \end{cases} \tag{4.2}$$

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.

$$\boldsymbol{A}_i = \begin{bmatrix} \boldsymbol{R}_1^{i-1} & \boldsymbol{o}_i^{i-1} \\ \mathbf{0} & 1 \end{bmatrix} \tag{4.3}$$

Solution to the Forward Kinematics Problem

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

$$\boldsymbol{A}_i(q_i) = \begin{bmatrix} c_{\theta_i} & -s_{\theta_i}c_{\alpha_i} & s_{\theta_i}s_{\alpha_i} & a_i c_{\theta_i} \\ s_{\theta_i} & c_{\theta_i}c_{\alpha_i} & -c_{\theta_i}s_{\alpha_i} & a_i s_{\theta_i} \\ 0 & s_{\alpha_i} & c_{\alpha_i} & d_i \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{4.4}$$

At this point, the resulting transformation matrix $\boldsymbol{T}_e^0$ from (4.2) describes the transformation from link-frame 0 to link-frame e involving only the joint variables as the unknown. In other words, the solution to the forward kinematics problem.

4.2 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 normally are governed by a set of 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 the within its workspace. Environment interaction requires the pose of the object in relation to the robot inertial frame. 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 matching 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, and the most effective approach depends the robot manipulator and its level of complexity.

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. 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 (4.5), 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.

$$\boldsymbol{T}_e^0(\boldsymbol{q}) = \begin{bmatrix} r_{11} & r_{12} & r_{13} & x_e^0 \\ r_{21} & r_{22} & r_{23} & y_e^0 \\ r_{31} & r_{32} & r_{33} & z_e^0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{4.5}$$

Not all robot arms have an analytical solution. There are publications which conclude that 6-DOF robot arms are proving to have a closed-form solution [18]. However, this criterion, 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.

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. 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 this thesis. The optimization solution is based on the DH-convention.

The setup of the numerical optimization approach is relatively straight forward. 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 [17]. A formal mathematical formulation of the optimization problem is to minimize the error between the forward kinematics solution and the desired pose $\boldsymbol{T}^*(\boldsymbol{q}^*)$

$$\text{Minimize } \boldsymbol{E(q)} = \left\| [\boldsymbol{T(q)}]_t - \boldsymbol{T}^*(\boldsymbol{q}^*) \right\| \tag{4.6}$$

subjected to workspace and geometrical constraints. The constraints is an attempt to avoid 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 [18].

The numerical solution has three main drawbacks.

1. Slow for practical applications.
2. Not always able to find all the possible solutions.
3. Might be unstable. Singularities occur when the robot is outside its workspace or in an impossible configuration.

Because the analytical approach has poor scalability and the numerical solution is unusable for most real-time problems, there are currently research invested in finding better and more generalized solutions. It is worth to mention that Jacobian-based inverse kinematics solver and techniques, allow for a numerical solution as an efficient real-time solver [19]. Another interesting approach for robots satisfying the Pieper-criterion is presented in Novel [20]. With the rapid development of artificial intelligence, this is mentioned as a key technology for further developments within robotics. Two examples inverse kinematics solvers based on artificial intelligence are [21] and [22].

Skriv om dette: Based on the available research today, an analytical solution is always preferable. First, the solution explicitly shows the multiple configurations of the robot and secondly the solutions tend to be compact and fast to execute.

Chapter 5

Dynamics

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 [23]. The most important concepts required for the derivation of the equation of motion are mentioned in Spong [2] and described in more detail in Egeland [24].

Section 4.1 presents the equation 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 two last sections of the chapter presents the forward dynamics (section 4.3) and the implicit ordinary differential equation (ODE) form of the equation of motion (section 4.4.). The theory in section 4.3 is useful for mathematical simulations of the robot dynamics.

The theory of this chapter is mainly based on Spong the work of [2], Egeland and Gravdal [24] and Siciliano [23].

5.1 Equation 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 external forces acting on its rigid links. For a robot arm each link supports the outward links. Therefore, link i is supported by link $i-1$. The mechanical connection between the links comes in the form of force and torque exerted by neighboring links. Joint torques are applied from the actuators in each joint and can be controlled to a achieve a particular manipulator state. Figure 5.1 demonstrates a simple two dimensional robot arm and its corresponding freebody 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 working environment, factors which are included the equation of motion.

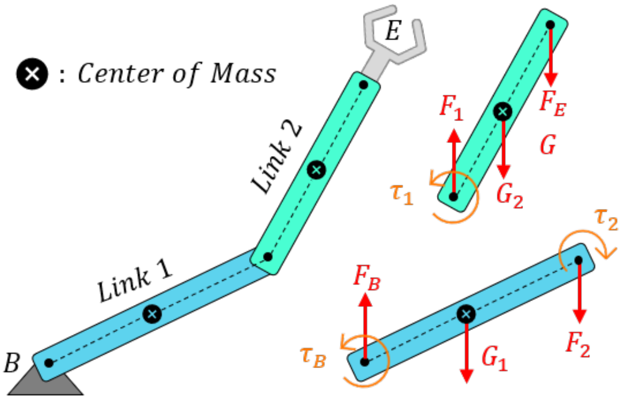


Figure 5.1: Simple 2-DOF 2D robot arm with coresponding freebody diagram.

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.

Euler-Lagrange Method :	Newton-Euler Method :
Energy based approach.	Newton's second law based approach (force balance).
Manipulator treated as one system.	Each link of the manipulator is treated in turn.

Newton-Euler and the Euler-Lagrange formulations lead to the same dynamic model (5.1), but the route to get there is different. Spong [2] structures the equation of motion as

$$\boldsymbol{M}(\boldsymbol{q})\ddot{\boldsymbol{q}} + \boldsymbol{C}(\boldsymbol{q}, \dot{\boldsymbol{q}})\dot{\boldsymbol{q}} + \boldsymbol{G}(\boldsymbol{q}) = \boldsymbol{\tau} \tag{5.1}$$

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.

The choice of method to derive (5.1) may be based on personal preference. Nonetheless, there are cases where one approach might be preferred over the other. For real-time control and simulation the equations of Newton-Euler are often favored, if working with Lyapunov designs or passivity, the energy based Euler-Lagrange equations might be suitable [24].

5.2 Euler-Lagrange Method

The Euler-Lagrange method is one alternative often used to derive the dynamic model. The 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

$$\mathcal{L} = \mathcal{K} - \mathcal{P}. \tag{5.2}$$

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 [23].

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 join variables [2]. The linear and angular velocities are expressed in terms of the Jacobian matrices in the form of (5.3) and (5.4) respectively.

$$\boldsymbol{J}_v = [\boldsymbol{J}_{v_1} ... \boldsymbol{J}_{v_n}] \tag{5.3}$$

$$\boldsymbol{J}_\omega = [\boldsymbol{J}_{\omega_1} ... \boldsymbol{J}_{\omega_n}] \tag{5.4}$$

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

$$\boldsymbol{J}_{v_i} = \begin{cases} \boldsymbol{z}_{i-1} \times (\boldsymbol{o}_n - \boldsymbol{o}_{i-1}) & \text{for revolute joint } i \\ \boldsymbol{z}_{i-1} & \text{for prismatic joint } i \end{cases} \tag{5.5}$$

$$\boldsymbol{J}_{\omega_i} = \begin{cases} \boldsymbol{z}_{i-1} & \text{for revolute joint } i \\ \boldsymbol{0} & \text{for prismatic joint } i \end{cases} \tag{5.6}$$

where z_i and o_n can be found directly from the transformation matrices found in the forward kinematics problem, see (5.7).

$$\begin{cases} \boldsymbol{z}_i & \text{given by the first three elements in the third column of } \boldsymbol{T}_i^0 \\ \boldsymbol{o}_i & \text{given by the first three elements in the fourth column of } \boldsymbol{T}_i^0 \end{cases} \tag{5.7}$$

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 ($\boldsymbol{D}_i$) by using the velocity Jacobians, the inertia tensor ($\boldsymbol{I}_i$) and the rotation matrix ($\boldsymbol{R_i}$) of the link.

$$\boldsymbol{D}_i(\boldsymbol{q}) = m_i \boldsymbol{J}_{v_i}(\boldsymbol{q})^T \boldsymbol{J}_{v_i}(\boldsymbol{q}) + \boldsymbol{J}_{\omega_i}(q)^T \boldsymbol{R}_i(\boldsymbol{q}) \boldsymbol{I}_i \boldsymbol{R}_i(\boldsymbol{q})^T \boldsymbol{J}_{\omega_i}(\boldsymbol{q}) \tag{5.8}$$

$\boldsymbol{I}_i$ is the inertia tensor in the body attached frame of link i, but evaluated around the link center of mass (see figure 5.1). The matrix values are only dependent on the geometrical configuration of the rigid body [2]. Consequently, the inertia matrix given by (5.9) is constant and independent of any motion. Furthermore, the value of the inertia matrix is typically included in the robot documentation.

$$\boldsymbol{I}_i = \left. \begin{bmatrix} I_{xx} & I_{xy} & I_{xz} \\ I_{yx} & I_{yy} & I_{yz} \\ I_{zx} & I_{zy} & I_{zz} \end{bmatrix} \right|_i \tag{5.9}$$

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

$$\boldsymbol{D}(\boldsymbol{q}) = \sum_{i=1}^{n} \boldsymbol{D}_i(\boldsymbol{q}) \tag{5.10}$$

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

$$\boldsymbol{\mathcal{K}} = \frac{1}{2} \dot{\boldsymbol{q}}^T \boldsymbol{D}(\boldsymbol{q}) \dot{\boldsymbol{q}} \tag{5.11}$$

where

$$\boldsymbol{q}^T = [q_1, q_2, \cdots, q_n]. \tag{5.12}$$

and

$$\dot{\boldsymbol{q}}^T = [\dot{q}_1, \dot{q}_2, \cdots, \dot{q}_n]. \tag{5.13}$$

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 5.1.

$$\boldsymbol{\mathcal{P}}_i = \boldsymbol{g}_0^T \boldsymbol{r}_{ci} m_i \tag{5.14}$$

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

$$\boldsymbol{\mathcal{P}} = \sum_{i=1}^{n} \boldsymbol{g}_0^T \boldsymbol{r}_{ci} m_i \tag{5.15}$$

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 [2]. By denoting the vectors as

$$\boldsymbol{r}_{ci} = [r_{cx},\ r_{cy},\ r_{cz}]^T \tag{5.16}$$

and

$$\boldsymbol{g}_0 = [g_x,\ g_y,\ g_z]^T \tag{5.17}$$

then $\boldsymbol{g}_0 = [0,\ 0,\ -9.81]^T$ if z is the vertical axis pointing upwards.

TODO: Add something about hydrodynamics

Euler-Lagrange Equation

Based on the Lagrangian equation (5.2), the Euler-Lagrange equation for an n-link manipulator is calculated as

$$\frac{d}{dt}\frac{\boldsymbol{\partial \mathcal{L}}}{\boldsymbol{\partial \dot{q}}} - \frac{\boldsymbol{\partial \mathcal{L}}}{\boldsymbol{\partial q}} = \boldsymbol{\tau}. \tag{5.18}$$

.

Here, $\boldsymbol{\tau}$ is a vector (5.19) which represents the input generalized forces equal to the motor torques at the joints [24]. The torque τ_k is associated with the generalized coordinate q_k [23].

$$\boldsymbol{\tau} = \begin{bmatrix} \tau_1 \\ \tau_2 \\ \vdots \\ \tau_n \end{bmatrix} \tag{5.19}$$

From the calculation of the Euler-Lagrange equation [2], it is possible to rewrite (5.18) as

$$\sum_{i=1}^{n} D_{kj}(\boldsymbol{q})\ddot{q}_j \ + \sum_{i,j=1}^{n} c_{ijk}(\boldsymbol{q})\dot{q}_i\dot{q}_j + \phi_k(\boldsymbol{q}) \ = \tau_k, \ k = 1, ..., n \tag{5.20}$$

where ϕ_k and D_{kj} may be calculated individually, and C_{ijk} (known as Christoffel symbols [2]) can be found from D_{kj}, see (5.21 - 5.24) .

$$\phi_k(\boldsymbol{q}) = \frac{\partial \mathcal{P}}{\partial q_k} \tag{5.21}$$

$$D_{kj}(\boldsymbol{q}) = \frac{\partial(\frac{\partial \mathcal{K}}{\partial \dot{q}_j})}{\partial \dot{q}_k} \tag{5.22}$$

$$c_{ijk}(\boldsymbol{q}) = \frac{1}{2}\left\{\frac{\partial D_{kj}}{\partial q_i} + \frac{\partial D_{ki}}{\partial q_j} - \frac{\partial D_{ij}}{\partial q_k}\right\} \tag{5.23}$$

$$C_{kj} = \sum_{i=1}^{n} c_{ijk}(q)\dot{q}_i \ = \frac{1}{2}\sum_{i=1}^{n}\left\{\frac{\partial D_{kj}}{\partial q_i} + \frac{\partial D_{ki}}{\partial q_j} - \frac{\partial D_{ij}}{\partial q_k}\right\}\dot{q}_i \tag{5.24}$$

In the case of manipulators of higher complexity, i.e. with 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 (5.1) is presented in section 4.1.

The correlation between (5.1) and (5.20) implies the following characteristics

ϕ_k = the k-th element of the gravity vector $\boldsymbol{G(q)} \in \mathbb{R}^n$
D_{kj} = the k, j-th element of the manipulator inertia matrix $\boldsymbol{D}(\boldsymbol{q}) \in \mathbb{R}^{n \times n}$
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 5.25.

$$\begin{bmatrix}\tau_1\\ \tau_2\\ \vdots\\ \tau_n\end{bmatrix} = \underbrace{\begin{bmatrix}D_{11} & D_{12} & \dots & D_{1n}\\ D_{21} & D_{22} & \dots & D_{2n}\\ \vdots & \ddots & \ddots & \vdots\\ D_{n1} & D_{n2} & \dots & D_{nn}\end{bmatrix}}_{\boldsymbol{D(q)}} \begin{bmatrix}\ddot{q}_1\\ \ddot{q}_2\\ \vdots\\ \ddot{q}_n\end{bmatrix} + \underbrace{\begin{bmatrix}C_{11} & C_{12} & \dots & C_{1n}\\ C_{21} & C_{22} & \dots & C_{2n}\\ \vdots & \ddots & \ddots & \vdots\\ C_{n1} & C_{n2} & \dots & C_{nn}\end{bmatrix}}_{\boldsymbol{C(q,\dot{q})}} \begin{bmatrix}\dot{q}_1\\ \dot{q}_2\\ \vdots\\ \dot{q}_n\end{bmatrix} + \underbrace{\begin{bmatrix}\phi_1\\ \phi_2\\ \vdots\\ \phi_n\end{bmatrix}}_{\boldsymbol{G(q)}} \tag{5.25}$$

There are a couple clarifying notes to be made about this equation. Firstly, $\boldsymbol{D(q)}$ can be computed either as done in (5.10) or (5.22). Secondly, the inertia matrix ($\boldsymbol{M}$) is equal to D. This is because Spong [2] define $\boldsymbol{M}$ as the inertia matrix with the added effects of the gear inertia ($\boldsymbol{J_m}$). Thus, the inertia matrix is dependent on the value of the gear ratio as well.

$$\boldsymbol{M(q)} = \boldsymbol{D(q)} + \boldsymbol{J_m} \tag{5.26}$$

Similar to Spong, in this thesis $\boldsymbol{J_m}$ is ignored, which means $\boldsymbol{D(q)} = \boldsymbol{M(q)}$.

5.3 Forward Dynamics

The matrix form of the Euler-Lagrange equation (5.1) is referred to as inverse dynamics because the equation setup maps motion to torque (5.27). This is useful for determining control laws [2].

$$\boldsymbol{(q, \dot{q}, \ddot{q})} \rightarrow \boldsymbol{\tau} \tag{5.27}$$

Forward dynamics is the other way around, where the joint acceleration $\boldsymbol{\ddot{q}}$ 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 (5.28).

$$\boldsymbol{\tau} \rightarrow \boldsymbol{(q, \dot{q}, \ddot{q})} \tag{5.28}$$

To be able to define the motion, based on the torque values, the Euler-Lagrange equation needs to be altered. The joint accelerations (5.29) are found based on the inverse of the inertia matrix [2].

$$\boldsymbol{\ddot{q}} = \boldsymbol{M}^{-1}\boldsymbol{(q)}\big[\boldsymbol{\tau} - \boldsymbol{C(q, \dot{q})\dot{q}} - \boldsymbol{g(q)}\big] \tag{5.29}$$

The form of (5.29) is the forward dynamics model in its explicit form where $\boldsymbol{M}$ is invertible[2]. $\boldsymbol{M(q)}$ might be a big and complex matrix, hence the inverse can become exceedingly complex [24]. In this case, having the model in its implicit form (5.30) or working with $\boldsymbol{M(q)}^{-1}$ numerically is sometimes required.

$$\boldsymbol{M}(\boldsymbol{q})\ddot{\boldsymbol{q}} = [\boldsymbol{\tau} - \boldsymbol{C}(\boldsymbol{q}, \dot{\boldsymbol{q}})\dot{\boldsymbol{q}} - \boldsymbol{G}(\boldsymbol{q})] \tag{5.30}$$

Part III

Method

Chapter 6

Kinematic and Dynamic Modelling of Reach Bravo 7

Blueprint Lab is the company behind the Reach Bravo 7, a robot arm released in 2020 [25] designed to conduct inspection, maintenance and repair tasks typically reserved for human divers in subsea operations [26]. This chapter involves the available information about the Bravo 7 (without purchasing the robot arm) in an attempt to derive the kinematic and dynamic model and the challenges that followed. The information available from Blueprint Lab is a data sheet (Appendix A and [27]) and documentation (Appendix B and [28]). Additionally, a specifically requested machine drawing (Appendix C) sent by Blueprint Lab. The documentation was especially interesting as it included the kinematic and dynamic parameters.

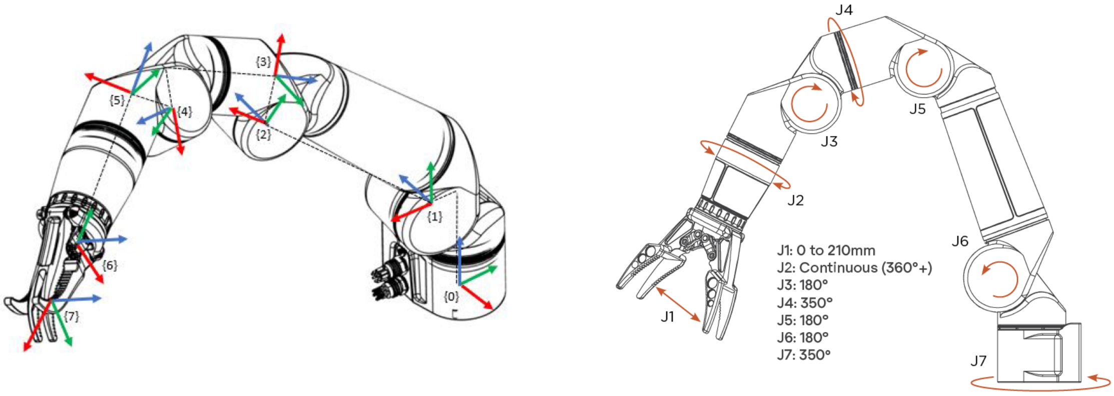
The set-up and general schematics of Bravo 7 are presented in section 5.1. In section 5.2 and 5.3 the appropriate data from the data sheet, documentation and machine drawing is used to derive the kinematics and dynamics. The last section describes how the code in MATLAB from Appendix D and E is structured to create a simulation of the dynamic model calculated in section 5.3.

The equations of this chapter are for the most part displayed in simplified symbolic forms analogous to how the equations are written in the MATLAB code. This is due to the complexity and the sheer length of some of the equations.To simplify the system equations a tiny bit, the end-effector of the robot arm is not considered in any of the derivations.

To the best of the author's knowledge, no work has been published about the kinematic or dynamic modelling of the Bravo 7, or any other type of publication involving the specific robot arm.

6.1 System Description and Schematics

The Reach Bravo 7 has a total of six revolute joints controlled by electric motors. This means it it has six degrees of freedom. Unlike most industrial robots with six degrees of freedom, the Bravo 7 has no spherical wrist. The schematics in figure



(a) Schematics of Bravo 7 with added coordinate frames (Appendix B).

(b) Schematics of Bravo 7 showing possible joint movements (Appendix A)

Figure 6.1: Two different schematics of Reach Bravo 7.

5.1 shows the position of the six joints and their possible rotational movements. In figure 6.1a coordinate frames are added at each joint ($\{0\} - \{5\}$). The last two coordinate frames $\{6\}$ and $\{7\}$ are attached at the beginning and end of the end effector, and are not attached to any type of joint, as confirmed by figure 6.1b.

The documentation from Blueprint Lab provides a DH-table consisting of the DH-parameters where $\theta_a = tan^{-1}(\frac{5.2}{293.55})$. 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.

Link	**d (mm)**	$\boldsymbol{\theta}$	**a (mm)**	$\boldsymbol{\alpha}$
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 6.1: DH-table from the documentation in Appendix B

6.2 Forward Kinematics

The forward kinematics of the Bravo 7 is based on the theory from section 4.1.

Kinematic Schematics

Following the preliminary rules and frame rules of the DH-convention (4.1), a kinematic schematic of the Bravo 7 is depicted in figure 6.2. The kinematic schematic is created based on figure 5.1 and table 6.1. The robot arm is set in its zero-configuration which is the base state of the manipulator as a function of its joint variables (Note: this is not the same zero-configuration as the one in table 6.1).

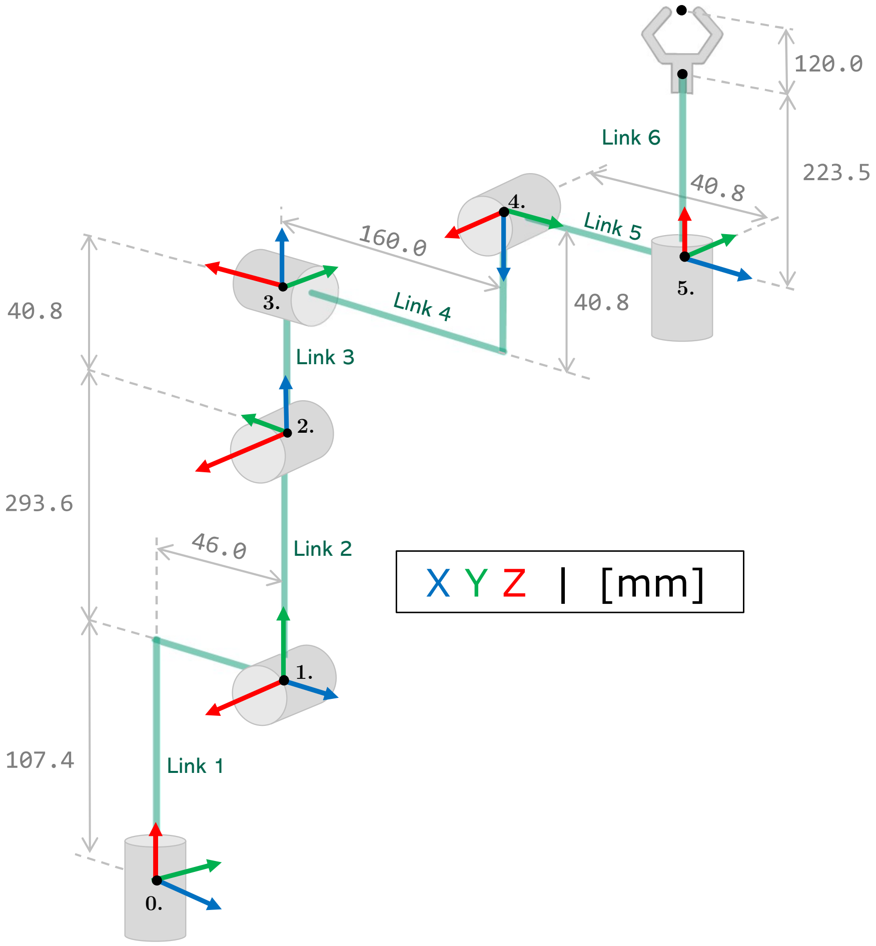


Figure 6.2: Kinematic schematic of Reach Bravo 7 in its zero-angle configuration.

Denavit-Hartenberg Table

The DH-table in table 6.2 is similar to table 6.1, but with a few alterations according to the kinematic schematics in figure 6.2. 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 [2]. Another alteration is regarding the joint variables θ. The joint values are set as simple variables while disregarding the equations in table 6.1 with the intention of simplifying the model equations. The simplification is also a consequence of the lack of details regarding the derivation of the DH-table in the documentation. When making more advanced models which adhere to the physical constraints of the robot arm, the DH-table created by Blueprint Lab are probably better suited to model the real physical properties of the Bravo 7.

Link	d_i	θ_i	a_i	α_i
1	0.1074	$\boldsymbol{\theta_1}$	0.046	$^{\pi}/_{2}$
2	0	$\boldsymbol{\theta_2}$	0.2936	0
3	0	$\boldsymbol{\theta_3}$	0.0408	$-^{\pi}/_{2}$
4	- 0.160	$\boldsymbol{\theta_4}$	0.0408	$-^{\pi}/_{2}$
5	0	$\boldsymbol{\theta_5}$	0.0408	$-^{\pi}/_{2}$
6	- 0.2235	0	0	$^{\pi}/_{2}$

Table 6.2: Altered and simplified DH table directly based on figure 6.2.

Forward Kinematics Solution

As mentioned in section 4.1, 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 6.2 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 6.2 to be $\boldsymbol{q} = [\theta_1, \theta_2, \theta_3, \theta_4, \theta_5, 0]^T$.

$$\boldsymbol{T_6^0}(\boldsymbol{q}) = \boldsymbol{A}_1\boldsymbol{A}_2\boldsymbol{A}_3\boldsymbol{A}_4\boldsymbol{A}_5\boldsymbol{A}_6 = \begin{bmatrix} r_{11} & r_{12} & r_{13} & x_6^0 \\ r_{21} & r_{22} & r_{23} & y_6^0 \\ r_{31} & r_{32} & r_{33} & z_6^0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{6.1}$$

where

$$\boldsymbol{A}_1 = \begin{bmatrix} c_1 & 0 & s_1 & a_1c_1 \\ s_1 & 0 & -c_1 & a_1s_1 \\ 0 & 1 & 0 & d_1 \\ 0 & 0 & 0 & 1 \end{bmatrix}, \boldsymbol{A}_2 = \begin{bmatrix} c_2 & -s_2 & 0 & a_2c_2 \\ s_2 & c_2 & 0 & a_2s_2 \\ 0 & 1 & 0 & d_2 \\ 0 & 0 & 0 & 1 \end{bmatrix}, \boldsymbol{A}_3 = \begin{bmatrix} c_3 & 0 & -s_3 & a_3c_2 \\ s_3 & 0 & c_3 & a_3s_3 \\ 0 & -1 & 0 & d_3 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

$$\boldsymbol{A}_4 = \begin{bmatrix} c_4 & 0 & -s_4 & a_4c_4 \\ s_4 & 0 & c_4 & a_4s_4 \\ 0 & -1 & 0 & d_4 \\ 0 & 0 & 0 & 1 \end{bmatrix}, \boldsymbol{A}_5 = \begin{bmatrix} c_5 & 0 & -s_5 & a_5c_5 \\ s_5 & 0 & c_5 & a_5s_5 \\ 0 & -1 & 0 & d_5 \\ 0 & 0 & 0 & 1 \end{bmatrix}, \boldsymbol{A}_5 = \begin{bmatrix} c_6 & 0 & s_6 & a_6c_6 \\ s_6 & 0 & -c_6 & a_6s_6 \\ 0 & 1 & 0 & d_6 \\ 0 & 0 & 0 & 1 \end{bmatrix}.$$

The transformation matrix in (6.1) 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 6.2, 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. In Appendix D, there is MATLAB code where line 1 to 68 demonstrates a symbolic solution to the forward kinematics problem of Bravo 7 using the MATLAB Symbolic Toolbox [29]

6.3 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 therefore does not satisfy the Pieper criterion. As mentioned in section 4.2, a spherical wrist simplifies the derivation of the inverse kinematics and there is a lower probability that an analytical solution exists.

Below are three different approaches attempted to find the inverse kinematic solution for Bravo 7.

Geometrical solution:

The geometrical solution for a complex 6 DOF robot arm as the Bravo 7 is probably a near impossible task, but finding a solution for the three first joints is not. In Spong [2] a geometrical solution for an elbow manipulator is exemplified. The three first joints of the Bravo 7 is very similar to the elbow manipulator, although the first link in the Bravo 7 is non-straight, thus involving an extra offset in the geometrical structure. The ability to find a geometrical solution often requires practice and experience due to the lack of generalization withing the method. Hence, inspiration from similar examples can be a deciding when finding a solution.

IKBT method (algebraic solution):

The authors of the IKBT (Inverse Kinematics Behavioural Three) method [22] have added tutorial videos on how to add your own manipulators in their framework on the IKBT *GitHub* page [30]. The procedure is unambiguous involving only the addition of the manipulator's DH-table. If the program finds a solution, there is an option to create a PDF document which presents each solution step.

Optimization method (numerical solution):

The numerical solution used on the Bravo 7 involves the same mathematical formulation as the optimization method introduced in section 4.2. A Matlab code example is added in Appendix G. The code uses the `fsolve` function from the *Optimization Toolbox* [31] with the *Levenberg-Marquardt* algorithm, a nonlinear least-squares algorithm solver [32].

6.4 Dynamics

The Euler-Lagrange approach for determining the dynamic model was derived in section 5. Section 6.2 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 are used for finding a symbolic representation of the dynamic model in this section.

Kinetic Energy

The kinetic energy of Bravo 7 is computed by using the the velocity Jacobians, inertia tensor, rotation matrix, joint variable and mass of all the links to find the inertia matrix (5.10). The mass and inertia tensors are constants found in table 2 in Appendix B where link j in the table equals link $j+1$ from section 6.2. As the robot only consists of revolute joints, the velocity Jacobians are computed the same way for every joint, see equation (5.3) and (5.4). By reference to (4.1), (4.2) and (5.5), the velocity Jacobian and rotation matrix are found from the transformation matrix for the individual links relative to the base frame.

Link 1

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

$$\boldsymbol{J}_{v_1} = \boldsymbol{z}_0 \times (\boldsymbol{o}_6 - \boldsymbol{o}_0) \tag{6.2}$$

$$\boldsymbol{J}_{\omega_1} = \boldsymbol{z}_0 \tag{6.3}$$

$$\boldsymbol{R}_1 = \boldsymbol{R}_1^0 \tag{6.4}$$

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

Link 2

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

$$\boldsymbol{J}_{v_2} = \boldsymbol{z}_1 \times (\boldsymbol{o}_6 - \boldsymbol{o}_1) \tag{6.5}$$

$$\boldsymbol{J}_{\omega_2} = \boldsymbol{z}_1 \tag{6.6}$$

$$\boldsymbol{R}_2 = \boldsymbol{R}_2^0 \tag{6.7}$$

Link 3

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

$$\boldsymbol{J}_{v_3} = \boldsymbol{z}_2 \times (\boldsymbol{o}_6 - \boldsymbol{o}_2) \tag{6.8}$$

$$\boldsymbol{J}_{\omega_3} = \boldsymbol{z}_2 \tag{6.9}$$

$$\boldsymbol{R}_3 = \boldsymbol{R}_3^0 \tag{6.10}$$

Link 4

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

$$\boldsymbol{J}_{v_4} = \boldsymbol{z}_3 \times (\boldsymbol{o}_6 - \boldsymbol{o}_3) \tag{6.11}$$

$$\boldsymbol{J}_{\omega_4} = \boldsymbol{z}_3 \tag{6.12}$$

$$\boldsymbol{R}_4 = \boldsymbol{R}_4^0 \tag{6.13}$$

Link 5

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

$$\boldsymbol{J}_{v_5} = \boldsymbol{z}_4 \times (\boldsymbol{o}_6 - \boldsymbol{o}_4) \tag{6.14}$$

$$\boldsymbol{J}_{\omega_5} = \boldsymbol{z}_4 \tag{6.15}$$

$$\boldsymbol{R}_5 = \boldsymbol{R}_5^0 \tag{6.16}$$

Link 6

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

$$\boldsymbol{J}_{v_6} = \boldsymbol{z}_5 \times (\boldsymbol{o}_6 - \boldsymbol{o}_5) \tag{6.17}$$

$$\boldsymbol{J}_{\omega_6} = \boldsymbol{z}_5 \tag{6.18}$$

$$\boldsymbol{R}_6 = \boldsymbol{R}_6^0 \tag{6.19}$$

The resulting symbolic representation of the inertia matrix is too massive to be included in the thesis, but in n line 143 to 146 of the Appendix D Matlab code the matrix is used to calculate the kinetic energy for each link.

Potential Energy

The computation of potential energy only depends on the mass, the center of mass coordinate vector and the gravity acceleration. The coordinate vector for the center of mass for every link is to be found in table 2 in Appendix B. This coordinate vector is in reference to the link frame and not the reference frame. Thus, the coordinate vector $\boldsymbol{r}_{c_i}$ in 5.15 is found by transformation of the vector from the link frame to the base frame as below.

$$\boldsymbol{r}_{c_1} = \boldsymbol{r}_{c_1}^0 = \boldsymbol{R}_1^0 \boldsymbol{r}_{c_1}^1 \tag{6.20}$$

$$\boldsymbol{r}_{c_2} = \boldsymbol{r}_{c_2}^0 = \boldsymbol{R}_2^0 \boldsymbol{r}_{c_2}^2 \tag{6.21}$$

$$\boldsymbol{r}_{c_3} = \boldsymbol{r}_{c_3}^0 = \boldsymbol{R}_3^0 \boldsymbol{r}_{c_1}^3 \tag{6.22}$$

$$\boldsymbol{r}_{c_4} = \boldsymbol{r}_{c_4}^0 = \boldsymbol{R}_4^0 \boldsymbol{r}_{c_4}^1 \tag{6.23}$$

$$\boldsymbol{r}_{c_5} = \boldsymbol{r}_{c_5}^0 = \boldsymbol{R}_5^0 \boldsymbol{r}_{c_5}^5 \tag{6.24}$$

$$\boldsymbol{r}_{c_6} = \boldsymbol{r}_{c_6}^0 = \boldsymbol{R}_6^0 \boldsymbol{r}_{c_6}^1 \tag{6.25}$$

In the Matlab code (Appendix D) the computation of the potential energy is performed in line 143 to 146 with the center of mass in reference to the base vector and $\boldsymbol{g}_0 = [0, 0, g_z]^T$.

Equation of Motion

The symbolic equation for the equation of motion (5.1) with $\boldsymbol{q}$, $\boldsymbol{q}$ and $\ddot{\boldsymbol{q}}$ as variables requires the inertia matrix($\boldsymbol{M}$), centrifugal and Coriolis terms matrix ($\boldsymbol{C}$), and gravity vector($\boldsymbol{G}$) to be computed. The inertia matrix is previously calculated, 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

$$\boldsymbol{G} = \begin{bmatrix} \frac{\partial \mathcal{P}}{\partial q_1} \\ \frac{\partial \mathcal{P}}{\partial q_2} \\ \frac{\partial \mathcal{P}}{\partial q_3} \\ \frac{\partial \mathcal{P}}{\partial q_4} \\ \frac{\partial \mathcal{P}}{\partial q_5} \\ \frac{\partial \mathcal{P}}{\partial q_6}. \end{bmatrix} \tag{6.26}$$

The computation of the gravity vector in the Matlab code (Appendix D) takes place at the lines 175-180.

The centrifugal and Coriolis terms matrix is not as straight forward to calculate as the gravity vector. 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 (5.24).

$$C_{kj} = \frac{1}{2} \sum_{i=1}^{6} \left\{ \frac{\partial D_{kj}}{\partial q_i} + \frac{\partial D_{ki}}{\partial q_j} - \frac{\partial D_{ij}}{\partial q_k} \right\} \dot{q}_i \tag{6.27}$$

With the $\boldsymbol{G}$ and $\boldsymbol{C}$ readily calculated, the left side of the equation of motion (5.1) is complete. In the Matlab code (Appendix D), $\boldsymbol{M(q)}$, $\boldsymbol{C(q, \dot{q})}$ and $\boldsymbol{G(q)}$ are in their symbolic forms. By the use of *matlabFunction* (see line 182), the symbolic expressions are converted to function files with $\boldsymbol{q}$, $\dot{\boldsymbol{q}}$ and g_z as variables. The function files are then available for computation of numerical values for simulations involving the dynamics model.

The function file created by `matlabFunction` is not included in this project thesis because it has more than 8000 lines of code. The Matlab file is however included in the delivered zip-file and the function is called `getDCGfunc`, see Appendix F for more information about the code files.

Chapter 7

Simulation of the Dynamics Model

The forward dynamics model presented in 5.3 is useful to validate the model through simulations. This chapter presents the system setup for simulations of the Bravo 7 equation of motion using Simulink. Section 6.1 explains the rewriting of the forward dynamics model to a first-order nonlinear state space representation. An open loop Simulink setup is described in section 6.2, including how the results are presented graphically. The last section contain comments regarding the simplicity of the model and how it may be expanded with feedback controllers.

7.1 State Space Representation

Section 5.3 characterizes explicit form of the forward dynamics which involves the inertia matrix in its inverse form. The inertia matrix for Bravo 7 is big and complex and the simulation of the dynamics model is therefore solved numerically in a Simulink environment.

To be able to use Simulink for the simulation, the explicit forward dynamics form (5.29) has to be reduced to its first-order nonlinear state space representation. Defining the state vector $\boldsymbol{x}$ as

$$x_1 = q_1, \qquad x_2 = \dot{q}_1 = \dot{x}_1 \tag{7.1}$$

$$x_3 = q_2, \qquad x_4 = \dot{q}_2 = \dot{x}_3 \tag{7.2}$$

$$\vdots \qquad\qquad \vdots$$

$$x_{11} = q_6, \qquad x_{12} = \dot{q}_6 = \dot{x}_{11}. \tag{7.3}$$

then the dynamic model can be expressed in the form

$$\dot{\boldsymbol{x}} = \boldsymbol{F}(\boldsymbol{x}, \boldsymbol{\tau}), \qquad \boldsymbol{F} \in \mathbb{R}^{12} \tag{7.4}$$

where

$$\dot{x_1} = x_2 = F_1(x, \tau), \qquad \dot{x_2} = \ddot{q}_1 = F_2(x, \tau) \tag{7.5}$$

$$\dot{x_3} = x_4 = F_3(x, \tau), \qquad \dot{x_4} = \ddot{q}_2 = F_4(x, \tau) \tag{7.6}$$

$$\dot{x_5} = x_4 = F_5(x, \tau), \qquad \dot{x_6} = \ddot{q}_3 = F_6(x, \tau) \tag{7.7}$$

$$\dot{x_7} = x_4 = F_7(x, \tau), \qquad \dot{x_8} = \ddot{q}_4 = F_8(x, \tau) \tag{7.8}$$

$$\dot{x_9} = x_4 = F_9(x, \tau), \qquad \dot{x_{10}} = \ddot{q}_5 = F_{10}(x, \tau) \tag{7.9}$$

$$\dot{x_{11}} = x_4 = F_{11}(x, \tau), \qquad \dot{x_{12}} = \ddot{q}_6 = F_{12}(x, \tau) \tag{7.10}$$

7.2 Simulink Setup

The setup of the Simulink environment is inspired by the work in [33], but with the necessary alterations to the Matlab code to accommodate the calculations of the equation of motion from section 6.4. The setup of Simulink diagram is shown in figure 7.1.

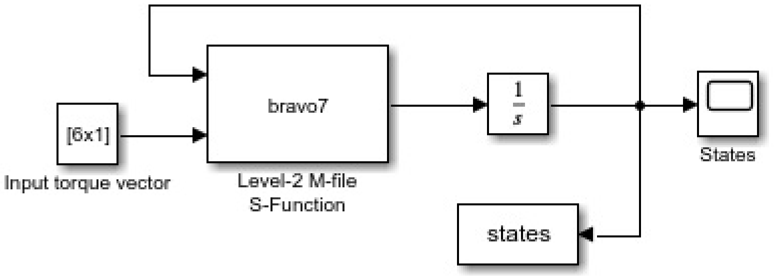


Figure 7.1: Open Loop Simulink diagram for simulation of the dynamics model.

The purpose of the Matlab code in Appendix E, is to transfer the inputs and the outputs between the Matlab and Simulink environment. A `Level-2 Matlab S-function` is used to create a custom Simulink block with multiple input and output ports [34]. Lines `89-95` adds the equation of motion to the code by applying the `getDCGfunc` function. The inputs to the Level-2 block are the state vector defined as (7.1 - 7.3) and the applied torque vector τ, and the output is the vector of state derivatives. For each time step, during the simulation, the updated vector of state derivatives is computed from the new inputs.

7.3 Simulations

The simulation setup revolve around keeping the robot arm stationary in its zero-configuration (see figure 6.2. The zero-configuration is given by $\boldsymbol{q}_d$ and the motionless state of the robot arm is when the joint velocities $\dot{\boldsymbol{q}}_d$ are all equal to zero.

$$\boldsymbol{q}_d = \begin{bmatrix} 0 & \frac{\pi}{2} & 0 & 0 & \frac{\pi}{2} & 0 \end{bmatrix}^T \tag{7.11}$$

$$\dot{\boldsymbol{q}}_d = \begin{bmatrix} 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix}^T \quad \Leftrightarrow \quad \ddot{\boldsymbol{q}}_d = \begin{bmatrix} 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix}^T \tag{7.12}$$

The torque values are constant and set to the values required to compensate for the weight of gravity. Hence, the required torque values are found by inserting (7.11) and (7.12) into the equation of motion.

$$\boldsymbol{\tau}_d = \boldsymbol{G}(\boldsymbol{q}_d, \boldsymbol{g}_0), \quad \boldsymbol{g}_0 = [0, 0, -9.81]^T \tag{7.13}$$

$$\Downarrow$$

$$\boldsymbol{\tau}_d = \begin{bmatrix} 0 \\ -6132.9 \\ -6026.4 \\ -45.8 \\ 1271.9 \\ 0 \end{bmatrix} \tag{7.14}$$

Note, the unit of the DH-parameters set in `mm` resulting in relatively big torque values.

With the parameters presented in (7.11 - 7.14), three simulations were initialized.

Simulation 1

The initial state of the Bravo 7 robot arm is set to its zero-configuration.

Initial conditions:

$$\boldsymbol{q}_i = \begin{bmatrix} 0 & \frac{\pi}{2} & 0 & 0 & \frac{\pi}{2} & 0 \end{bmatrix}^T \tag{7.15}$$

$$\dot{\boldsymbol{q}}_i = \begin{bmatrix} 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix}^T \tag{7.16}$$

Simulation 2

The initial state of the Bravo 7 robot arm is close to its zero-configuration, but shifted by 0.5 for joint number 2 and 5.

Initial conditions:

$$\boldsymbol{q}_i = \begin{bmatrix} 0 & \frac{\pi}{2} + 0.5 & 0 & 0 & \frac{\pi}{2} + 0.5 & 0 \end{bmatrix}^T \tag{7.17}$$

$$\dot{\boldsymbol{q}}_i = \begin{bmatrix} 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix}^T \tag{7.18}$$

Simulation 3

The initial state of the Bravo 7 robot arm is set further from the zero-configuration (laying down sideways).

$$\boldsymbol{q}_i = \begin{bmatrix} 0 & \frac{\pi}{2} & 0 & \frac{\pi}{2} & \frac{\pi}{2} & \frac{\pi}{2} \end{bmatrix}^T \tag{7.19}$$

$$\dot{\boldsymbol{q}}_i = \begin{bmatrix} 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix}^T \tag{7.20}$$

Part IV

Results and Conclusive Remarks

Chapter 8

Results

The chapter presents the simulation plot results, a conclusive summary and possible future work based on the findings in this project thesis.

Specific results for the forward kinematics are not presented in this chapter. However, the forward kinematics code were confirmed to be correct using the *Peter Corke Robotics Toolbox* for Matlab [35] by comparing output values. Nevertheless, the simulations of the dynamics model were based on the forward kinematics, and therefore is an indirect result of the forward kinematics. The plots regarding the different simulations, as described in section 7.3, are presented in section 8.2. Section 8.1 presents the challenges discovered when attempting to find a solution to the inverse kinematics.

8.1 Challenges of the Inverse Kinematics

Attempting to find a solution to the inverse kinematics problem for the Bravo 7 proved to be challenging. The methods tested in section 6.3 provided satisfying results only when using a numerical approach. The results in regards to the different methods are summarized below.

Geometrical solution:

Finding a geometrical solution to a robot arm, even in its reduced form, is hard because the approach is robot specific with considerable chance for errors along the way. The attempted solution for the Bravo 7 was not proven to be correct in comparison with the forward kinematics solutions.

IKBT method:

The IKBT method showed some promise when a simplified version of the DH-table was used as input. The output provided an algebraic solution, but some parts of the solution were lacking. The program was never designed for 6 DOF robots without a spherical wrist, thus it was never expected to provide a solution even for a simplified version of the Bravo 7.

Optimization method:

The optimization method gave the correct joint value outputs with the *Levenberg Marquardt* algorithm, meanwhile the default (*Interior Point*) algorithm crashed in most cases because of the lack of progress at each time step. It is worth mentioning that the *Levenberg Marquardt* algorithm was not tested extensively as to find configurations where the algorithm might crash.

Discussion

One of the main findings regarding the inverse kinematics of the Bravo 7 was the realization concerning the sheer complexity of the problem. The analytical solution is preferred and even though none of the attempted solutions were successful, there might be worth investing more time and focus on finding a simplified analytical solution. The results of the numerical optimization method were positive, but this was also just an introduction to a possible inverse kinematics solution. At this point it is hard to say what demands are present in a solution for practical applications.

8.2 Dynamics Model Simulation

The three simulations with different initial conditions are presented below in the form of graphs and comments about the results. Because of the simplified setup, the time consumption for each simulation was more than acceptable.

Simulation 1

In simulation 1 (see figure **??**), the initial joint positions are the same as the desired positions. The graph response is as expected. All the joint positions are more or less kept at constant values because the actuator torques are set to compensate for gravity.

Simulation 2

Simulation 2 (see figure **??**) displays an increasingly unstable system when the initial positions are not in immediate proximity of the desired positions. This is because the gravity affecting the links are dependant on the joint variables, hence causing an uncontrollable motion by the slightest drift away from the position deciding the constant torque.

Simulation 3

Simulation 3 (see figure **??**) involves initial joint positions further away from the desired positions than in Simulation 2 and as a result, the system becomes even more unstable.

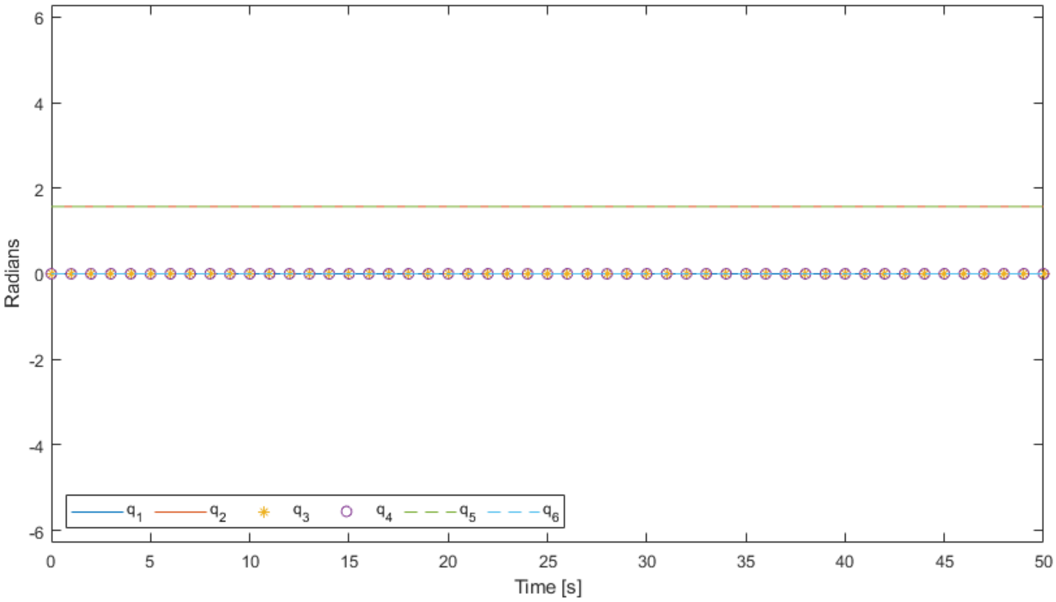


Figure 8.1: Plot of simulation 1.

Discussion

All the simulation plots are as anticipated for an open loop system with no control feedback. The predicted response is positive in terms of the theoretical dynamic model because it implies model validity. Still, the simulation setup is simplistic and the joint velocities are set to zero which means the inertia matrix, and centrifugal and Coriolis terms of the model are in fact not taken into consideration. More complex setups will most likely be more time consuming when using the Euler-Lagrange approach, and the Newton-Euler approach could be required.

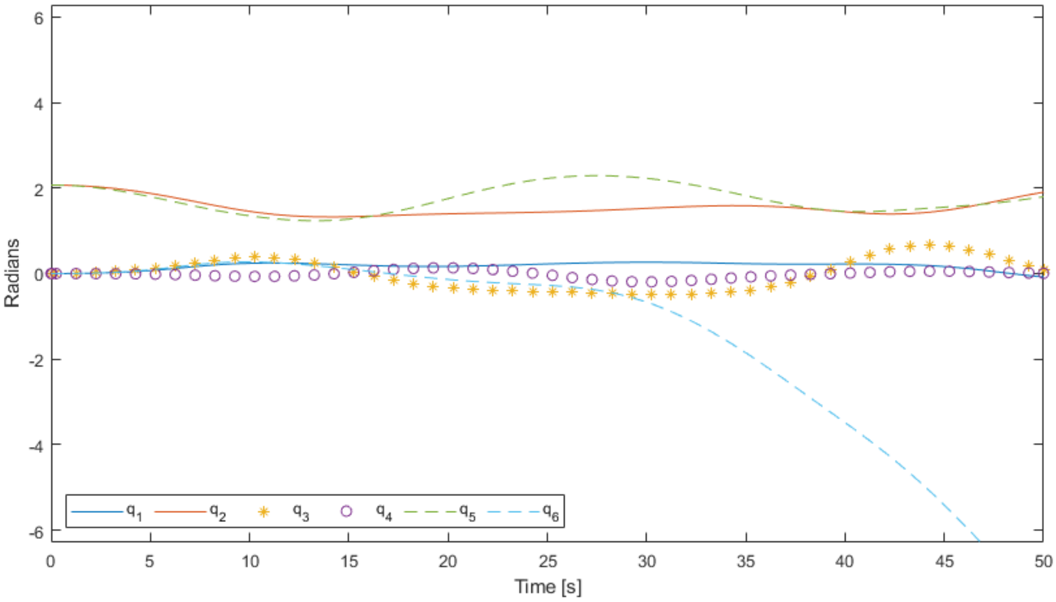


Figure 8.2: Plot of simulation 2.

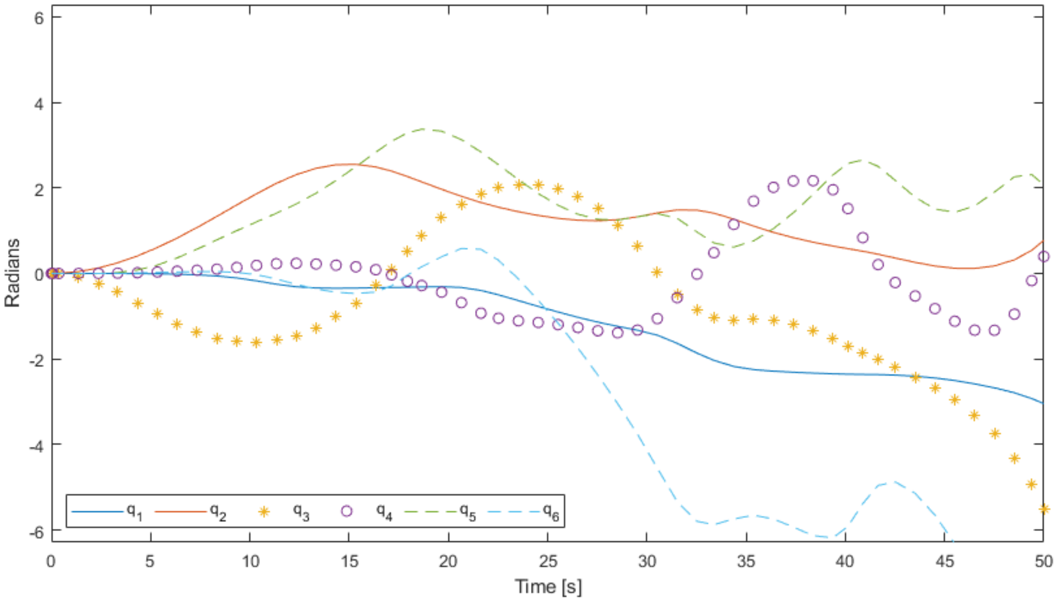


Figure 8.3: Plot of simulation 3.

Chapter 9

Conclusive Remarks

This chapter includes a brief conclusion as well as suggestions for future work based on contents of the project thesis.

9.1 Conclusion

In this project thesis a theoretical foundation for calculating the kinematic and dynamic models of robot manipulators have been presented. The forward kinematics solution based on the Denavit-Hartenberg convention were introduced as the first building block of the mathematical description of the robot arm. Analytical and numerical approaches were presented as solutions to the inverse kinematics problem. The last part of the theoretical foundation was the dynamic model which can be solved using the Euler-Lagrange or the Newton-Euler formulation. The Euler-Lagrange approach was given separate attention and was used to find the dynamic model of the Reach Bravo 7 robot arm. During the model calculations for Bravo 7, challenges were discovered. The inverse kinematics were proven to be especially challenging and time-consuming. It requires more time and a more direct focus to be solved. The simulations of the dynamic model were made to see if the model made sense. A simplified simulation setup with an open loop configuration gave expected results, but the simulation complexity were not high enough to make any definite conclusion about the model validation. As a conclusion, the kinematic and dynamic models together with the discovered challenges in this project thesis can be used as inspiration for future work and more advanced models involving the Reach Bravo 7.

9.2 Future Work

One of the goals of this project thesis was to create a general foundation for future work, some of the most imminent possibilities are presented below.

- A more advanced dynamic model with added dynamic effects like hydrodynamics, joint friction and/or joint flexibility.

- Adding physical constraints to the robot models.
- Validation of the dynamic model based on energy properties and energy conservation.
- Create control laws for closed loop position control.
- Investigation more advanced solutions for the inverse kinematics.
- Incorporating the models to be used in FhSim and present case studies in a Smolt tank.

Bibliography

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

[2] Seth Hutchinson Mark W. Spong and M. Vidyasagar. *Robot Modeling and Control.* John Wiley Sons, 1st edition, 2005.

[3] Food and Agriculture Organization of the United Nations. *Conserve and sustainably use the oceans, seas and marine resources.* URL `http://www.fao.org/sustainable-development-goals/goals/goal-14/en/`.

[4] Siwa Msangi et al. *FISH TO 2030 - Prospects for Fisheries and Aquaculture.* The World Bank, 2013.

[5] Norwegian Seafood Federation. *Aquaculture 2030 - Think globally, act locally.* Norwegian Seafood Federation, 2017.

[6] A. Delaporte M. J. Phillips M. Beveridge Hall, S.J. and M. O'Keefe. *Blue Frontiers: Managing the Environmental Costs of Aquaculture.* The WorldFish Center, Penang, Malaysia, 2011.

[7] *New Growth, Proud History: The Norwegian Government's Ocean Strategy.* Norwegian Ministry of Trade, industry and Fisheries and Norwegian Ministry of Petroleum and Energy, 2017.

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

[9] *Salmon Farming: Industry Handbook 2020.* Mowi, 2020.

[10] Salmar. *ABC of Salmon Farming.* URL `https://www.salmar.no/en/abc-of-salmon-farming/`.

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

[12] Torrissen et al. *Impact on wild salmonids and salmon aquaculture.* Journal of Fish Diseases, 2013.

[13] Trond Bjørndal and Amalie Tusvik. *Economic analysis of on-growing of salmon post-smolts.* Taylor Francis Online, 2020.

[14] Heiner Lasi et. al. *Industry 4.0.* Springer, 2014.

[15] Pål Hofset Skeide. *Automating tank operations in smolt production – A concept study for automating tank cleaning using robotic arms.* Master Thesis, 2020.

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

[17] Peter Corke. *Robotics, Vision and Control.* Springer, 2nd edition, 2017.

[18] Luo Qingsheng Wang Shanda, Luo Xiao and Han Baoling. *Existence Conditions and General Solutions of Closed-form Inverse Kinematics for Revolute Serial Robots.* MDPI, 2019.

[19] Michael Meredith and Steve Maddock. *Real-Time Inverse Kinematics: The Return of the Jacobian.* University of Sheffield, 2004.

[20] Yang Zhang Huashan Liu and Shiqiang Zhu. *Novel Inverse Kinematic Approaches for Robot Manipulators with Pieper-criterion based Geometry.* International Journal of Control, Automation, and Systems, 2015.

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

[22] Dianmu Zhang and Blake Hannaford. *IKBT: Solving Symbolic Inverse Kinematics with Behavior Tree.* Journal of Artificial Intelligence Research, 2019.

[23] Luigi Villani Bruno Siciliano, Lorenzo Sciavicco and Giuseppe Oriolo. *Robotics: Modelling, Planning and Control.* Springer, 3rd edition, 2010.

[24] Olav Egeland and Jan Tommy Gravdahl. *Modelling and Simulation for Automatic Control.* Norwegian University of Science and Technology, 2002.

[25] Blueprint Lab. *About Blueprint Lab*, . URL `https://blueprintlab.com/about-us/`. Visited : 13.12.2020.

[26] Blueprint Lab. *Reach Bravo*, . URL `https://blueprintlab.com/products/reach-bravo/`. Visited : 13.12.2020.

[27] Blueprint Lab. *Bravo 7 Datasheet*, . URL `https://blueprintlab.com/wp-content/uploads/2020/08/Bravo_7_Datasheet.pdf`. Visited 04.01.2021.

[28] Blueprint Lab. *Reach Bravo Kinematics and Dynamics*, . URL `https://github.com/blueprint-lab/Blueprint_Lab_Software/blob/master/Documentation`. Visited : 04.01.2021.

[29] Inc. The MathWorks. *Symbolic Math Toolbox.* Natick, Massachusetts, United State, 2020. URL `https://www.mathworks.com/help/symbolic/`.

[30] Blake Hannaford. *IKBT.* URL `https://github.com/uw-biorobotics/IKBT`. Visited : 20.11.2020.

[31] Inc. The MathWorks. *Optimization Toolbox.* Natick, Massachusetts, United State, 2020. URL `https://www.mathworks.com/help/optim/`.

[32] Inc. The MathWorks. *Optimization Toolbox: Least-Squares (Model Fitting) Algorithms.* Natick, Massachusetts, United State, 2020. URL `https://www.mathworks.com/help/optim/ug/least-squares-model-fitting-algorithms.html`.

[33] Herman Høifødt. *Dynamic Modeling and Simulation of Robot Manipulators: The Newton-Euler Formulation.* Norges teknisk-naturvitenskapelige universitet, NTNU, 2011.

[34] MATLAB. *About Level-2 MATLAB S-Functions.* URL `https://www.mathworks.com/help/simulink/sfg/writing-level-2-matlab-s-functions.html`. Visited : 11.01.2020.

[35] Peter I. Corke. *Robotics, Vision & Control: Fundamental Algorithms in MATLAB.* Springer, second edition, 2017.

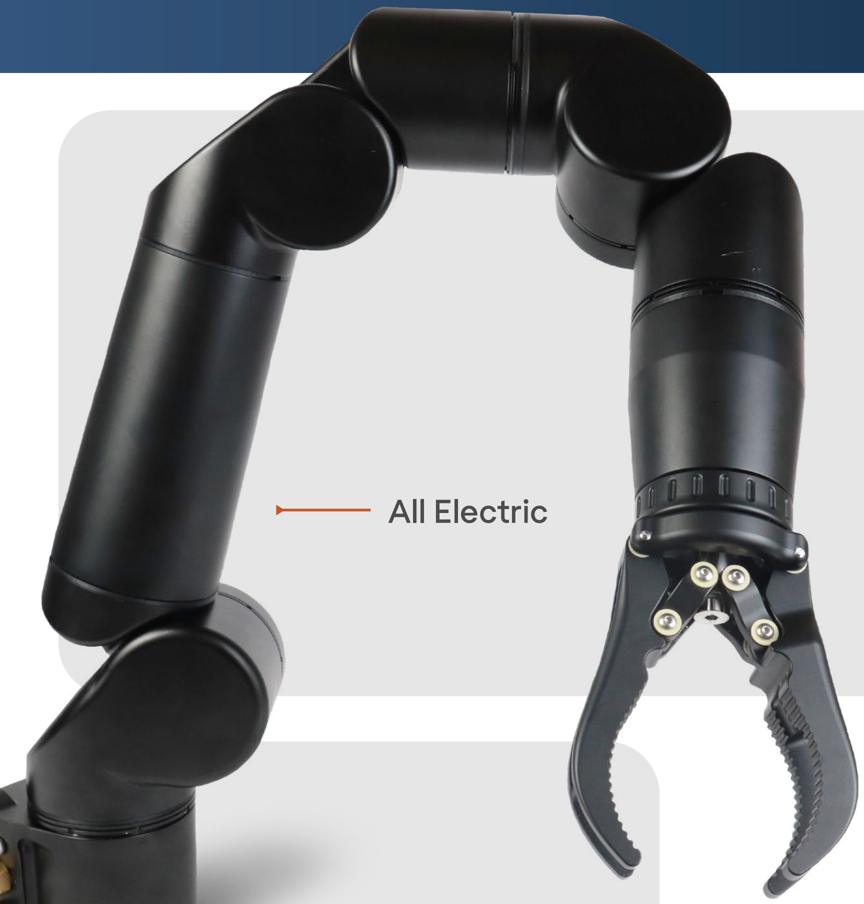
Appendix A

Bravo Reach 7 Datasheet

BLUEPRINTLAB

BRAVO 7

A Tough, 7-Function Manipulator
for Inspection Class Vehicles



Be Confident.

The Bravo 7 is a 7-Function manipulator that opens up new compact inspection and intervention opportunities for service providers, researchers, and other operators.

Designed to conduct tasks usually reserved for human divers, the arm's dexterity and responsiveness pave the way for advanced applications.

The form factor was specifically designed for industry leading inspection-class ROVs making it a ready-to-go option for existing fleets.

Master Controller Enabled

Note: Master Arm not included

FEATURES

- Ready-built and designed for industry leading inspection-class vehicles
- Master Arm Controlled
- All-Electric, Zero Oil
- End effectors: grabbers, probe handlers, cutters (or BYO)
- Advanced software interface with 3D visualisation
- Adjustable grab force - pick up a sea urchin or cut a cable
- One-click deploy/stow position

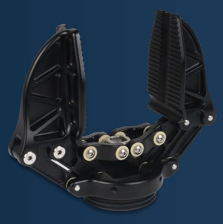
DEXTEROUS
7-Function (or custom fit)
Highly Modular & Configurable

RUGGED
300m Depth Rating
10kg Full Reach Lift

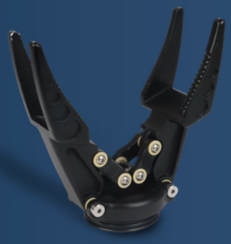
SMART
In-Built Kinematics
Master Arm Controlled

Mission Specific End-Effectors | *Interchange With Ease*

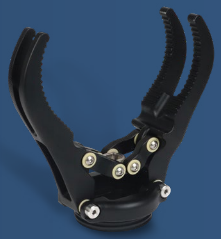
Parallel Jaws



Quad Jaws



Interlocking Quad Jaws



SPECIFICATIONS

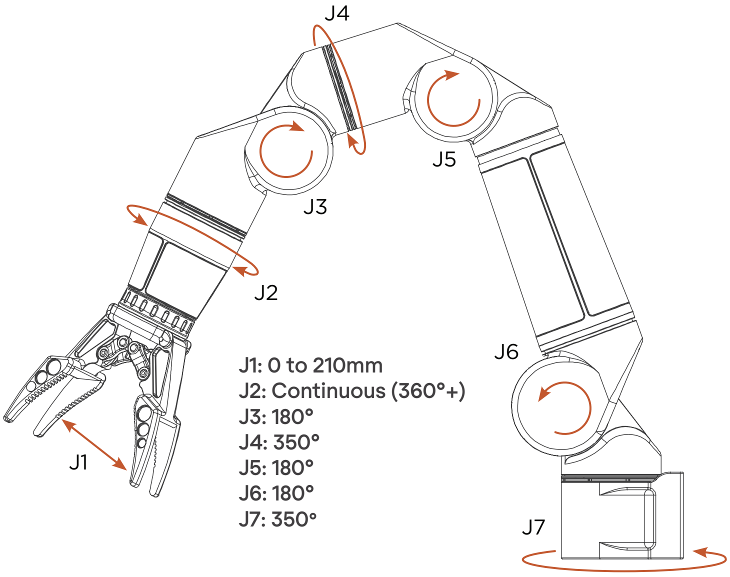
Reach	0.9m
Full Reach Lift	10kg
Max Lift Capacity	20kg
Joint Speed	60deg/s nominal
End-effector Accuracy	<1cm
Grabber Close Force	1000N
Optional Sensor Interface	PWR + 232 + 485 + Ethernet
Control Modes	Position, Velocity, Cartesian (XYZ)

Power Interface

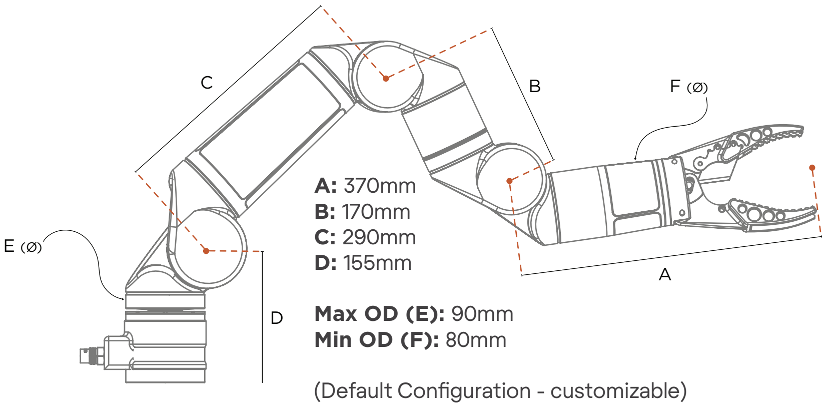
Bulkhead	MCBH4M – MC 4C Male
Pigtail	MCIL4F – MC Inline, 4C Female, 60cm

Comms Interface

Bulkhead	MCBH8ME – MC Ethernet, 8C Male
Pigtail	MC Ethernet Inline, 8C Female 100cm



J1: 0 to 210mm
J2: Continuous (360°+)
J3: 180°
J4: 350°
J5: 180°
J6: 180°
J7: 350°



A: 370mm
B: 170mm
C: 290mm
D: 155mm

Max OD (E): 90mm
Min OD (F): 80mm

(Default Configuration - customizable)

- ## ELECTRICAL

Voltage	20-48V
Nominal Power	200W (10kg Load)
Peak Power	300W (10kg Load)
COMS	RS485/Ethernet

- ## ENVIRONMENTAL

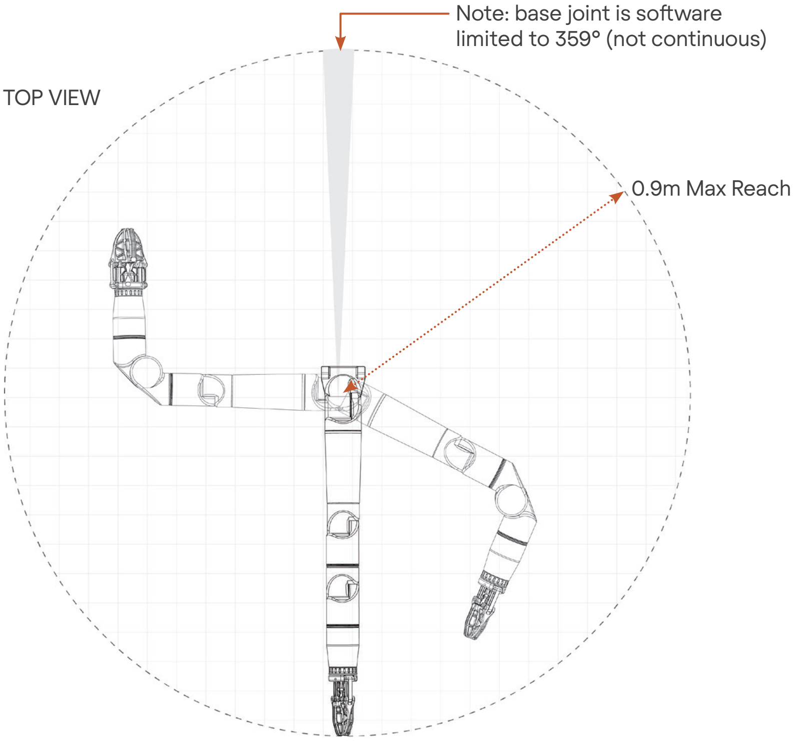
Depth	300MSW
Temp	-10°C to 35°C
Material	AL6061

- ## MECHANICAL

Lift Capacity	10kg (full reach)
Max Lift	20kg
Weight (Air)	9.5kg
Weight (Water)	4.5kg

High Mobility & Compact

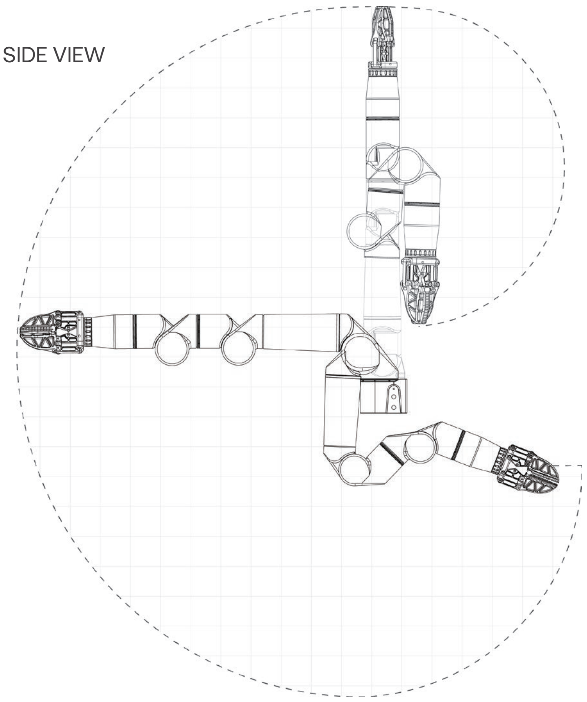
TOP VIEW



Note: base joint is software limited to 359° (not continuous)

0.9m Max Reach

SIDE VIEW



Appendix B

Bravo Reach 7 Documentation

Reach Bravo Kinematics and Dynamics

Blueprint Lab

August 2020

1 Kinematics

Link	**d (mm)**	$\boldsymbol{\theta}$	**a (mm)**	$\boldsymbol{\alpha}$
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 1: Standard DH Parameters for Bravo 7 where $\theta_a = \tan^{-1}\left(\frac{5.2}{293.55}\right)$

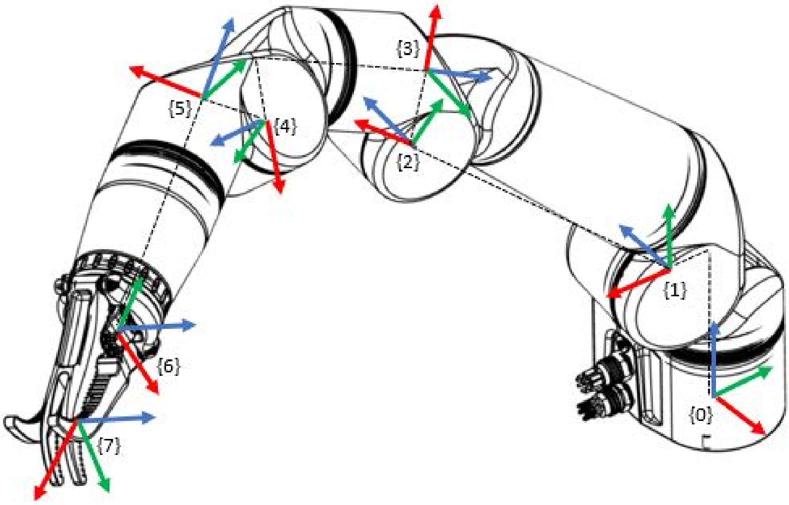


Figure 1: Bravo 7 joint frames (x,y,z)

2 Inertial Properties

Link	**Mass** ($\boldsymbol{kg}$)	**COM** ($\boldsymbol{mm}$)	**I** ($\boldsymbol{kg.mm^2}$)
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 2: Inertial properties for Bravo 7 (Link 7 is for interlocking jaws in the closed position)

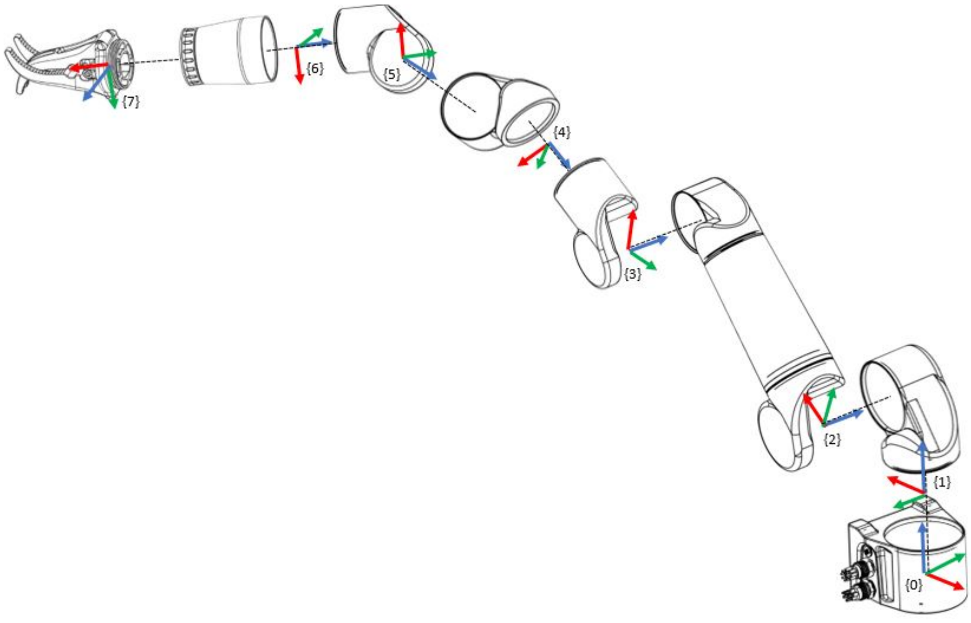


Figure 2: Inertial frames for Bravo 7

3 Hydrodynamic Properties

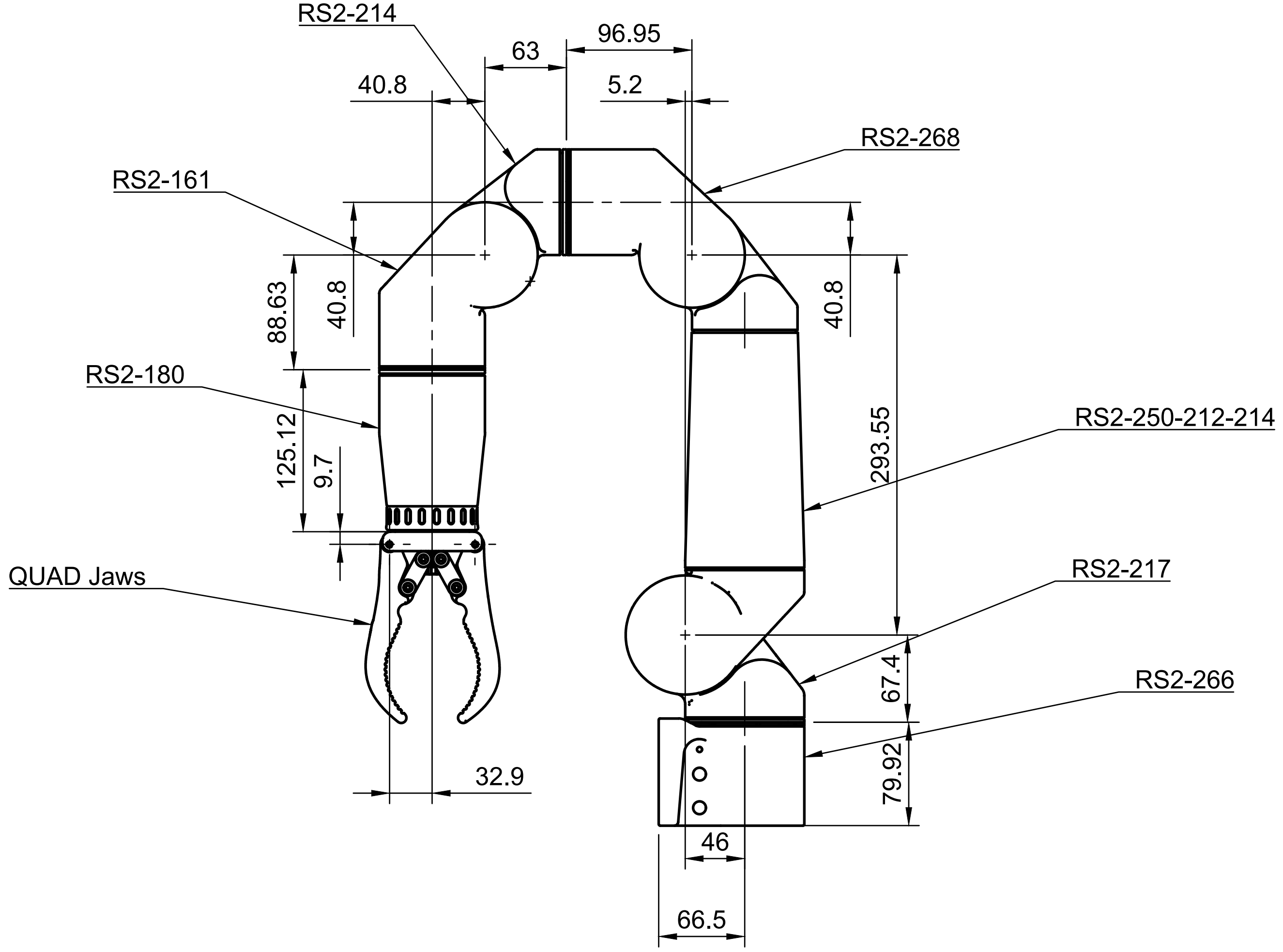
3.1 Buoyancy

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 3: Buoyancy terms with Centre of Buoyancy (COB)

Appendix C

Bravo Reach 7 Machine Drawing



Dept.	Technical reference	Created by John Sumskas 10/06/2020	Approved by
		Document type	Document status
		Title Bravo 7 Rigid	DWG No.
			Rev.
			Date of issue
			Sheet 1/1

Appendix D

MATLAB: Dynamic Parameter Calculation

```

%% Forward Kinematics
    % Number of joints:
    n = 6;

    % To simplify the notation of sin and cos expressions:
    syms q1 q2 q3 q4 q5 q6
    syms dq1 dq2 dq3 dq4 dq5 dq6  % dq == dq/dt
    syms gz % gravitational acceleration in the inertial frame along z-axis


    g_vec = [0;0;gz]; % Gravitational acceleration vector given inertia frame



    q = [q1; q2; q3; q4; q5; q6];
    dq = [dq1; dq2; dq3; dq4; dq5; dq6];



    c1 = cos(q1); c2 = cos(q2); c3 = cos(q3); c4 = cos(q4); c5 = cos(q5);
    c6 = cos(q6);
    s1 = sin(q1); s2 = sin(q2); s3 = sin(q3); s4 = sin(q4); s5 = sin(q5);
    s6 = sin(q6);

    s = [s1 s2 s3 s4 s5 s6];
    c = [c1 c2 c3 c4 c5 c6];

    % From DH-table [mm]:
    d = [107.4, 0, 0, -160, 0, -223.5];
    a = [46, 293.6 , 40.8, 40.8, 40.8, 0];
```

```

% Homogeneous transformation matrices:


A1 = [(c(1)),  0,     (s(1)),  (a(1)*c(1));
      (s(1)),  0,    (-c(1)),  (a(1)*s(1));
           0,  1,          0,       (d(1));
           0,  0,          0,            1];
A2 = [c(2) -s(2)  0     a(2)*c(2);
      s(2)  c(2)  0     a(2)*s(2);
      0        0  1          d(2);
      0        0  0             1];
A3 = [c(3)  0    -s(3)  a(3)*c(3);
      s(3)  0     c(3)  a(3)*s(3);
      0    -1       0        d(3);
      0     0       0           1];
A4 = [c(4)  0    -s(4)  a(4)*c(4);
      s(4)  0     c(4)  a(4)*s(4);
      0    -1       0        d(4);
      0     0       0           1];
A5 = [c(5)  0    -s(5)  a(5)*c(5);
      s(5)  0     c(5)  a(5)*s(5);
      0    -1       0        d(5);
      0     0       0           1];
A6 = [c(6)  0     s(6)  a(6)*c(6);
      s(6)  0    -c(6)  a(6)*s(6);
      0     1       0        d(6);
      0     0       0          1];

T = cell(1,n);

T{1} = simplify(A1);
T{2} = simplify(T{1}*A2);
T{3} = simplify(T{2}*A3);
T{4} = simplify(T{3}*A4);
T{5} = simplify(T{4}*A5);
T{6} = simplify(T{5}*A6);

% Coordinates of CoM:
rci = cell(1,n); % coordinates of CoM in frame i

rci{1} = [-18; -4; -1];
rci{2} = [17; -7; -57];
rci{3} = [117; 15; 6];
rci{4} = [22; -29; 1];
rci{5} = [18; 6; -117];
rci{6} = [20; -24; 1];

```

```
    rc = cell(1,n); % coordinates of CoM in the inertial frame

    for i = 1:n
        rc{i} = simplify([eye(3),zeros(3,1)]*T{i}*[rci{i};1]);
    end

    %Velocity Jacobians
    Jv = cell(1,n);
    Jw = cell(1,n);
    for i = 1:n
        z_pre = [0;0;1];
        o_pre = [0;0;0];
        Jvt = sym(zeros(3,n));
        Jwt = sym(zeros(3,n));
        for j = 1:i
            Jvt(:,j) = cross(z_pre,rc{i}-o_pre);
            Jwt(:,j) = z_pre;

            z_pre = T{j}(1:3,3);
            o_pre = T{j}(1:3,4);
        end
        Jv{i} = Jvt;
        Jw{i} = Jwt;
    end

    %Inertia in body fixed frame:
    I = cell(1,n);

    I{1} = [2108 182 -15; 182 2573 -21; -15 -21 3483];
    I{2} = [11442 -484 3405; -484 12980 -1265; 3405 -1265 3202];
    I{3} = [3960 4200 3204; 4200 69099 -24; 3204 -24 70450];
    I{4} = [3213 -1548 -31; -1548 2327 6; -31 6 4340];
    I{5} = [21232 330 -3738; 330 22252 -1278; -3738 -1278 2054];
    I{6} = [2430 -1144 -40; -1144 2026 11; -40 11 3330];

    %Inertia tensors:

    It = cell(1,n);
    for i = 1:n
        R = T{i}(1:3,1:3);
        It{i} = simplify(R*I{i}*R');
    end




%% Kinetic and Potential Energy

```

```
% Mass of each link: [kg] (From Blueprint Documentation)
m1 = 1.25;
m2 = 1.55;
m3 = 1.98;
m4 = 1.14;
m5 = 1.14;
m6 = 1.03;

m = [m1 m2 m3 m4 m5 m6];

%Calculations:

K = sym(0);
P = sym(0);

for i = 1:n
    K = K + 0.5*dq'*(m(i)*Jv{i}'*Jv{i} + Jw{i}'*It{i}*Jw{i})*dq;
    P = P + m(i)*g_vec'*rc{i};
end


%% Generate matrices for equations of motion

%Inertia Matrix:
D = sym(zeros(n));

for i = 1:length(q)
    for j = 1:length(q)
        D(i,j) = diff(diff(K,dq(j)), dq(i));
    end
end

%Coriolis and Centrifugal matrix:
C = sym(zeros(n));

for i = 1:length(q)
    for j = 1:length(q)
        for k = 1:length(q)
            christ = 0.5*(diff(D(i,j),q(k))+diff(D(i,k),q(j))...
                    -diff(D(k,j),q(i)));
            C(i,j) = C(i,j) + christ*dq(k);
        end
    end
end

Cdq = C*dq;

%Gravity vector
```

```
    G = sym(zeros(n,1));

    for i = 1:length(q)
        G(i) = diff(P,q(i));
    end

    matlabFunction(D,Cdq,G,'File','getDCGfunc','Vars',{[q,dq],gz});
```

Appendix E

MATLAB: Functions for Simulink

```
function bravo7(block)


% This function is used to setup the basic attributes of the
% S-function such as ports, parameters, etc. .

setup(block);


function setup(block)

  % Register number of ports
  block.NumInputPorts  = 2;
  block.NumOutputPorts = 1;

  % Setup port properties to be inherited or dynamic
  block.SetPreCompInpPortInfoToDynamic;
  block.SetPreCompOutPortInfoToDynamic;

  % Override input port properties
  block.InputPort(1).Dimensions  = [12 1];
  block.InputPort(2).Dimensions  = [6 1];
  block.InputPort(1).DirectFeedthrough = false;
  block.InputPort(2).DirectFeedthrough = false;

  % Override output port properties
  block.OutputPort(1).Dimensions  = [12 1];

  % Register parameters
  block.NumDialogPrms = 0;

  % Register sample times
  block.SampleTimes = [-1 0];


  %% ----------------------------------------------------------------
  %% Options
  %% ----------------------------------------------------------------
  % Specify if Accelerator should use TLC or call back into
  % M-file
  block.SetAccelRunOnTLC(false);


  %% ----------------------------------------------------------------
  %% Register methods
  %% ----------------------------------------------------------------
  block.RegBlockMethod('InitializeConditions', @InitializeConditions);
  block.RegBlockMethod('Outputs', @Outputs);

  function InitializeConditions(block)
block.InputPort(1).Data(1) = 0;
block.InputPort(1).Data(2) = 0;
block.InputPort(1).Data(3) = 0;
block.InputPort(1).Data(4) = 0;
block.InputPort(1).Data(5) = 0;
block.InputPort(1).Data(6) = 0;
block.InputPort(1).Data(7) = 0;
block.InputPort(1).Data(8) = 0;
block.InputPort(1).Data(9) = 0;
block.InputPort(1).Data(10) = 0;
block.InputPort(1).Data(11) = 0;
block.InputPort(1).Data(12) = 0;


function Outputs(block)


%Defining input ports
tau1 = block.InputPort(2).Data(1);
tau2 = block.InputPort(2).Data(2);
tau3 = block.InputPort(2).Data(3);
tau4 = block.InputPort(2).Data(4);
tau5 = block.InputPort(2).Data(5);
tau6 = block.InputPort(2).Data(6);

q1 = block.InputPort(1).Data(1);
dq1 = block.InputPort(1).Data(2);
q2 = block.InputPort(1).Data(3);
dq2 = block.InputPort(1).Data(4);
q3 = block.InputPort(1).Data(5);
dq3 = block.InputPort(1).Data(6);
```

```
q4 = block.InputPort(1).Data(7);
dq4 = block.InputPort(1).Data(8);
q5 = block.InputPort(1).Data(9);
dq5 = block.InputPort(1).Data(10);
q6 = block.InputPort(1).Data(11);
dq6 = block.InputPort(1).Data(12);

q = [q1;q2;q3;q4;q5;q6];
dq = [dq1;dq2;dq3;dq4;dq5;dq6];
gz = -9.81;

[M,Cdq,G] = getDCGfunc([q,dq],gz);

DynSys = M\(-Cdq - G + [tau1;tau2;tau3;tau4;tau5;tau6]);
    %Mapping to first-order system
    dx1 = dq1;
        dx2 = DynSys(1);
    dx3 = dq2;
    dx4 = DynSys(2);
    dx5 = dq3;
    dx6 = DynSys(3);
    dx7 = dq4;
    dx8 = DynSys(4);
    dx9 = dq5;
    dx10 = DynSys(5);
    dx11 = dq6;
    dx12 = DynSys(6);

    %Sending to output port
    dx = [dx1;dx2;dx3;dx4;dx5;dx6;dx7;dx8;dx9;dx10;dx11;dx12];
    dx = double(dx);
    block.OutputPort(1).Data = dx;



```

Appendix F

Contents of Delivered ZIP-File

PDF Files

- *Project Thesis:* The specialization project delivery.
- *project_task_description:* A text explaining the specialization project background information and tasks to be performed.

Matlab Files

- *getDCGfunc.m:* Matlab function to compute matrices for the dynamics model.
- *bravo_fk.m:* Matlab function to compute the forward kinematics.
- *bravo_ik.m:* Matlab script to compute optimization problem of the inverse kinematics.
- *bravo7.m:* Matlab functions setup for for simulink.
- *calculate_x_dot.m:* Matlab script creating symbolic equations for the dynamic model and using the symbolic equations to form a matlabFunction.

Simulink Files

- *SixDofOpenLoop_bravo7.m:* Simulink diagram open loop system for simulation of the dynamic model.

Appendix G

MATLAB: Optimization Method for Inverse Kinematics

```
q = [0 0 0 0 0 0 0]; % Initial joint values
option = optimset('Algorithm','levenberg-marquardt','Display','off');

q = fsolve(@bravo_ik2,q,option); % Solution to the Optimization problem


function F = bravo_ik2(q)

    %% Forward Kinematics
    A06 = bravo_fk(q);

    %% Output to solve Inverse Kinematics numerically

    D = [ 1 0 0 0.2468; 0 0 1 0; 0 -1 0 0.7061; 0 0 0 1];
    % D is the destination we want to reach

    F = A06(1:3,:) - D(1:3,:);

end
```

```
function A = bravo_fk(q)
%% Parameters
    % From DH-table
    d = [0.1074, 0, 0, -0.160, 0, -0.2235];
    a = [0.046, 0.2936 , 0.0408, 0.0408, 0.0408, 0];

    % To simplify the notation
    s = zeros(1,6);
    c = zeros(1,6);

    for i = 1:6
```

```
        s(i) = sin(q(i));
        c(i) = cos(q(i));
    end
    %% Homogeneous transformation matrices for each link

    A1 = [c(1)  0     s(1)  a(1)*c(1); s(1)  0    -c(1)  a(1)*s(1);
        0  1 0 d(1); 0 0 0 1];
    A2 = [c(2) -s(2)  0     a(2)*c(2); s(2)  c(2)  0     a(2)*s(2);
        0  0 1 d(2); 0 0 0 1];
    A3 = [c(3)  0    -s(3)  a(3)*c(3); s(3)  0     c(3)  a(3)*s(3);
        0 -1 0 d(3); 0 0 0 1];
    A4 = [c(4)  0    -s(4)  a(4)*c(4); s(4)  0     c(4)  a(4)*s(4);
        0 -1 0 d(4); 0 0 0 1];
    A5 = [c(5)  0    -s(5)  a(5)*c(5); s(5)  0     c(5)  a(5)*s(5);
        0 -1 0 d(5); 0 0 0 1];
    A6 = [c(6)  0     s(6)  a(6)*c(6); s(6)  0    -c(6)  a(6)*s(6);
        0  1 0 d(6); 0 0 0 1];

    %% Forward Kinematics
    A = A1*A2*A3*A4*A5*A6;
end
```