Master's thesis

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

Henrik Baldishol

Practical application of potential fields path planning on robot manipulator

Master's thesis in Teknisk kybernetikk
Supervisor: Martin Føre
Co-supervisor: Bent Haugaløkken
July 2022

Henrik Baldishol

Practical application of potential fields path planning on robot manipulator

Master’s thesis in Teknisk kybernetikk
Supervisor: Martin Føre
Co-supervisor: Bent Haugaløkken
July 2022

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

Sammendrag

Lakseoppdrett er en av Norges største industrier, med landbasert produksjon basert på innovasjon fra 1980-tallet. Mange teknologiske fremskritt er gjort siden konstruksjonen av de tidligste produksjonsfasilitetene, er industrien fortsatt avhengig av manuel arbeidskraft for de fleste elementene i produksjonsprosessen. Denne oppgaven (i med prosjektet Autosmolt2025[18]) er dedikert til å finne autonome løsninger til en av de mer essensielle undergruppene av disse manuelle arbeidsoppgavene, fjerningen av død fisk og avfall i smolttanker.

Løsningen har blitt jobbet med i oppgaven bruker revolutt-ledd robotmanipulatorer som er montert på en Remotely Operated underwater Vechile (ROV). Kun selve robotmanipulatoren er brukt i oppgaven.

Den teoretiske delen av oppgaven omfatter teori rundt modellering, baneplanlegging og reguleringen som kreves for å styre robotmanipulatorens end-effector mot ønsket posisjon, samtidig som at rotasjonsledd og lenker unngår å kollidere med hindringer i arbeidsområdet.

Løsninger for baneplanlegging ble implementert med bruken av potensialfelt. Dette ble gjennomført ved å modellere tiltrekkende felter som drar robotens rotasjonsledd mot ønsket sluttkonfigurasjon, mens frastøtende felter ble implementert for å styre leddene bort fra hindringer. Minimalisering av potensialer i disse feltene ble brukt som optimaliseringsobjektiv for å gradvis bevege manipulatoren mot ønsket endeposisjon.

Den implementerte metoden med potensiale felter hadde en tendens til å bli fanget i lokale minimum. En "random walk" metode ble implementert for å unngå dette problemet.

Flere tester ble gjennomført for å verifisere at baneplanleggeren klarer å finne kollisjonsfrie baner mellom forskjellige konfigurasjoner. Testene viste at baneplanleggeren fortsatt har problemer med lokale minimum, og at baneplanleggeren ikke klarer å genere komplette baner mellom konfigurasjoner generelt. Optimaliseringsalgoritmen ble forsøkt forbedret ved å implementere Wolfe-kriterier for å begrense steglengden på algoritmen. En feil ble funnet i implementasjonen av Wolfe-kriteriene, som derfor ikke kan brukes i den endelige optimaliseringsalgoritmen.

Banene som ble generert ble deretter brukt til å flytte robotmanipulatoren mellom forskjellige konfigurasjoner. Manipulatoren klarte ikke å følge hele den planlagte banen.

Oppgaveen konkluderer med at potensiale felter var utilstrekkelig for et problem av slik kompleksitet. De fleste konfigurasjoner førte til ukomplette baner, da algoritmen ble låst i et lokalt minima. Noen forslag ble gitt til hvordan man kan forbedre delene av algoritmen som ikke virket som konsekvenns av implementasjonen.

Abstract

Salmon production remains as one of Norways largest industries, with land-based production sites ranging back to the 1980's. While many technological advancements have been made since the construction of the initial sites, the industry still relies heavily on manual workforce. This thesis (in accordance with the Autosmolt 2025 project[18]) is devoted to find autonomous solutions to one of the more essential subgroup of these labors, namely the removal of dead fish and waste in the smolt tanks.

The solution pursued in this thesis made use of a revolute-joint robot manipulator, mounted on a Remotely Operated underwater Vehcile (ROV). The robot manipulator alone is the subject of study for this thesis

The theoretical research in the thesis concerns the modeling, path planning and control theory required to control the robot manipulator's end-effector towards a desired position, while the joints and links of the manipulator avoid potential obstacles in the workspace.

Solutions for path planning were implemented by employing the potential fields method. This was done by modeling attractive fields used to force the robot manipulator's joints towards a desired end configuration, while repulsive fields were implemented to enforce obstacle avoidance. Minimization of potentials in these fields was then used as an optimization objective, in order to incrementally move the manipulator to the desired end position.

The potential fields method is prone to get trapped in local minima of the optimization problem. A method called "randomised walk" was implemented in order to circumvent this issue.

A series of tests were conducted in order to verify the ability of the potential fields path planning algorithm to find collision-free paths between different configurations. The tests revealed that the problem of local minima had not been solved, and the path planning algorithm was not able to generate complete paths for a general set of configurations. An effort was made to improve the optimization algorithm by implementing Wolfe conditions, used to decide how far the optimization algorithm should move towards the minima. An error was found in the implementation of the Wolfe conditions, which meant that they could not be used in the optimization algorithm.

The paths generated were then used to move the robot manipulator between different configuration. The manipulator was not able to follow the full path.

The thesis concluded that the potential fields approach to path planning was inadequate for problems of this complexity. Most configurations would lead to incomplete paths, as the algorithm was trapped in a local minima. Some suggestions were given on how to improve the parts of the algorithm that were faulty due to faulty implementation.

Table of Contents

List of Figures

Chapter 1

Introduction

The contents of section 1.1 was written for a project preceding this thesis. It is included here to give relevant information on the background of this thesis

1.1 Background

Aquaculture is an important contributor to worldwide food production, and is a major industry in Norway. Salmon is the most heavily produced fish in Norway, and the land-based production sites used today were first developed in the 1980's[18]. Few advancements in the technology used has been made since then, and the industry relies heavily on a manual workforce. These land-based sites are used for the *smoltification phase*[20] of salmon production, which is a biological process in which salmon adapt to saltwater. The research organization SINTEF has begun work aiming at modernizing the production process by implementing automated solutions into smolt production sites[18][17]. This work lead to the project Autosmolt2025, with a vision of increasing profits, safety, and efficiency in the industry. One of the proposed solutions is the use of unmanned underwater vehicles (UUVs) to perform routine operations in the smolt tanks[11]. This specialisation project aims to contribute to the research in using UUVs in smolt production.

1.1.1 Autosmolt 2025

Present day smolt production plants operate on the same principles used since they were first developed in the 1980's[18][17]. Autosmolt 2025 aims to move away from dependencies on manual labor and towards automated solutions. Dependencies on manual labor are seen in several operations of the smolt production cycle, including washing of tanks and moving fish. Large facilities exacerbate the cost of manual labor, as workers need to physically move between different parts of the production site. By implementing the theory of Precision Fish Farming[PFF][8] at different stages of the smolt production process, Autosmolt works towards realizing smolt production within the framework of Industry 4.0. Figure 1.1 is an illustration their concept.

1.1.2 Robot manipulators

Robot manipulators are versatile tools that can be used in a wide variety of automation processes. Autosmolt2025 has previously worked on utilizing robot manipulators mounted on UUVs to perform various tasks in the tanks, e.g cleaning and removing debris. A master's thesis written in 2021 by Oscar Nissen[14] dealt with modeling a robot arm in an underwater environment, and a rudimentary controller was implemented to test the model. This project will also concern modeling, as well as path planning, of an underwater robot manipulator.

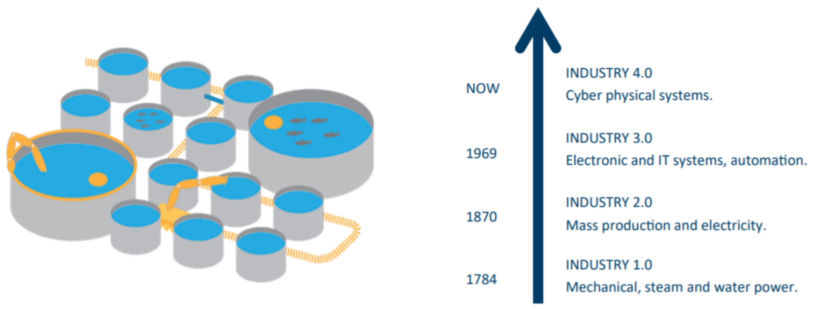

NOW
INDUSTRY 4.0
Cyber physical systems.
1969
INDUSTRY 3.0
Electronic and IT systems, automation.
1870
INDUSTRY 2.0
Mass production and electricity.
1784
INDUSTRY 1.0
Mechanical, steam and water power.

Figure 1.1: Autosmolt2025 aims to implement automation in several areas of smolt production.

The robot that was modeled in Nissen's thesis was the *Reach Bravo 7*, produced by the Australian company Blueprint Lab. The manipulator was chosen based on another thesis written by Pål Hofset Skeide in 2020[19].

SINTEF has chosen a new manipulator by the name of *Reach Alpha 5* to be used on the UUVs instead, which will be the subject of this project. The manipulator is a 5 degree-of-freedom robotic arm, with the 5th DoF being the adjustable end-effector. This end-effector can be exchanged for a variety of different tools, making the manipulator versatile in the tasks it can perform.

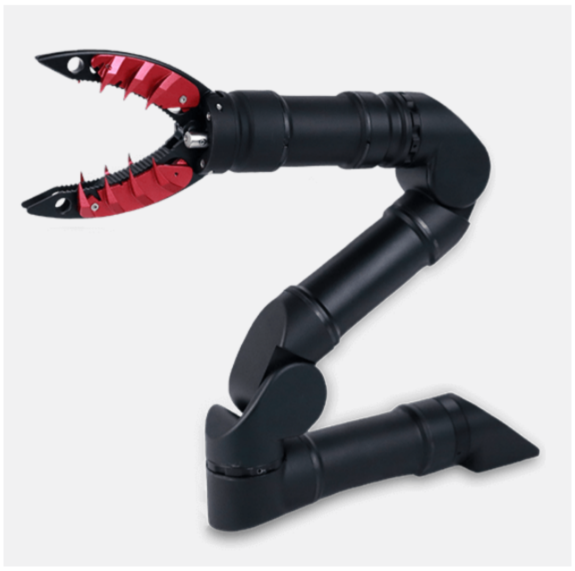
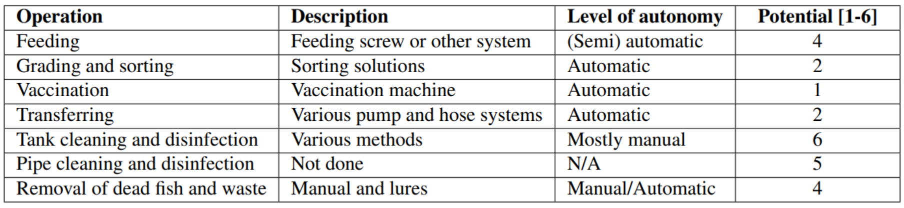


Figure 1.2: Reach Alpha 5, produced by Blueprint Lab.

1.2 Motivation

Pål Hofset Skeide outlined a table of potential operations for robot manipulators in smolt facilities in his thesis[17] (Figure 1.3)



Operation	Description	Level of autonomy	Potential [1-6]
Feeding	Feeding screw or other system	(Semi) automatic	4
Grading and sorting	Sorting solutions	Automatic	2
Vaccination	Vaccination machine	Automatic	1
Transferring	Various pump and hose systems	Automatic	2
Tank cleaning and disinfection	Various methods	Mostly manual	6
Pipe cleaning and disinfection	Not done	N/A	5
Removal of dead fish and waste	Manual and lures	Manual/Automatic	4

Figure 1.3: Operations in smolt facilities[17]

This thesis aims to work towards solutions to the autonomous-rated operation with the highest potential rating, namely removal of dead fish and waste. Removal of objects from smolt facilities is an essential task in the industry that needs to be performed, and it is likely a problem with a solution applicable to more generic settings, with a shorter path to commercial solutions. More concretely, the motivation for work in this thesis boils down to object removal in combination with object avoidance, which in turn motivates the application of robot modeling and control.

1.3 Software

The previous work done on the robot manipulator was programmed in Python 3[21], which includes the I/O-interface to operate the robot manipulator, in addition to a graphical interface (section 3.1) to monitor runtime performance. Resources for 3D trajectory visualizations have previously been implemented in MatLab[13], which remains the plotting tool used throughout this thesis. The ease of use and rapid prototyping capabilities makes these interpreted languages preferable over *compiled* languages. The algorithms implemented in this thesis have been written in MatLab.

1.4 Preliminaries

The theory covered in this thesis revolves around modeling, path-planning and optimal control on revolute-joint robots. Prior experience with linear algebra is recommended for the theory concerning forward and inverse kinematics. Euler-Lagrange dynamics is thoroughly explained in Robot Modeling and Control [22], but is more briefly presented in this thesis. NTNU-courses TMA4110 Linear Algebra and TTK4195 Modeling and Control of Robots, along with TTK4135 Optimization and Control should provide a good preliminary understanding of the topics discussed in this thesis.

1.5 Structure

The structure of this thesis is intended to clearly separate the work into different stages, based on the genericness of the concerned topics. The first section presents the theory used in the thesis. The theoretical sections provide foundations for the computation of joint positions (Kinematics), along with methods for relating linear velocity in the manipulator to angular velocity in the joints (Jacobian). Kinematics and the Jacobian are used to plan and enforce motion for the robot manipulator (Motion and Path planning).

The Implementation-chapter describe how the presented theory is used to implement algorithms, along with heuristics and other details concerning the MatLab-implementation.

The resulting implementation is evaluated by running a set of tests on the robot manipulator, where each test is intended to show the robustness of the implementation under different conditions. The implementation of these tests are described in chapter 4.

The results from the performed tests are presented in chapter 5, which provides illustrative convergence plots for the individual joints, along with 3-dimensional plots for the whole robot manipulator. The tests were performed using different sets of parameters, which are correspondingly divided into separate subsections.

Chapter 6 aims to relate the results from chapter 5 back to the introduced theory, in order to determine the robustness and quality of the implementation. The general convergence failure is discussed along with remedies which may be achievable within the scope of the introduced theory.

Final conclusions are made in chapter 7, where the discussed aspects of the results are summarized and related to the initial problem statement of the thesis.

1.6 Problem Statement

This thesis will be a continuation of the TTK4550 specialisation project that was completed last semester. The report for that project will be submitted alongside this thesis. The problem statement of this thesis is a combination of 3 elements:

- **Software framework development**: Develop software for collision-free motion planning using the potential fields method.
- **Virtual experiments to verify control system components**:
  1. Test of basic motion planning capabilities in open air.
  2. Test of collision avoidance capabilities for avoiding obstacles in the manipulator workspace.
- **Physical experiments with the Reach Alpha 5**: Conduct physical experiments using results from virtual experiments with the motion planning algorithm.

The complete project description is included in appendix .1

Chapter 2

Theory

Sections 2.1 through 2.3.1.4 were written for the project preceding this thesis. These sections were included as they present the theory used for large parts of the work done in this thesis.

2.1 Kinematics

Kinematics refers to the position and motion of the manipulator links and joints without considering any internal or external forces acting on the system. Solving the problem of kinematics is the first step in manipulator motion control, as it is a fundamental component when modeling any manipulator. This section will discuss two sub-problems of kinematics, namely *the forward kinematics problem* and *the inverse kinematics problem*. The theory presented in this section is based on methods used in the textbook *Robot Modeling And Control*[22].

2.1.1 Forward kinematics

Forward kinematics relates to the problem of mapping the positional joint variables of a robot manipulator to the position and orientation of the end-effector. Solving the forward kinematics problem is important when working with any robot manipulator, as it is needed for designing controllers, path planning, and simulation. This section will go through the necessary theory of how a robot manipulator is modeled, and show how the forward kinematics problem can be solved using what is known as the Denavit-Hartenberg convention.

2.1.1.1 Kinematic chains

Any robot manipulator can be viewed as a set of links connected by joints. These sets of links and joints are referred to as a kinematic chain, as motion in one joint will also cause motion in all subsequent links and joints. The connective joints can vary in complexity, from simple prismatic joints that extend and retract along a single axis, to spherical joints that can rotate in three degrees of freedom. However, any joint with multiple degrees of freedom can be thought of as a combination of single degree-of-freedom joints connected by links of length zero. In the case of the spherical joint, a chain of three single degree-of-freedom revolute joints rotating about orthogonal axes will be able to produce the same motion in the attached link. By making the assumption that all manipulator joints are single degree-of-freedom, the motion in each joint can be quantified by a single number: joint displacement for prismatic joints and joint angle for revolute joints.

For a robot manipulator with n joints there are $n + 1$ links, as each joint connects two links. Common convention is to number joints from 1 to n and links from 0 to n starting from the base of the manipulator[22][4]. This means that joint i connects links $i - 1$ and i. Motion in link i is then produced by actuating joint i. Link 0 is the base of the manipulator and remains stationary for actuation of any joint. Each joint i in the chain is associated with their respective joint variable denoted q_i. For the two types of single degree-of-freedom joints, q_i is defined as follows:

$$q_i = \begin{cases} d_i & \text{if joint i is prismatic} \\ \theta_i & \text{if joint i is revolute} \end{cases} \tag{2.1}$$

where d_i denotes displacement along the axis of actuation by joint i, and θ_i denotes the angle of rotation in joint i.

The next step in analyzing the kinematic chain is to describe how change in joint variable q_i affects the rest of the chain. This is done by attaching a coordinate frame to each link in the chain. The i^{th} frame is most commonly attached to the center of the i^{th} joint, but certain manipulator configurations may require the frame to be placed somewhere else on the i^{th} link (more on this in section 2.1.1.2). When frame i is attached rigidly to link i, any point on link i will remain constant in the i^{th} frame. Figure 2.1 shows a diagram of a 4-link manipulator with attached coordinate frames. Any change in coordinates of a point viewed from frame i can be described in frame $i-1$ through a *homogenous transformation matrix* relating the position and orientation (pose) of the two frames. These transformation matrices take the form

$$H = \begin{bmatrix} R & o \\ 0 & 1 \end{bmatrix} \tag{2.2}$$

where R is a 3×3 rotation matrix which describes the orientation of the point, and o is a vector describing the position in Cartesian coordinates. Given a point P^i fixed in the i^{th} frame, the same point viewed from the $(i-1)^{th}$ frame can be found using the equation

$$P^i = \begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix}$$
$$P^{i-1} = HP^i \tag{2.3}$$

Note that there is a 1 attached to the bottom of the point vector P^i. This is added because the homogenous transformation matrix H is of dimention 4×4. The bottom row of the matrix will not contribute to the solution p^{i-1}.

Let A_i denote the homogenous transformation matrix that gives the pose of coordinate frame $o_i x_i y_i z_i$(see figure 2.1) in frame $o_{i-1}x_{i-1}y_{i-1}z_{i-1}$. This means that a moving point in frame $i-1$ can be accurately described through A_i and the constant coordinates given in frame i. Furthermore, a homogenous transformation matrix that gives the pose of frame $o_j x_j y_j z_j$ in frame $o_i x_i y_i z_i$ is given by the formula

$$T_j^i = A_i A_{i+1} A_{i+2} ... A_j \tag{2.4}$$

This matrix is called a commonly denoted T. Through the use of this formula, one can calculate a transformation matrix that relates the pose of the end-effector to the frame at the base of the chain. This base frame is often named the *world frame* or *inertial frame*. Thus, the complete homogenous transformation matrix from world frame to end effector is calculated as

$$T_n^0 = A_i A_{i+1} A_{i+2} ... A_{n-1} A_n \tag{2.5}$$

Similar to how a point on link i is constant in the i^{th} frame, the position of the end-effector is constant in the n^{th} frame. Let o_n^0 denote the position of the end-effector with respect to the world frame. The complete matrix will take the form

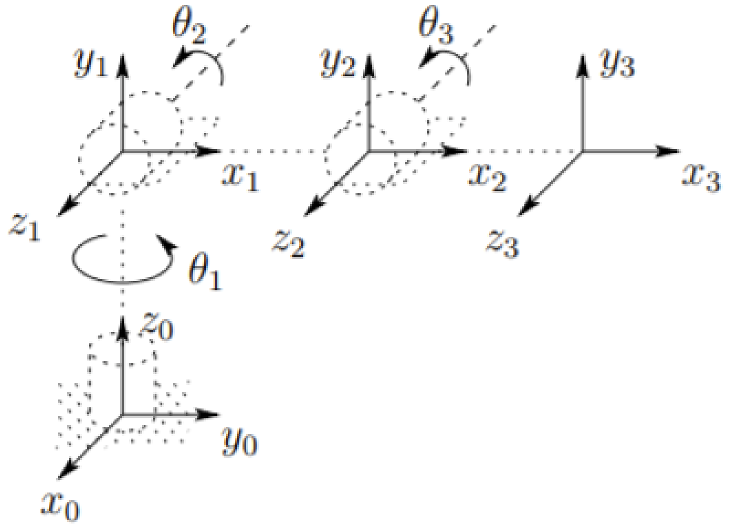


Figure 2.1: Diagram of an elbow manipulator with attached coordinate frames. Figure from *Robot Modeling And Control*[22]

$$T_n^0 = \begin{bmatrix} R_n^0 & o_n^0 \\ 0 & 1 \end{bmatrix} \tag{2.6}$$

where R_n^0 is a 3×3 rotation matrix.

By calculating this homogenous transformation matrix, one can solve the forward kinematics problem. Each matrix A_i is a function of q_i, and the complete transformation matrix allows for mapping of any pose of the end-effector to the world frame (or any other frame) given a any set of joint values. The following section will show how these homogenous transformation matrices can be created using a systematic approach known as the *Denavit-Hartenberg convention.*

2.1.1.2 The Denavit-Hartenberg convention

There are certain manipulators that are constructed in such a way that the forward kinematics problem can be solved using simple geometry. The two-link planar elbow manipulator is an example of such a system, where the position of the end-effector can be solved by hand. Figure 2.2 shows the manipulator and the method used to find the relation between the position (x, y) and the joint values (θ_1, θ_2). Equations 2.7 and 2.8 show the resulting relation.

$$x = a_1 \cos(\theta_1) + a_2 \cos(\theta_1 + \theta_2) \tag{2.7}$$
$$y = a_1 \sin(\theta_1) + a_2 \sin(\theta_1 + \theta_2) \tag{2.8}$$

Similarly one can derive the orientation of (x, y) using the same methods.

However, solving the forward kinematics problem for a general n-link manipulator is a much more complex problem. Robotic arms are often designed to work in a 3-dimensional space, which means that an arbitrary homogenous transformation matrix would require three numbers to describe the position, and three numbers to describe orientation of the end-effector. The Denavit-Hartenberg convention is a systematic approach to kinematic analysis which greatly simplifies the challenge by reducing the required number of parameters to four. The method was first proposed by Jacques Denavit and Richard Hartenberg in 1955[5], and has since been a popular choice and industry standard for solving forward kinematics[22][4]. The idea is to set specific criteria on how the

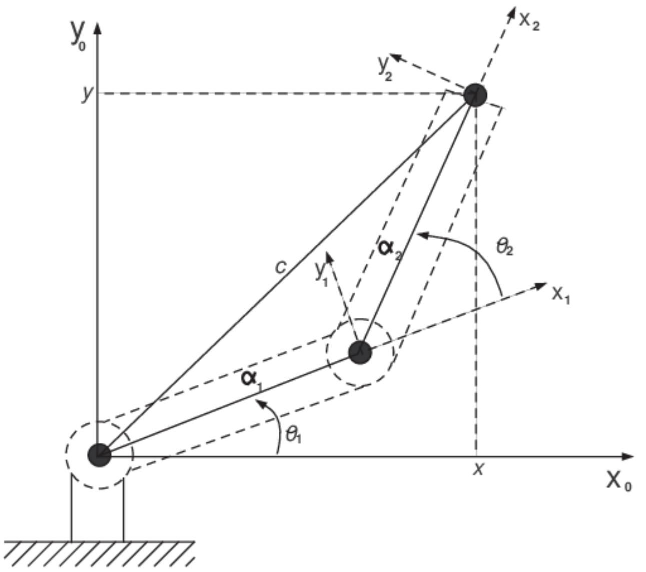


Figure 2.2: Two-link planar elbow manipulator. Figure from *Robot Modeling and Control*[22]

coordinate frames $o_i x_i y_i z_i$ are attached to each link of the manipulator. While each frame must be rigidly attached to the corresponding link, one has great freedom in choosing the position and orientation of said frame. By following the criteria set by the Denavit-Hartenberg convention, the necessary parameters for describing a homogenous transformation will be decreased to four.

There are two necessary restrictions on placement and orientation of the coordinate frames which must be upheld for the reduction in parameters to be possible. Those restriction are:

1. The axis x_i is perpendicular to z_{i-1}
2. The axis x_1 intersects the axis z_{i-1}

Figure 2.3 shows an example of two frames placed in this way. By studying this figure one can see the physical representation of the four parameters necessary to create a homogenous transformation matrix. These parameters are called Denavit-Hartenberg parameters(DH-parameters). These parameters are given as follows:

- a_i: Distance from z_{i-1} to z_i along the x_i-axis. This parameter is named *link length.*
- α_i: Angle between z_{i-1} and z_i as seen from a plane normal on x_i. This parameter is named *link twist.*
- d_i: Distance from the origin o_{i-1} to the intersection of x_i and z_{i-1} along the z_{i-1}-axis. This parameter is named *link offset.* This will be the joint variable for a prismatic joint.
- θ_i: Angle from x_{i-1} to x_i seen from a plane normal to z_{i-1}. This parameter is named *joint angle.* This will be the joint variable for a revolute joint.

By following these restrictions one can guarantee existence and uniqueness of a solution for the homogenous transformation, as show by Denavit and Hartenberg[5][6]. It is worth noting that these restrictions allow for several different configurations of the coordinate frames. One thing to note in particular is that the frame does not necessarily have to be located within the corresponding link, as long as it is rigidly attached. There are, however, some useful guidelines to how one should place the frames in an intuitive way. One such set of guidelines goes as follows:

1. Choose z_i to align with the actuation axis of the joint $i+1$. Do this for all coordinate frames.

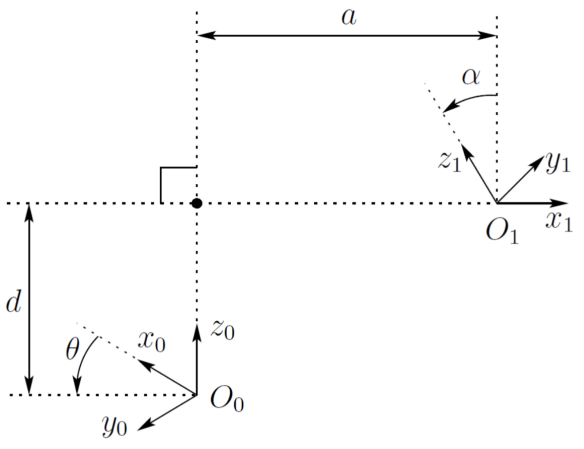


Figure 2.3: Coordinate frames placed in accordance with the Denavit-Hartenberg convention. The figure gives a visual representation of the Denavit-Hartenberg parameters. Figure found in *Robot Modeling And Control*[22]

2. Choose o_0 along the z_0 axis(most intuitively where the manipulator connects to the ground). x_i is chosen along the shortest distance between z_{i-1} and z_i, and o_i is chosen where x_i intersects z_i. Do this for all coordinate frames. In cases where z_{i-1} and z_i are parallel, one is free to choose the placement of o_i. In cases where z_{i-1} intersects z_i, x_i is chosen orthogonal to bot axes and o_i is chosen at the point of intersection.

3. Choose y_i orthogonal to z_i and x_i according to the right-hand rule ($y_i = z_i X x_i$). Do this for all frames.

Once all coordinate frames have been placed according to the DH-convention, the homogenous transformation matrices A_i can be calculated. A_i is defined as the product of four basic transformations about each DH-parameter:

$$\begin{aligned} A_i &= Rot_{z,\theta_i} Trans_{x,d_i} Trans_{x,a_i} Rot_{x,\alpha_i} \\ &= \begin{bmatrix} \cos(\theta_i) & -\sin(\theta_i)\cos(\alpha_i) & \sin(\theta_i)\sin(\alpha_i) & a_i\cos(\theta_i) \\ \sin(\theta_i) & \cos(\theta_i)\cos(\alpha_i) & -\cos(\theta_i)\sin(\alpha_i) & a_i\sin(\theta_i) \\ 0 & \sin(\alpha_i) & \cos(\alpha_i) & d_i \\ 0 & 0 & 0 & 1 \end{bmatrix} \end{aligned} \tag{2.9}$$

where $Rot_{a,b}$ is a rotation of b radians about the a-axis, and $Trans_{a,b}$ is a linear transformation of length b along the a-axis. Using the DH-convention has lead to an explicit solution for calculating the homogenous transformation matrix, and thus a general solution to the forward kinematics problem. Take note that while there are different ways to place each coordinate frame in the kinematic chain, the final transformation matrix T_n^0 will always be the same for any valid set of placements.

2.1.2 Inverse kinematics

While the forward kinematics problem relates to mapping joint variables to the pose of the end-effector, the inverse kinematics problem relates to mapping the end-effector pose to a set of joint variables. By solving this problem, it is possible to calculate the exact joint variables needed to obtain a desired pose for the end-effector, which is very important when one wants the robot to interact with objects in the workspace. The inverse kinematics problem is generally more difficult than forward kinematics, and the complexity scales with the number of joints. The general problem

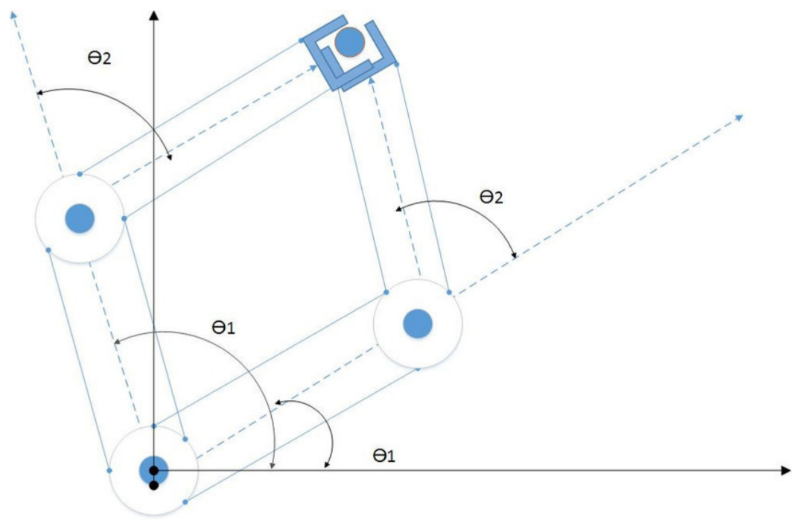


Figure 2.4: Two different joint configurations lead to the same position for the end-effector[15]

formulation for inverse kinematics can be state as: Given the desired homogenous transformation $H = \begin{bmatrix} R & o \\ 0 & 1 \end{bmatrix}$, find a solution to the equation

$$T_n^0(q_1, \dots, q_n) = H \tag{2.10}$$

Since both matrices are of size 4×4, equation 2.10 results in sixteen sub-equations on the form $T_{ij}(q_1, \dots, q_n) = h_{ij}$. As was show in section 2.1.1.2 and equation 2.9, as the number of joints n increases, each element T_{ij} becomes more complex. The goal is to use these sixteen equations to solve for the joint variables

$$q_i = f_i(h_{11}, h_{12}, \dots) \tag{2.11}$$

Unlike with forward kinematics, there is no guarantee for a unique solution to this problem. The equations may not be solvable, but even if they are, doing so might not be feasible due to the difficulty. There is also the problem that there exist multiple solutions for a given pose of the end effector. This is often the case for robots mimicking human arms, as their ”elbows” may rotate two ways for the same end-effector pose. Figure 2.4 illustrates how two different angles in the elbow result in the same position.

When trying to solve inverse kinematics, it is common to split the problem into two decoupled sub-problems: inverse position, and inverse orientation. Separating these elements and solving them separately can often make finding a solution easier, but also here there is no guarantee. The difficulty in solving inverse kinematics is often an influence on how robots are built, as certain structures allow for the use of this simplification. Another reason to split the problem in two is that some tasks may only require the position of the end-effector, not the orientation.

When dealing with inverse position specifically, equation 2.10 is one of two common approaches. The other method is to solve for q_i geometrically. For manipulators with few joints, typically two or three, this may be the preferred way to solve inverse position. The robot in figure 2.4 is an example of a manipulator where the inverse kinematics can be solved geometrically.

2.1.3 Jacobian

In section 2.1.1 the forward kinematics problem was solved, which relates joint variables to the position of the end effector (or any point on the manipulator) in Cartesian space. It is now necessary to relate joint velocities to velocity in the manipulator. This is done through the use of the *Jacobian* matrix, denoted J. The Jacobian consists of two parts: the linear velocity Jacobian, and the angular velocity Jacobian. These matricies maps joint velocities to manipulator velocity as follows

$$J = \begin{bmatrix} J_v \\ J_\omega \end{bmatrix} \tag{2.12}$$

$$v = J_v \dot{q} \tag{2.13}$$

$$\omega = J_\omega \dot{q} \tag{2.14}$$

For an n-link manipulator the Jacobian will be an $6 \times n$ matrix, where the top three rows correspond to linear velocity, and the bottom three rows correspond to angular velocity. Similarly to the inertia tensor, the Jacobian changes with the configuration of the manipulator. Because of this, it is necessary to calculate the homogenous transformation matrix T_n^0 derived in section 2.1.1 to know the position of each body frame. From this matrix, the values of orientation in the z-direction and the position o_i will be used to calculate J. The formula for calculating the Jacobian is as follows

$$\begin{aligned} J_v(i) &= \begin{cases} z_{i-1} \times (o_n - o_{i-1}) & \text{for revolute joint i} \\ z_{i-1} & \text{for prismatic joint i} \end{cases} \\ J_\omega(i) &= \begin{cases} z_{i-1} & \text{for revolute joint i} \\ 0 & \text{for prismatic joint i} \end{cases} \end{aligned} \tag{2.15}$$

where $J_v(i)$ denotes the i^{th} column of J_v, and z_i denotes the third column of R_i^0. There is a unique set of Jacobians, denoted J_{v_i} and J_{ω_i}, for each frame on the manipulator, as joint motion higher in the kinematic chain will not impact the lower part. However, each Jacobian is built using the same formula, with any column $j > i$ set to zero. Thus, calculating each Jacobian is trivial once J_{v_n} and J_{ω_n} are found. Now that J is calculated, equations 2.13 and 2.14 will give linear and angular velocities v and ω.

2.2 Motion planning

Previous sections have focused on the manipulators inherent properties without taking the workspace into account. For autonomous motion, it is important to consider potential obstacles that could limit the manipulator's range of motion. This section will cover theory used for planning a collision-free path for the manipulator trough the workspace. Motion planning can be divided into sub-problems of path planning and trajectory planning. Path planning relates to generating a geometric path in space which avoids any obstacles, while trajectory planning introduces constraints on parameters like time, velocity, acceleration etc. to generate the proper motion in the manipulator to follow said path. One may use a wide variety of approaches to path and trajectory planning depending on the workspace and the task the manipulator seeks to achieve.

To formalize the motion planning problem, the concept of the *configuration space* is introduced. The configuration space is the set of all possible combinations of joint values $(q_1, \dots, q_n)$, denoted as $\mathcal{Q}$[22]. Let $\mathcal{W}$ denote the Cartesian workspace, and $\mathcal{O}_i$ denotes the obstacles in the workspace. $\mathcal{A}$ denotes the robot itself, and $\mathcal{A}(q)$ denotes a specific configuration. The problem of motion planning is to avoid any configuration q which causes the manipulator to make contact with an obstacle. These configurations are defined as

$$\mathcal{QO} = \{q \in \mathcal{Q} | \mathcal{A}(q) \cap \mathcal{O} \neq 0\} \tag{2.16}$$

where $\mathcal{QO}$ is the set of all such configurations.

2.3 Path planning

Path planning is the problem of finding a path from a given initial configuration q_s to a desired target configuration q_f in a way that avoids any collision with obstacles in the workspace. There are many ways to approach this problem, all with different strengths and weaknesses depending on the specifics of the manipulator and the workspace it will operate in. One important factor is whether or not the workspace is known beforehand. Seeing as how the Reach Alpha 5 will be mounted on a ROV in the future, it is assumed that the workspace is unfamiliar.

Computing the set $\mathcal{QO}$ of configurations which causes collisions becomes more difficult for increasingly complex manipulators. When working with manipulators that can both translate and rotate, with multiple degrees of freedom, solving for $\mathcal{QO}$ quickly becomes infeasible. Because of this, the path planning problem will most often consist of creating a optimization based search algorithm that incrementally generates set points to take the manipulator from the start point q_s to the target destination q_f. This search is done by minimizing the error between configurations q_s and q_f in the configuration space. For this project, the *potential fields* method will be used to solve the path planning problem. This is due to the relative simplicity of the method, as well as thorough documentation in literature and scientific publications. A gradient descent algorithm will be used to minimize the configuration error. More on this algorithm in section 2.3.1.4.

2.3.1 Potential fields

The potential fields method is based around the use of an *artificial vector field*, $U(q)$, to steer the search algorithm. This field will induce artificial forces on the robot which pull toward the target configuration q_f, while pushing away from any obstacles[22][1][16]. The field is built up of attractive and repulsive components acting on the robot.

$$U(q) = U_{att}(q) + U_{rep}(q) \tag{2.17}$$

$$\tau(q) = -\nabla U(q) \tag{2.18}$$

The path can then be generated using an optimization algorithm to find the global minima of the field U. The algorithm uses the negative gradient $\tau(q)$ of $U(q)$ to find the shortest path which minimizes the potential function, where $\tau(q)$ can be considered as a vector of torques acting on the robot in the configuration space. By letting the torque vector act on the robot, it will move along this optimized path. The optimization algorithm used in this project is the *gradient descent algorithm*. The gradient descent algorithm is explained in more detail in section 2.3.1.4. Figure 2.5 illustrates how the manipulator is affected by the potential fields.

Due to the complexity of mapping the configuration space, the field U will be defined on the manipulator workspace itself. Specifically, each coordinate frame in the kinematic chain will be subjected to its own potential field U_i defined at its origin. By using the methods discussed in sections 2.1.1, 2.1.2, and 2.1.3, the position of each frame and target destination can be found through kinematics from the manipulator configuration q and target configuration q_f. The manipulator Jacobian J can then be used to define motion of the robot in the configuration space. The next section will discuss how the fields U_i are defined.

2.3.1.1 The attractive field

The attractive field $U_{att,i}$ is the component of U_i that attracts frame $o_i x_i y_i z_i$ to its target destination defined by the target configuration q_f. The attractive field needs to satisfy certain criteria to be suitable for this task. Firstly, the field should increase with distance to o_i so the pull is stronger the further away the manipulator is. The simplest field which achieves this will grow linearly with this distance. Let $o_i(q)$ denote the position of frame i, and q_f denote the target configuration. The attractive field then takes the form

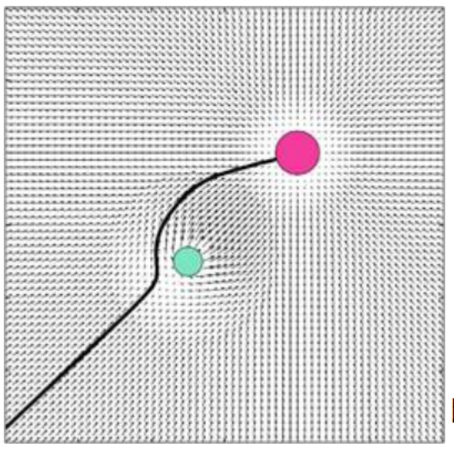


Figure 2.5: An object being attracted towards the goal by an attractive field, while being repelled away from an obstacle by a repulsive field. Figure created by Hani Safadi[16].

$$U_{att,i}(q) = \gamma_i \left\| o_i(q) - o_i(q_f) \right\| \tag{2.19}$$

$$-\nabla U_{att,i}(q) = \begin{cases} -\gamma_i \frac{(o_i(q)-o_i(q_f))}{\|o_i(q)-o_i(q_f)\|} & \text{if } \|o_i(q) - o_i(q_f)\| \neq 0 \\ 0 & \text{if } \|o_i(q) - o_i(q_f)\| = 0 \end{cases} \tag{2.20}$$

where γ_i is a scaling factor for the attractive force. The problem with this field is that the gradient is clearly defined everywhere but the target position, which might lead to instability. This problem can be avoided by using a field which follows the second criteria, namely that the field is continuously differentiable. The simplest such field will grow quadratically with distance

$$U_{att,i}(q) = \frac{1}{2}\gamma_i \left\| o_i(q) - o_i(q_f) \right\|^2 \tag{2.21}$$

$$-\nabla U_{att,i}(q) = -\gamma_i(o_i(q) - o_i(q_f)) \tag{2.22}$$

This field solves the problem of having a discontinuous gradient, but introduces another problem. The gradient scales linearly with the distance $o_d(q)$ from $o_i(q)$ to $o_i(q_f)$, which means it is unbounded. This could lead to excessively large attractive forces. It is therefore common to use some combination of different fields to avoid these problems. By using the quadratic expression 2.21 when $o_d(q)$ is small, and the linear expression 2.19 when the model is large, both problems are avoided.

2.3.1.2 The repulsive field

The repulsive field $U_{rep,i}$ is the component of U_i that pushes frame $o_i x_i y_i z_i$ away from any obstacles. The criteria for this field are different from those of the attractive field, as the robot cannot collide with obstacles at any point. This means the field should infinitely increase as $o_d(q)$ approaches zero. The field should also not interfere with the robot when the obstacle is far away. The parameter d_0 is introduced as the distance at which the repulsive field should start taking effect. For any $o_d(q) > d_0$, the field is zero. The simplest field which follows these criteria while also being continuously differentiable is

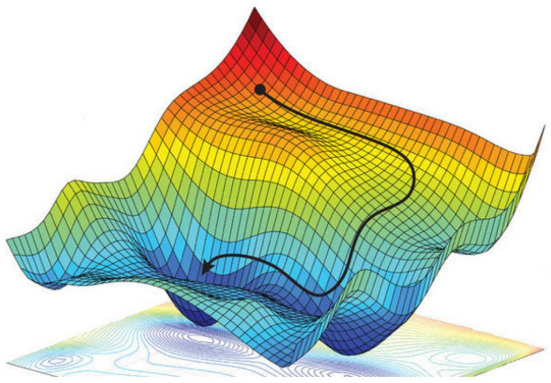


Figure 2.6: Gradient descent algorithm search in the configuration space. Figure created by Alexander Amini[1].

$$U_{rep,i}(q) = \begin{cases} \frac{1}{2}\delta_i \frac{1}{o_d(q)} & ; \text{if} \quad o_d(q) \leq d_0 \\ 0 & ; \text{if} \quad o_d(q) > d_0 \end{cases} \tag{2.23}$$

$$-\nabla U_{rep,i} = \begin{cases} \delta_i \frac{1}{o_d(q)} \frac{1}{o_d(q)^2} \nabla o_d(q) & ; \text{if} \quad o_d(q) \leq d_0 \\ 0 & ; \text{if} \quad o_d(q) > d_0 \end{cases} \tag{2.24}$$

where

$$\nabla o_d(q)|_{x=o_i(q)} = \frac{o_i(q) - b}{\|o_i(q) - b\|} \tag{2.25}$$

and b is the point on the obstacle closest to o_i.

2.3.1.3 Creating joint torques for the search algorithm

Once the potential fields have been constructed, it is time to map the forces generated by these fields into joint torques. This is done using the manipulator Jacobian $J(q)$. These torques are generated using the equation

$$\tau(q)_i = -J_{v_i}(q)^T \nabla U_i(q) \tag{2.26}$$

where $J(q) = \begin{bmatrix} J_v(q) \\ J_\omega(q) \end{bmatrix}$. $J_\omega(q)$ is not considered in this case, as U only generates attractive and repulsive forces, not torques. One set of torques will be generated for every field U_i. The sum of these torques will be the input that optimizes the path of the manipulator.

2.3.1.4 The gradient descent algorithm

Gradient descent is a commonly used algorithm used to solve a large variety of optimization problems, like machine learning, manipulator control, and path planning[1][10][16][7][2]. The algorithm uses what is known as a line search method which seeks to minimize the objective function by moving along the *search direction* p. In the case of gradient descent, p is the negative gradient of the objective function. By changing the values of the function input in the search direction, the function value will decrease. By incrementally applying this method, the function value will decrease until it reaches a minima. Figure 2.6 shows a visualisation of this process.

The algorithm starts at the initial configuration q_s of the robot, and incrementally steps along the negative gradient to minimize the function potential. At each increment, a new set point is created in the configuration space. The step used to find each set point is defined as

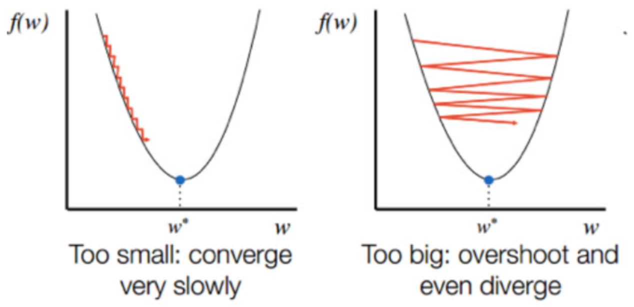
$$x_{k+1} = x_k + \alpha_k p_k \tag{2.27}$$

where x_k is the function input at the k^{th} iteration, p_k is the search direction, and α_k is an adjustable parameter used to decide the size of the step. When this method is applied to the potential fields path planner, q becomes the function input, and the normalized τ becomes the search direction. Pseudocode for the full search algorithm can be written as

```
**Gradient descent algorithm**
$q^0 \leftarrow q_s, i \leftarrow 0$
**while** $\|q^i - q_f\| > \epsilon$ **do**
    $q^{i+1} \leftarrow q^i + \alpha^i \frac{\tau(q^i)}{\|\tau(q^i)\|}$
    $i \leftarrow i + 1$
**end while**
return $< q^0, q^1, \ldots, q^i >$
```

where q^i denotes q at the i^{th} iteration. Parameter ϵ is the threshold for how close the manipulator needs to be the target q_f for the motion to be considered complete. This parameter is chosen based on the users requirement for accuracy, and may vary for different tasks.

Choosing an appropriate step size is important when using a gradient descent algorithm, as it can drastically influence both runtimes and stability. If the step is too large, the next set point might be placed beyond the target destination. On the other hand, small steps might lead to long runtimes. This is illustrated in figure 2.7. There are many different methods one can use to chose α_i. The simplest way is to use a constant value $\alpha_i = \alpha_0$ for each iteration of the gradient descent algorithm. Doing so puts the user at risk of taking too large or too small steps. This is because the step needs to be adequately small as the search approaches the desired configuration to not overshoot. Such a step will often be inefficient in the earlier stages of the search. Therefore, it is most common to use an *augmented step*, which changes in size for every iteration. The choice of how to augment the step size α_i depends entirely on the optimization task itself, and the requirements set by the user.



f(w)
w*
w
Too small: converge very slowly
f(w)
w*
w
Too big: overshoot and even diverge

Figure 2.7: The effects of choosing large and small step sizes. Figure created by Sahdev Kansal[10]

One approach used to decide an appropriate step size α_i for several line search methods is to apply the *Wolfe conditions* [9, p.33]. This approach is explained in the following section.

2.3.1.5 The Wolfe conditions

As mentioned in the previous section, selecting a step size α_k comes with drawbacks. If α_k is too large, the search will not be accurate, while a small α_k leads to an inefficient search. Two conditions in the form of inequalities are introduced to find an appropriate value for α_k. These conditions are called the *Armijo condition* and *curvature condition* respectively. The Armijo condition is fulfilled by upholding the inequality

$$f(x_k + \alpha_k p_k) \leq f(x_k) + c_1 \alpha_k \nabla f_k^T p_k \tag{2.28}$$

where $c_1 \in (0,1)$. This inequality states that the function at iteration $k+1$ must be smaller than the function at the current iteration with a weighted step in the direction of the gradient, scaled by both α_k and the *directional derivative* $f_k^T p_k$. One can decide how much the function needs to be decreased to satisfy this condition by adjusting the parameter c_1.

The curvature condition ensures that the step does not get too small. This condition is met by upholding the inequality

$$\nabla f(x_k + \alpha_k p_k)^T p_k \geq c_2 \nabla f_k^T p_k \tag{2.29}$$

The inequality says that the gradient at the next step must be greater than the parameter $c_2 \in (c_1, 1)$ times the gradient at the next step, given $\alpha_k = 0$.

2.3.1.6 Local minima

One problem with the gradient descent algorithm is that it can get stuck in a local minima. A local minima in the context of potential fields would be a joint configuration where there is no direction to go which reduces the field $U(q)$. This is likely to happen when the manipulator reaches around an obstacle or gets trapped between multiple repulsive fields of the same strength. Figure 2.8 shows a configuration where the manipulator receives attractive and repulsive forces of equal magnitude.

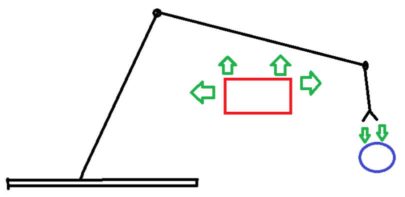


Figure 2.8: Manipulator stuck in a local minima. The repulsive forces of the red block cancel the attractive forces of the blue circle. Reaching the global minima would require increasing the objective function by backing away from the target blue circle.

One approach to avoiding local minima is to use a gradient descent with *randomized walk*. This algorithm searches along the negative gradient until it gets trapped in a local minima, at which point a random walk is executed to move the manipulator to a different joint configuration. The first element in this approach is to recognize when the algorithm is stuck in a local minima. There is no general solution for guaranteed detection of a minima, which means the problem is often solved using a practical approach. One such idea is to assume the search is stuck when a series of set points are all very close together. This would require some knowledge of the expected change between set points. The second element is to generate the random walk. This walk consists of a series of random jumps in each joint. Specifically, the random walk takes adds or subtracts q_v to each joint at random s times. The probability of adding or subtracting q_v equals $1/2$, resulting in an even distribution. Each step takes the form

$$q_{s+1} = [q(1)_s \pm q_v,\ q(2)_s \pm q_v,\ \ldots,\ q(n)_s \pm q_v] \tag{2.30}$$

where $q(n)$ is the n^{th} joint. The random walk can be characterized by its density function

$$p_i(q_i, s) \approx \frac{1}{q_v\sqrt{2\pi s}} exp(-\frac{q_i^2}{2q_v^2 s}) \tag{2.31}$$

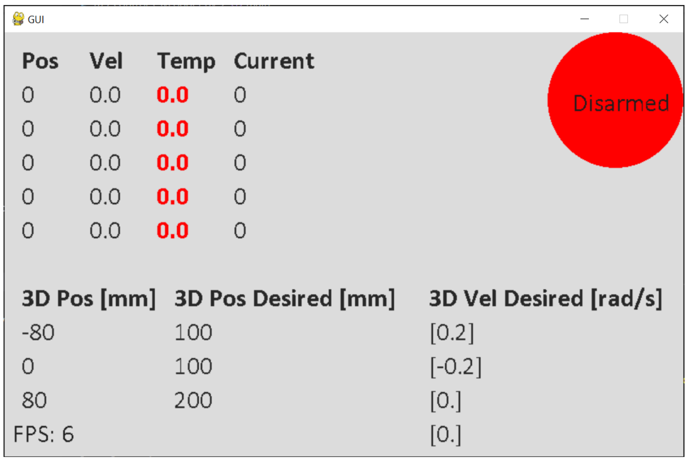
which has a variance of $q_d^2 t$. This value shows how far the manipulator can be expected to walk, given the expectancy value of the density function being zero[9]

Chapter 3

Implementation

3.1 Creating a program for operation of the Reach Alpha 5 manipulator

When the Reach Alpha 5 was made available for experiments, project supervisor Bent Haugaløkken had already created python code for basic manual and automatic operation, using the built in PID-controllers. The code had functionalities for manual control using a joystick, automatic control for a single reference joint configuration, as well as capabilities for displaying important data live on a GUI. Figure 3.1 shows the user interface.


GUI

Pos	Vel	Temp	Current
0	0.0	0.0	0
0	0.0	0.0	0
0	0.0	0.0	0
0	0.0	0.0	0
0	0.0	0.0	0

Disarmed

3D Pos [mm]	3D Pos Desired [mm]	3D Vel Desired [rad/s]
-80	100	[0.2]
0	100	[-0.2]
80	200	[0.]
FPS: 6		[0.]

Figure 3.1: GUI used for displaying data during live operation

The code was expanded to include more functionality in automatic operation. The code would no allow for a series of references to the joint positions to be used to move the manipulator through a list ow set points. The code was also expanded to include functionality for printing data into .mat-files which can be loaded directly to MatLab for plotting.

To test was conducted to verify the functionality of the new reference system, as well as the exportation of data to MatLab. The manipulator was given an initial position of $q_s = [pi/2 \quad pi/2 \quad pi/2 \quad pi/2 \quad pi/2]$. A reference $q_f = [pi \quad pi/2 \quad pi/2 \quad pi/2 \quad pi/2]$ was then given to to the manipulator, and once the manipulator reached that reference, a new reference would be pushed with the next joint value $q(i) = pi$. This would continue until the manipulator reached the final configuration $q_f = [pi \quad pi \quad pi \quad pi \quad 400$. Joint 5 is the end effector and has a different range of inputs than the other joints. Figure 3.2 shows the result of the test.

All joints converged to the target value. The data was then sent to MatLab for plotting.

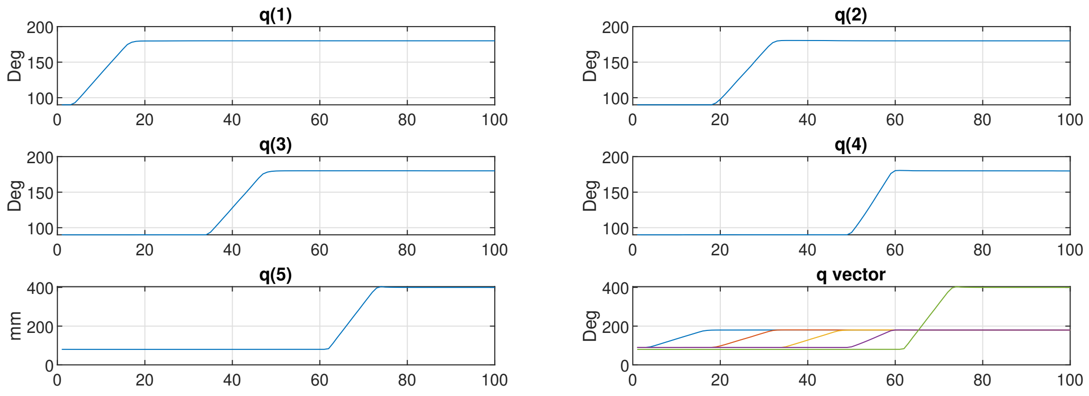


Figure 3.2: Convergence of each joint of the Reach Alpha 5 to reference values.

3.2 Re-implementing the potential fields path planning algorithm

In the project that preceded this thesis, a path planning algorithm was created for motion planning for the Reach Alpha 5. However, the program did not run properly, going so far as to crash when obstacles were introduced into the workspace, and was therefore unsuitable for gathering meaningful results retaining to the validity of the potential fields approach. The project concluded that the approach could not be verified. It was therefore deemed necessary to re-implement the algorithm from the ground up for this thesis. The new algorithm is based on the theory discussed in section 2.3.1, but will also include methods for selecting suitable gradient steps, avoiding local minima, and a new approach to how the repulsive fields acts upon the manipulator. The algorithm was implemented in a way that excludes the joint variable of the end-effector tool. It is assumed that the tool will be operated separately, as several different tools are available for the Reach Alpha 5.

The first step to implementing potential fields is to define the attractive and repulsive fields acting on the manipulator. These fields were defined as recommended in the textbook ”Robot Modeling And Control[22]”. As discussed in section 2.3.1, there is one attractive and one repulsive field acting each individual joint.

3.2.1 Defining attractive fields

Similarly to the previous version of the program, the attractive fields were defined as a combination of linear and quadratic fields. The quadratic field is well defined for all configurations, and the linear field avoids excessively large torques when the target configuration is far away in the configuration space. The fields were defined as follows:

$$U_{att,i}(q) = \begin{cases} d\gamma_i \left\| o_i(q) - o_i(q_f) \right\| - \frac{1}{2}\gamma_i d^2 & ; \text{if} \quad \left\| o_i(q) - o_i(q_f) \right\| > d \\ \frac{1}{2}\gamma_i \left\| o_i(q) - o_i(q_f) \right\|^2 & ; \text{if} \quad \left\| o_i(q) - o_i(q_f) \right\| \le d \end{cases} \tag{3.1}$$

$$-\nabla U_{att,i}(q) = \begin{cases} -d\gamma_i \frac{(o_i(q) - o_i(q_f))}{\left\| o_i(q) - o_i(q_f) \right\|} & ; \text{if} \quad \left\| o_i(q) - o_i(q_f) \right\| > d \\ -\gamma_i (o_i(q) - o_i(q_f)) & ; \text{if} \quad \left\| o_i(q) - o_i(q_f) \right\| \le d \end{cases} \tag{3.2}$$

where d is the threshold for when each field should be used, γ_i is the attractive field gain, and $o_i(.)$ is the position of the i^{th} joint for the given configuration q. These gains are set by the user and will decide the relative magnitude between each field. Different paths can be generated by tuning γ to prioritize certain joints over others. Section 4.1 describes a test for how these gains affect the generated path. The linear field used here is a modified version of the field discussed in section

2.3.1. This modification ensures continuity in the bridge between the linear and quadratic fields. The quadratic field remains unchanged.

3.2.2 Defining repulsive fields

For the same reasons as with the attractive fields, quadratic fields are chosen for the repulsive force. These fields are well defined for all configurations q. The fields also generate forces approaching infinity as the manipulator approaches an obstacle, ensuring collision-free motion. The repulsive fields were defined as follows:

$$U_{rep,i}(q) = \begin{cases} \frac{1}{2}\delta_i(\frac{1}{o_d(q)} - \frac{1}{d_0})^2 & ; \text{if} \quad o_d(q) \leq d_0 \\ 0 & ; \text{if} \quad o_d(q) > d_0 \end{cases} \tag{3.3}$$

$$-\nabla U_{rep,i} = \begin{cases} \delta_i(\frac{1}{o_d(q)} - \frac{1}{d_0})\frac{1}{o_d(q)^2}\nabla o_d(q) & ; \text{if} \quad o_d(q) \leq d_0 \\ 0 & ; \text{if} \quad o_d(q) > d_0 \end{cases} \tag{3.4}$$

$$\nabla 0_d(q)|_{x=o_i(q)} = \frac{o_i(q) - b}{||o_i(q) - b||} \tag{3.5}$$

where d_0 is the threshold for where the fields are active, b is the point on the obstacle closest to the manipulator, and o_d is the distance from the i^{th} coordinate frame to the obstacle. The repulsive fields were also modified in a similar manner as the linear attractive fields. This ensures continuity when the manipulator is passes over the repulsive threshold d_0.

3.2.3 Gradient descent algorithm

The gradient descent algorithm was implemented as discussed in section 2.3.1.4. The algorithm runs on a while-loop with added exit conditions for excessive runtimes and path lengths. The success-threshold ϵ was set as 2°for the norm of the full joint vector q. This value was chosen as it was deemed sufficient for the tasks the Reach Alpha 5 is intended to handle, specifically grabbing dead fish. The parameter does not impact the characteristics of the search, only when it is considered successful. The step size was chosen as the constant $\alpha = 0.01$. This constant step size was used as a benchmark for comparisons between gradient descent algorithms with and without the Wolfe method for selecting α^i. The gradient descent algorithm as described here was used as a baseline when running various tests. Each added improvement, like the random step method and Wolfe method, was compared to this baseline when tested. More on these tests in section 4.

3.3 Implementing a floating point approach to the repulsive field

One thing that was discovered in the project preceding this thesis was that the implementation of the repulsive fields was flawed. The fields would generate a force based on the distance between the given joint frame and the closest obstacle. However, this means that only the position of the joints themselves were taken into account for obstacle avoidance. This means that a collision-free path could not be guaranteed, as the links would be able to make contact without being repelled. To avoid this, a *floating repulsive reference point*, or floating point, was introduced to each link. The idea behind these reference points is that they move along each link as the manipulator is in motion, always coinciding with the point on the link closest to the obstacle. To achieve this, one must first find the position of each frame of the manipulator, as well as create models of the links between them. It was assumed that all links of the manipulator were perfectly cylindrical, as more precision was deemed negligible. The one exception is the camera mounted at the top of the arm. This was not accounted for in this project.

First the position of each joint frame o_i was found. This is accomplished through the use of the homogenous transformation matrix T_i^0. As discussed in section 2.1.1.1, this matrix takes the form

$$T_i^0 = \begin{bmatrix} R_i^0 & o_i^0 \\ 0 & 1 \end{bmatrix} \tag{3.6}$$

where o_i can be read directly from the top 3 values in the 4th column. Secondly, a method for locating the point on each link closest to the obstacle was found. When all links are assumed to be cylindrical, the links can be viewed as straight line segments between each joint, offset by the diameter. The problem could then be translated to finding the point on a line segment which is closest to an external point(the point on the obstacle closest to the link). Let o_i and o_{i+1} be the position of joints i and $i+1$. Let L denote the line segment between these joints. The vector equation for the line segment L is

$$L(t) = o_i + tD \tag{3.7}$$

$$D = o_{i-1} - o_i \tag{3.8}$$

where D is the directional vector between o_i and o_{i-1}, and $t \in [0,1]$ is a running parameter. Once the line segment was defined, an orthogonal projection of the obstacle point b was found. This was done using the equation

$$p = o_i + \left[\frac{(b - o_i) \cdot D}{D \cdot D}\right] D \tag{3.9}$$

where t_0 is the portion of the line segment closest to b. p is the intersection point between the line segment L and the obstacle b.

In addition to finding the intersection point p, it was also necessary to find the distance d_b from p to b. First, define $p = L(t_0)$, where t_0 is the parameter t at the intersection point p. It can be seen from equations 3.7 and 3.9 that

$$t_0 = \frac{(b - o_i) \cdot D}{D \cdot D} \tag{3.10}$$

After the running parameter t_0 was found, the distance d_b from intersection point p to obstacle b was defined for 3 cases:

$$d_b = \begin{cases} \|o_i - b\| & t_0 < 0 \\ \|p - b\| & 0 \leq t_0 \leq 1 \\ \|o_{i+1} - b\| & t_0 > 1 \end{cases} \tag{3.11}$$

After both the intersection point p and the distance d_b were found, the repulsive fields defined in equations 3.3 and 3.4 were redefined as

$$U_{rep,i}(q) = \begin{cases} \frac{1}{2}\delta_i(\frac{1}{d_b(q)} - \frac{1}{d_0})^2 & ;\text{ if } \quad d_b(q) \leq d_0 \\ 0 & ;\text{ if } \quad d_b(q) > d_0 \end{cases} \tag{3.12}$$

$$-\nabla U_{rep,i} = \begin{cases} \delta_i(\frac{1}{d_b(q)} - \frac{1}{d_0})\frac{1}{d_b(q)^2}\nabla d_b(q) & ;\text{ if } \quad d_b(q) \leq d_0 \\ 0 & ;\text{ if } \quad d_b(q) > d_0 \end{cases} \tag{3.13}$$

Take note that the floating repulsive reference point approach was mentioned in the theory chapter of the project preceding this thesis. However, it was not implemented in practice, as the path planning algorithm would not run when obstacles were used.

3.4 Implementing the Wolfe conditions to the gradient descent algorithm

As mentioned in section 3.2.3, a gradient descent algorithm using a constant $\alpha = 0.01$ was created. Another variant of the same search algorithm was also created, this one making use of the Wolfe conditions for selecting an appropriate step size. Section 2.3.1.5 describes how the Armijo condition and curvature condition are implemented as a set of inequalities. These inequalities are used as exit conditions for an inner loop in the gradient descent algorithm. Said loop will initialize with an initial value for α_0 and insert the resulting set point into the Wolfe conditions. If the Armijo condition is not fulfilled, $\alpha_{k+1} = \alpha_k/2$ and if the curvature condition is not fulfilled, $\alpha_{k+1} = \alpha_k \times 1.5$.

The Armijo condition was implemented first. This inequality relies on the value of the objective function for the following step, the objective function at current step and the gradient of the objective function at current step. From section 2.3.1.4 one can see that the objective function of the search is $\|q - q_f\|$. While this is the term the algorithm is trying to minimize, field strength U is directly correlated with the term, meaning it can be used as a substitute for the objective function in this inequality. The only difference between using $\|q - q_f\|$ and U is the practicality of reading the values in the program. One major advantage of using U, however, is that the gradient is clearly defined in equations 2.22, 2.20, and3.13 In addition, the directional derivative $\nabla f_k^T p_k$ becomes the torque vector τ_k. The Armijo condition was then implemented as the inequality

$$U(\alpha_k) \leq U(\alpha_{k-1}) + c_1 \alpha_k \tau \tag{3.14}$$

Secondly, the curvature condition was implemented. This condition relies on the directional derivative at the following step, and the directional derivative at current step. Directional derivative was found to be τ_k, so the curvature condition was implemented as

$$\tau_k \geq c_2 \tau_{k-1} \tag{3.15}$$

Pseudocode for the new gradient descent algorithm:

```
**Gradient descent algorithm w/Wolfe**
$q^0 \leftarrow q_s, i \leftarrow 0$
while $\left\|q^i - q_f\right\| > \epsilon$ do
    while $\alpha^i$ is unsuitable do
        $q^{i+1} \leftarrow q^i + \alpha^i \frac{\tau(q^i)}{\|\tau(q^i)\|}$
        if NOT Armijo then
            $\alpha^i \leftarrow \alpha^i/2$
        else if NOT curvature then
            $\alpha^i \leftarrow \alpha^i \times 1.5$
        end if
    end while
    $i \leftarrow i + 1$
end while
return $< q^0, q^1, \ldots, q^i >$
```

Finally, values had to be decided for the parameters $0 < c_1 < c_2 < 1$, and the initial α_0. These values were chosen based on recommendations given in the textbook "Numerical Optimization[9]

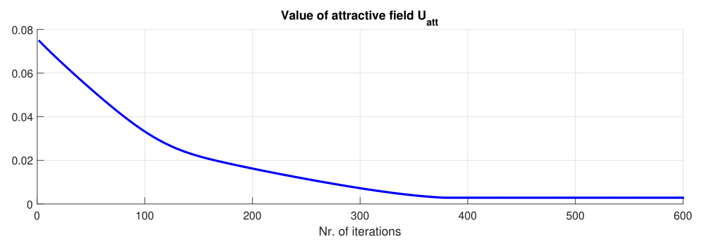


Figure 3.3: The value of potential fields U during a gradient descent search with constant $\alpha = 0.01$

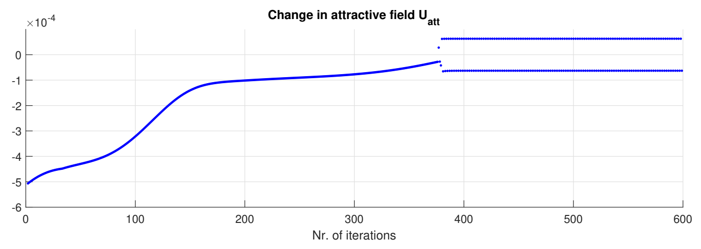


Figure 3.4: The change in potential fields U during a gradient descent search with constant $\alpha = 0.01$

$$c1 = 10e^{-4} \tag{3.16}$$

$$c2 = 0.9 \tag{3.17}$$

$$\alpha_0 = 1 \tag{3.18}$$

3.5 Implementing the random step method to the gradient descent algorithm

The next addition to the new path planning algorithm was to utilize the randomized walk method for escaping local minima. Following what was discussed in section 2.3.1.6, the method relies on the ability for the algorithm to recognize when it is stuck in a local minima. As stated in section 2.3.1.6, recognizing a local minima is not trivial. The textbook suggests comparing the change in joint positions q over multiple successive steps. Let ϵ_m be a small positive value. One can assume local minima when $\|q^i - q^{i+1} < \epsilon_m\|$, $\|q^{i+1} - q^{i+2} < \epsilon_m\|$, $\|q^{i+2} - q^{i+3} < \epsilon_m\|$ for however many terms one deems adequate. A simulation was run using the baseline gradient descent algorithm, as well as the gradient descent algorithm w/Wolfe conditions. The simulation was run to purposefully get stuck in a local minima to analyze the resulting data. Figure 3.3 shows the value of the norm of the field U^i, while figure 3.4 shows the change in between U^{i-1} and U^i for a gradient descent search using $\alpha = 0.01$.

One can clearly see from these figures that a local minima has led to alternating positive and negative change in the field U. This would suggest that locating a positive change in U is an indicator of a local minima. However, seeing as how the step size was fixed, one cannot guarantee that this phenomenon is not caused by the algorithms inability to take smaller steps. Figures 3.5 and 3.6 show the result of the same test when using Wolfe steps.

Take note that the differences in number of steps are caused by the relative speed at which the two algorithms reached the local minima. In this case the graph seems to approach zero over time. However, this cannot be the case as the algorithm did get stuck in a local minima. Figure 3.7 shows the same graph as figure 3.6, but zoomed in. The effect noted in figure 3.4 is still present, and the assumption that a positive change in U indicates a local minima was therefore used when implementing the randomized walk.

Once the indicator for local minima was decided upon, the randomized step function was implemented. The method used was the same as presented in section 2.3.1.6. This lead to the updated

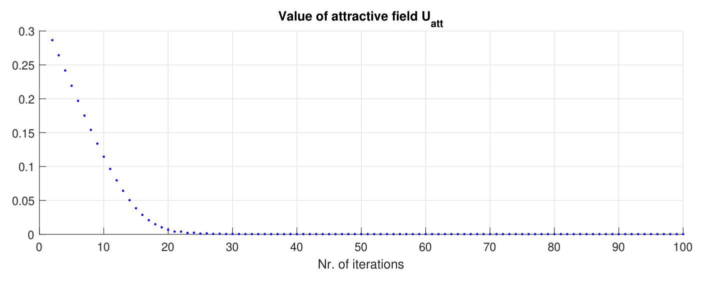


Figure 3.5: The value of potential fields U during a gradient descent search with Wolfe steps

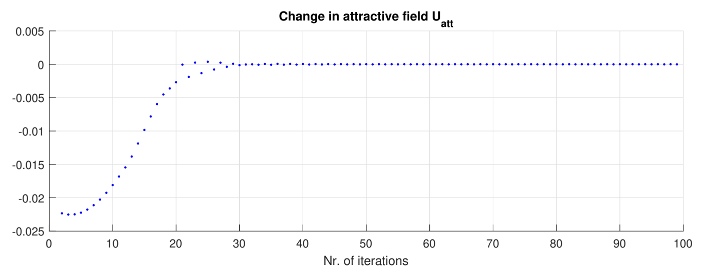


Figure 3.6: The change in potential fields U during a gradient descent search with Wolfe steps

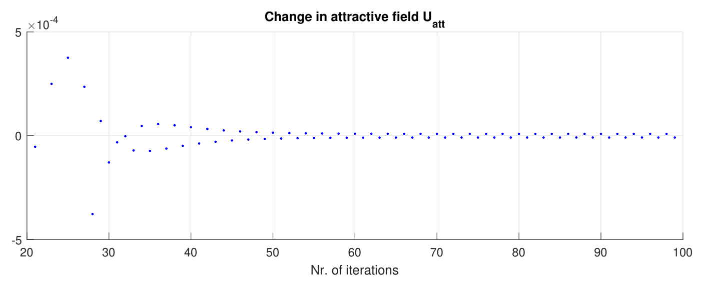


Figure 3.7: The phenomenon is still present

gradient descent algorithm:

```
**Gradient descent algorithm w/Wolfe & randomized step**
$q^0 \leftarrow q_s, i \leftarrow 0$
while $\left\|q^i - q_f\right\| > \epsilon$ do
    while $\alpha^i$ is unsuitable do
        $q^{i+1} \leftarrow q^i + \alpha^i \frac{\tau(q^i)}{\|\tau(q^i)\|}$
        if NOT Armijo then
            $\alpha^i \leftarrow \alpha^i/2$
        else if NOT curvature then
            $\alpha^i \leftarrow \alpha^i \times 1.5$
        end if
    end while
    $i \leftarrow i + 1$
    if stuck in local minima then
        random walk to $q^k$
        $q^{i+1} \leftarrow q^k$
    end if
end while
return $< q^0, q^1, \ldots, q^i >$
```

Lastly, the randomized step had to be defined by parameters q_v and s, being the size of the random steps and number of random steps respectively. These parameters were initially set as

$$q_v = 0.3 \tag{3.19}$$

$$s = 4 \tag{3.20}$$

A test was conducted to evaluate these parameters. More on this in section 4. It was also decided that the algorithm would only recognize a local minima if a positive change in U was detected 3 times. The purpose was to exclude potential outliers in the data.

Chapter 4

Tests

A series of experiments was conducted to verify the methods implemented in chapter 3. These experiments were conducted by comparing the path planning algorithm with and without given components included. Simulation were run for arbitrarily chosen initial and final conditions q_s and q_f, given that a feasible path was achievable. Generated paths were then given to the Reach Alpha 5 to test the practical effects when applying the generated path. A gradient descent algorithm with constant step size $\alpha = 0.01$ was used when testing the various components of the path planner. This was done as such a small step size would ensure a stable search, albeit an inefficient one. With this step size the program was expected to run a minimum of 100 iterations, with a high likelihood of significantly longer paths depending on choices for several adjustable parameters in the algorithm. The algorithm was terminated at 1000 steps, as this was seen as an indicator that the search would not be able to finish at q_f. The value of 1000 steps was found through experimentation, where no successful search ever finished after this point. The threshold ϵ, which indicates that the configuration q^i is sufficiently close to q_f, was set as 2°for the norm $\left\|q^i - q_f\right\|$, as this precision was deemed sufficient for the tasks the Reach Alpha 5 will execute. Each experiment outlined in this section is presented the way it was originally envisioned. During testing, certain observations were made which lead to alterations and extensions of some of the experiments. These observations, and subsequent changes to the experiments, are presented in the results section of the thesis. These results are discussed further in section 6

4.1 Test attractive potential fields gain

The first experiment was meant to explore the effect of changing the relative magnitudes of the attractive potential field gains γ_i. Changes in these parameters will alter the generated path, as the attractive fields will affect each joint differently. The magnitude of these parameters are relative, meaning the attractive fields will increase or decrease relative to each other proportionally to the magnitudes in γ. An excessively large value in γ_i will not cause the gradient descent to take excessively large steps, at a normalized torque vector is used of the step.

The robotics textbook[22] recommends weighing one joint higher than the others. This will lead to what is referred to as a "follow the leader" motion characteristic. The joint with the large weight will be prioritized by the algorithm, causing the frame of said joint to reach its destination first. The other frames will then follow suit.

The experiment was conducted for 5 different cases. Case 1 used an adaptive gain vector $\gamma = [1 \quad 1 \quad 1 \quad 1]$ while the other cases used a weight of $\gamma_i = 10$ for each joint frame respectively. No obstacles were introduced to the workspace. A constant step size $\alpha = 0.01$ was used. The initial and final configuration were set as $q_s = [60° \quad 100° \quad 60° \quad 0°]^T$ and $q_f = [220° \quad 110° \quad 115° \quad 0°]^T$.

The experiment was expected to produce successful paths for all cases. Regardless of which joint is prioritized, the negative gradient of the potential fields will point towards a minima. The performances when weighing each joint higher were expected to reflect the relative weights. In each case, the weighted joint should reach the target position before the others. Results of the experiment are presented in section 5.1.

4.2 Test random walk

The next experiment was done to verify the random walk method and its ability to help the gradient descent algorithm escape local minima. The random walk is parameterized by the number of steps and the size of each step taken. Increasing these parameters will allow the random walk to move the manipulator further, increasing the likelihood of escaping a local minima. One must take care not to choose too large values for these parameters, as it will lead to unexpected movements that might compromise the obstacle avoidance element of the path planner. Large random walks when close to obstacles can therefore be very dangerous.

This experiment was originally meant to be conducted using obstacles to created a local minima. However, results from the attractive gain experiment, presented in section 5.1, showed that the algorithm was prone to getting stuck in local minima even when on obstacles were present.

The experiment was conducted by simulating the same two sets of q_s and q_f presented in the attractive gains test, but this time the random walk will be activated whenever a local minima is encountered. The parameters used for each case are

$$
\begin{aligned}
q_s^1 &= [60^\circ \quad 100^\circ \quad 60^\circ \quad 0^\circ]^T & q_s^2 &= [150^\circ \quad 90^\circ \quad 80^\circ \quad 0^\circ]^T \\
q_f^1 &= [220^\circ \quad 110^\circ \quad 115^\circ \quad 0^\circ]^T & q_f^2 &= [10^\circ \quad 190^\circ \quad 100^\circ \quad 0^\circ]^T \\
\gamma^1 &= [1 \quad 1 \quad 1 \quad 1] & \gamma^2 &= [1 \quad 1 \quad 1 \quad 1]
\end{aligned}
$$

with the randomized walk being parameterized by walk size $q_v = 1^\circ$ and number of steps $s = 4$. A local minima would be recognized once the change in potential fields strength U was positive 3 times.

The experiment was expected to produce successful paths for both cases. The randomized walk was selected to be large to ensure that the algorithm would escape any local minima in the gradient descent search. The results of the experiment are presented in section 5.2

4.3 Test Wolfe

This experiment was conducted to evaluate the effectiveness of the Wolfe method for selecting step size α^i in the gradient descent search.

The experiment was conducted using joint configurations $q_s = [5^\circ \quad 90^\circ \quad 125^\circ \quad 0^\circ]^T$ and $q_f = [5^\circ \quad 195^\circ \quad 66^\circ \quad 0^\circ]^T$, and an initial step size $\alpha_0 = 1$ with $c_1 = 10e^{-4}$ and $c_2 = 0.9$. The joint configurations were chosen after being verified that a path could be successfully generated by the algorithm using a constant $\alpha = 0.01$.

The experiment was expected to produce a successful path from q_s to q_f. The path was also expected to be generated in significantly fewer iterations than the path planned by the baseline algorithm. This algorithm required 215 iterations to finish. Figure 4.1 shows the reference path. The results of the experiment are presented in section 5.3.

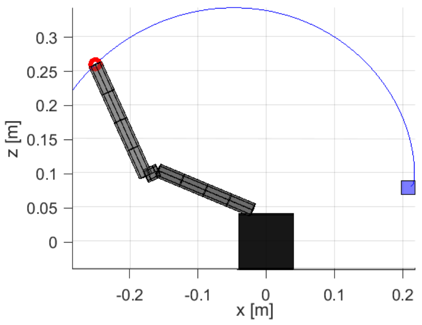


Figure 4.1: Reference path for the gradient descent algorithm using Wolfe conditions.

4.4 Test obstacle avoidance

This experiment was conducted to evaluate whether or not the algorithm was able to avoid obstacles in the workspace using repulsive potential fields. The experiment was to make use of random walks, Wolfe conditions and floating potential fields reference points to see how these elements affected algorithm when planning around an obstacle. The results of the Wolfe condition experiment in section 5.3 revealed that the method was not properly implemented, causing α^i to not update. The Wolfe conditions were therefore excluded from this experiment.

A point obstacle was introduced into the workspace of the manipulator, and the path planner was tasked with finding a collision-free path from q_s to q_f. As obstacles can cause local minima to appear in the gradient search, the experiment will be ran both with and without randomized walks.

The initial and final joint configurations were set as

$$q_s = [60^\circ \quad 100^\circ \quad 60^\circ \quad 0^\circ]^T$$
$$q_f = [220^\circ \quad 110^\circ \quad 115^\circ \quad 0^\circ]^T$$

the same path used in the Wolfe experiment. All attractive and repulsive gains were set to 1. The cutoff distance for the repulsive fields was set to 8cm. The random walk was defined by walk size $q_v = 1$ and number of walks $s = 4$. The point obstacle was placed at coordinates $b = [0 \quad -2e-2 \quad 29.3e-2]^T$. The cutoff threshold for terminating the search was increased to 3000 due to the added complexity of the problem.

After having conducted the previous experiments, the expectations for this one were less optimistic. The algorithm was expected to successfully avoid the obstacle, but would likely get stuck in a local minima before completing the path. The randomized walk was expected to not be able to escape a local minima.

4.5 Test practical application

The final experiment was to move the Reach Alpha 5 along the paths generated by the potential fields path planner. The test of using Wolfe conditions to decide a step size α^i for each iteration of the algorithm did not yield satisfactory results when tested. See section 5.3. Because of this, all paths generated by the path planning algorithm use a constant step size $\alpha = 0.01$. This results in paths with a great amount of set points close together. These set points can be given as a series of joint references to the integrated PID-controllers in the manipulator. Once the manipulator is close to the given reference, the next set point is used as the new reference. This forces the manipulator to follow the generated path of set points.

The planned paths were exported from Matlab to python, where they were used as a series of joint references. The joint values of the Reach Alpha 5 were recorded and imported to Matlab for data analysis and plotting.

Chapter 5

Results

5.1 Attractive field gains

This section will present the results of the simulations that were done for the experiment described in section 4.1.

Simulation 1

The first simulation was executed using an attractive gain of $\gamma = [1 \quad 1 \quad 1 \quad 1]$. The simulation ran for 1000 steps and was unable to reach the desired destination. Figure 5.1 shows a 3D-model of the manipulator with a trace of the end-effector position.

Figure 5.2 shows the manipulator joint values throughout the full simulation, as well as the distance between the end-effector and its target destination. From the figure it can be seen that a local minima was reached almost 400 iterations into the search. None of the joint values converge over time.

Figure 5.3 plots the distance between each joint frame and their target positions throughout the simulation. Joint 1 is omitted from the plot at it barely moves regardless of motion. None of the joint frames converge to their target destinations over time.

Simulation 2

The second simulation was executed using an attractive gain of $\gamma = [10 \quad 1 \quad 1 \quad 1]$. Like simulation 1, the second simulation ran for 1000 steps and was unable to reach the desired destination. Figure

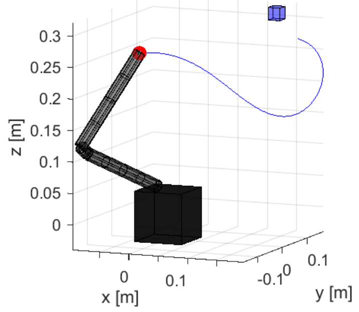


Figure 5.1: 3D-model of the Reach Alpha 5 with trace of the end effector when $\gamma = [1 \quad 1 \quad 1 \quad 1]$.

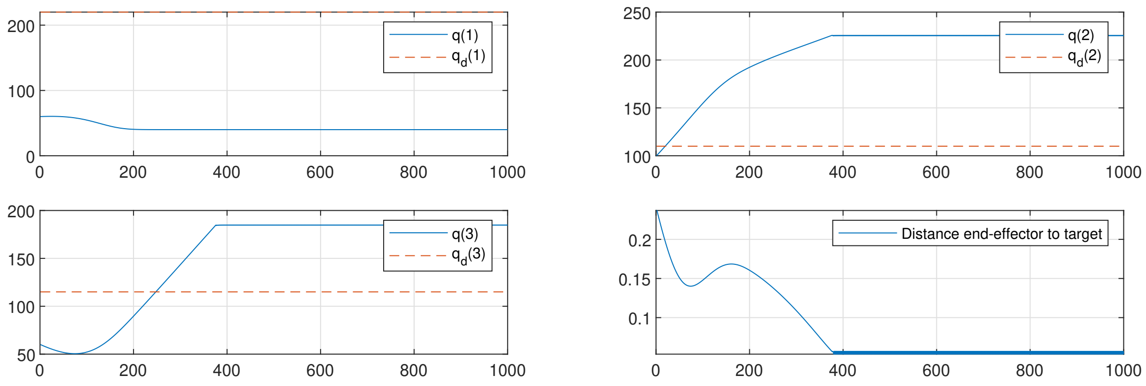


Figure 5.2: Joint variables $q(i)$ compared to the given reference $q_f(i)$ for $\gamma = [1 \quad 1 \quad 1 \quad 1]$.

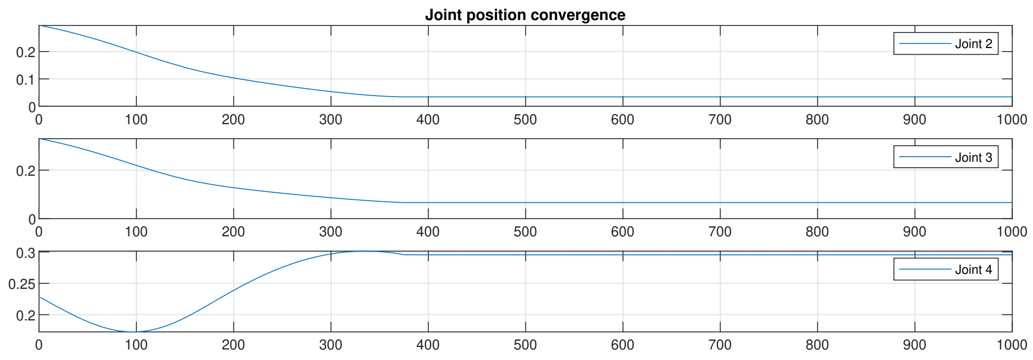


Figure 5.3: Distance between joint frame and target position for $\gamma = [1 \quad 1 \quad 1 \quad 1]$.

5.1 shows a 3D-model of the manipulator with a trace of the end-effector position. The path is visually almost identical with the path in simulation 1. This is to be expected as joint 1 will barely move, meaning that a weighted $\gamma_1 = 10$ will not change much, as the position of joint 1 is already approximately at the target position.

Figure 5.5 shows the manipulator joint values throughout the full simulation, as well as the distance between the end-effector and its target destination. As with simulation 1, it can be seen from the figure that a local minima was reached almost 400 iterations into the search. None of the joint values converge over time.

Figure 5.6 plots the distance between each joint frame and their target positions throughout the simulation. None of the joint frames converge to their target destinations over time.

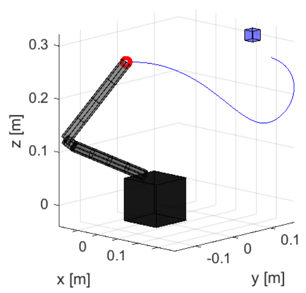


Figure 5.4: 3D-model of the Reach Alpha 5 with trace of the end effector when $\gamma = [10 \quad 1 \quad 1 \quad 1]$.

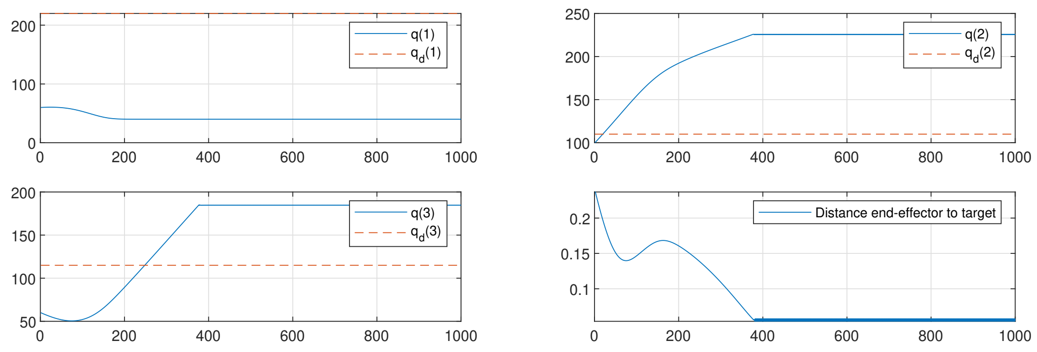


Figure 5.5: Joint variables $q(i)$ compared to the given reference $q_f(i)$ for $\gamma = [10 \quad 1 \quad 1 \quad 1]$.

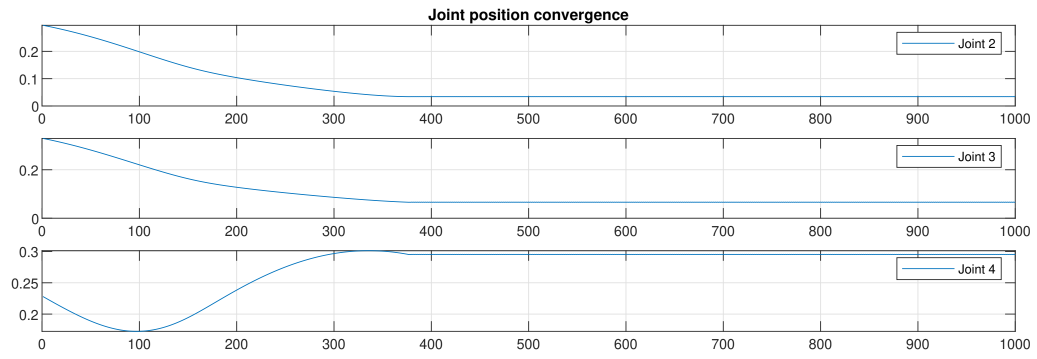


Figure 5.6: Distance between joint frame and target position for $\gamma = [10 \quad 1 \quad 1 \quad 1]$.

Simulation 3

The second simulation was executed using an attractive gain of $\gamma = [1 \quad 10 \quad 1 \quad 1]$. This set of attractive gains prioritizes the position of the second joint. Like simulation 1 and 2, the third simulation ran for 1000 steps and was unable to reach the desired destination. Figure 5.7 shows a 3D-model of the manipulator with a trace of the end-effector position. From the trace of the end-effector one can see that the change in attractive gain caused a change in manipulator motion.

Figure 5.8 shows the manipulator joint values throughout the full simulation, as well as the distance between the end-effector and its target destination. None of the joint values converge over time.

Figure 5.9 plots the distance between each joint frame and their target positions throughout the simulation. None of the joint frames converge to their target destinations over time. One can, however, see a change in how quickly the third joint approaches its destination for the first 300 steps of the simulation. The new weight caused joint 3 to converge more quickly. The simulation still ended up stuck in a local minima.

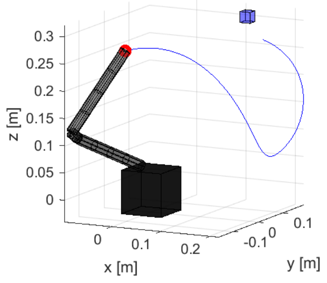


Figure 5.7: 3D-model of the Reach Alpha 5 with trace of the end effector when $\gamma = [1 \quad 10 \quad 1 \quad 1]$.

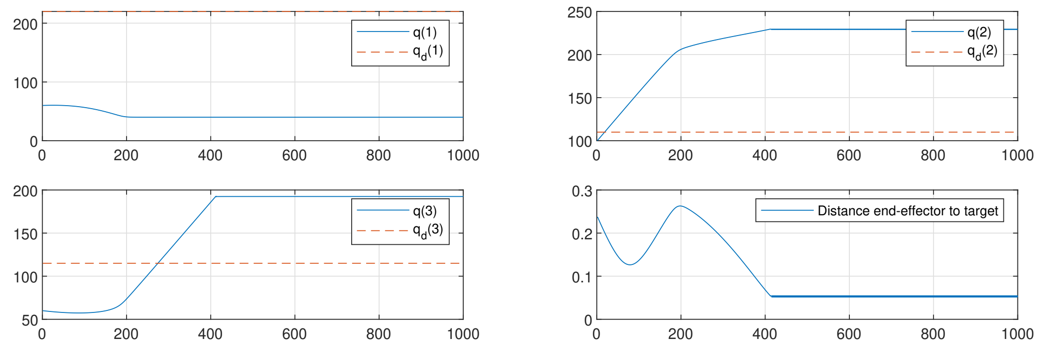


Figure 5.8: Joint variables $q(i)$ compared to the given reference $q_f(i)$ for $\gamma = [1 \quad 10 \quad 1 \quad 1]$.

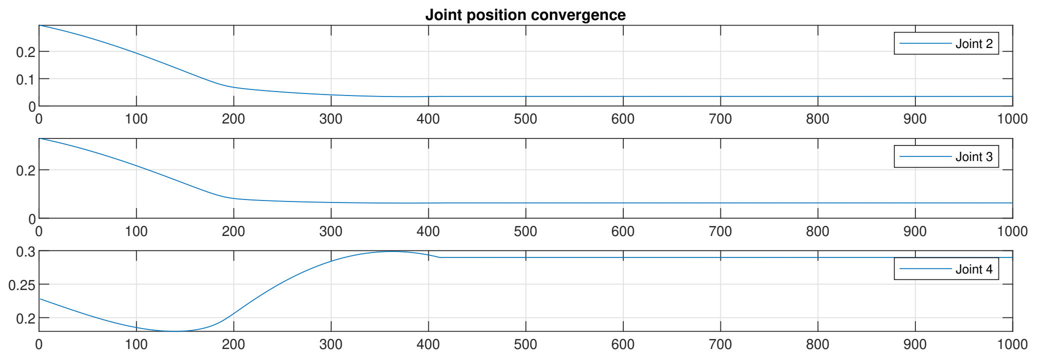


Figure 5.9: Distance between joint frame and target position for $\gamma = [1 \quad 10 \quad 1 \quad 1]$.

Simulation 4

The fourth simulation was executed using attractive gains of $\gamma = [1 \quad 1 \quad 10 \quad 1]$. As with previous simulations, it terminated after 1000 iterations, having been caught in a local minima. Figure 5.10 shows the trace of the end-effector for simulation 4. This case ends up with the end-effector further away than any of the previous cases.

Figure 5.11 shows the manipulator joint values throughout the full simulation, as well as the distance between the end-effector and its target destination. Despite how the end-effector gets stuck further away from the goal, one can see that the local minima is encountered at roughly the same iteration of the search. None of the joint values converge over time.

Figure 5.12 plots the distance between each joint frame and their target positions throughout the simulation. None of the joint frames converge to their target destinations over time.

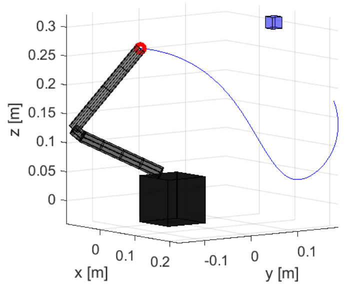


Figure 5.10: 3D-model of the Reach Alpha 5 with trace of the end effector when $\gamma = [1 \quad 1 \quad 10 \quad 1]$.

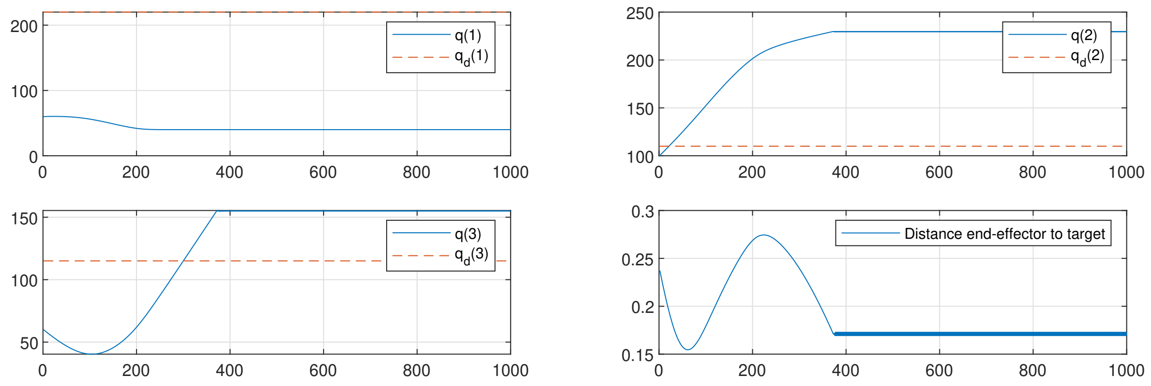


Figure 5.11: Joint variables $q(i)$ compared to the given reference $q_f(i)$ for $\gamma = [1 \quad 1 \quad 10 \quad 1]$.

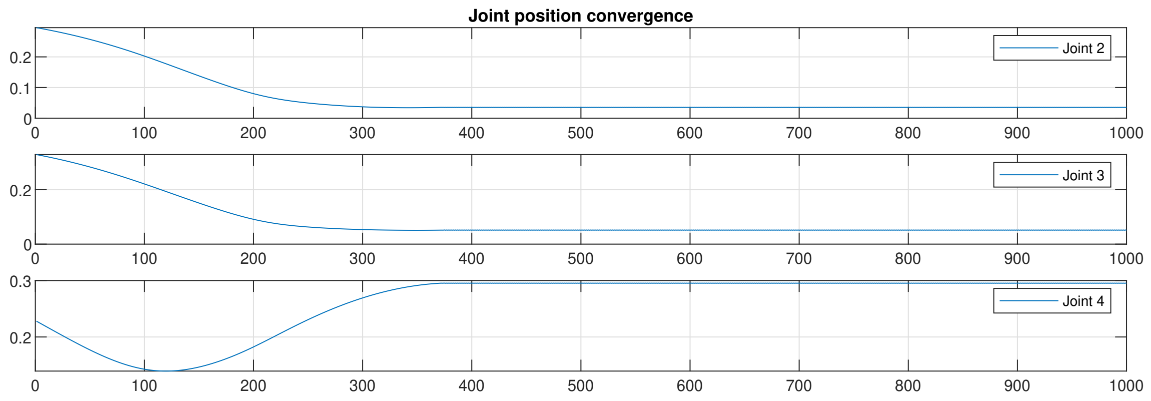


Figure 5.12: Distance between joint frame and target position for $\gamma = [1 \quad 10 \quad 1 \quad 1]$.

Simulation 5

The fifth and final simulation was executed using attractive gains of $\gamma = [1 \quad 1 \quad 1 \quad 10]$. This simulation also terminated after 1000 iterations, having been caught in a local minima. Figure 5.13 shows the trace of the end-effector for simulation 5. The visual representation shows that the end-effector got much closer to the target for this set of attractive field gains. This was expected, as joint 4, and by extension the end-effector, was weighted higher that the other joints.

Figure 5.14 shows the manipulator joint values throughout the full simulation, as well as the distance between the end-effector and its target destination. Although the end-effector approached its desired position, it can be seen from the figure that no joints converged towards the reference. Take note that the reference comes in the form of a joint configuration, not an end-effector position

Figure 5.12 plots the distance between each joint frame and their target positions throughout the

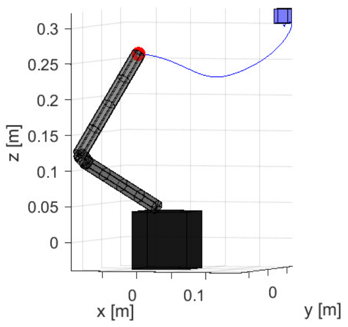


Figure 5.13: 3D-model of the Reach Alpha 5 with trace of the end effector when $\gamma = [1 \quad 1 \quad 1 \quad 10]$.

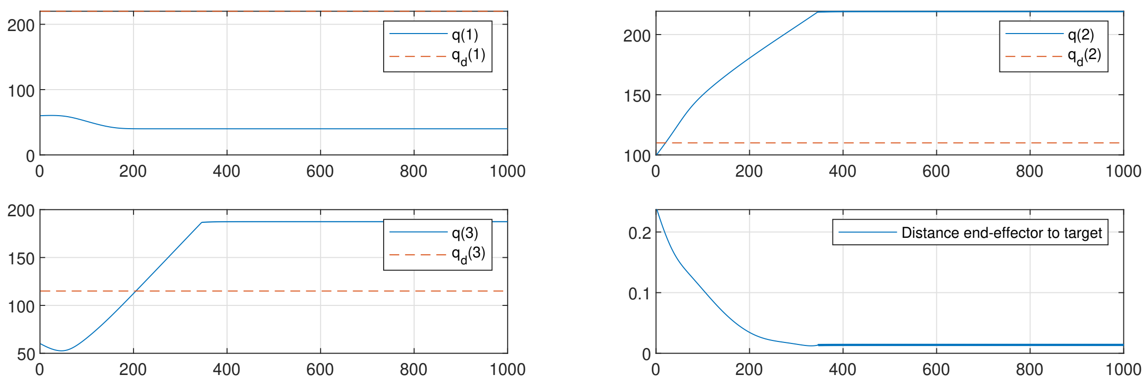


Figure 5.14: Joint variables $q(i)$ compared to the given reference $q_f(i)$ for $\gamma = [1 \quad 1 \quad 1 \quad 10]$.

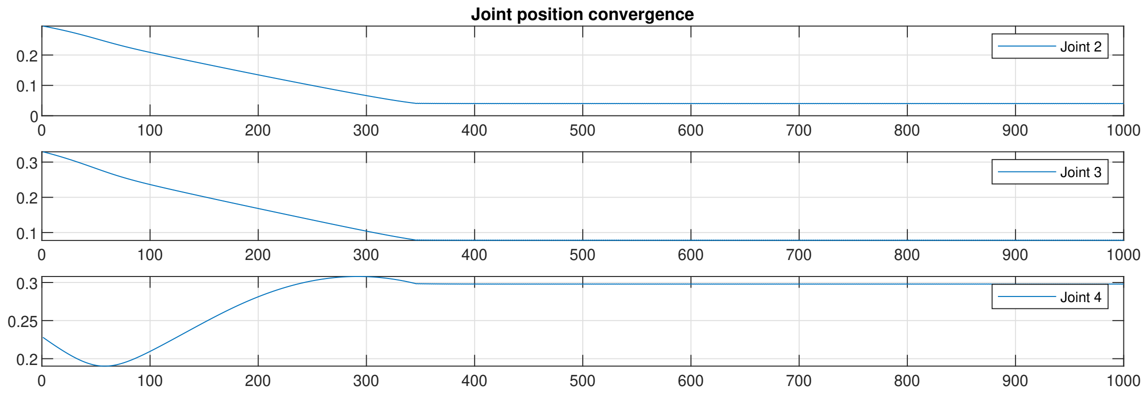


Figure 5.15: Distance between joint frame and target position for $\gamma = [1 \quad 1 \quad 10 \quad 1]$.

simulation. None of the joint frames converge to their target destinations over time. Joint 2 and 3 were slower to move in this case, which follows from the theory. The fourth joint is prioritized and moves faster.

Observations

None of the attractive gains were able to generate a successful path from q_s to q_f. In every case, a local minima was encountered before the 400th iteration of the search. The joint values q^i did not converge towards the given reference, even before getting stuck in the local minima. Joint 1 is noteworthy, as it converges efficiently to $q_f(1) + 180°$ in every case.

Due to the consistent failure to escape the local minima, other paths were tested to see if the initial configuration and reference q_s and q_f was the cause of the algorithms inability to converge. The results from one of these alternative paths will be presented here. This path used an initial configuration $q_s = [150° \quad 90° \quad 80° \quad 0°]$ and reference configuration $q_f = [10° \quad 190° \quad 100° \quad 0°]$.

Simulation 1 secondary path

The first simulation of the secondary path was executed using neutral weights $\gamma = [1n \quad 1 \quad 1 \quad 1]$. Like with the original path, the simulation terminated after 1000 steps and was unable to reach the desired destination. Figure 5.16 shows the trace of the end-effector. The new path did not change the result and the gradient descent algorithm got stuck in a local minima.

Figure 5.17 shows the manipulator joint values throughout the full simulation, as well as the distance between the end-effector and its target destination. Joint 1 still converges towards $q_f(1) + 180°$, and no joint converges toward the given reference.

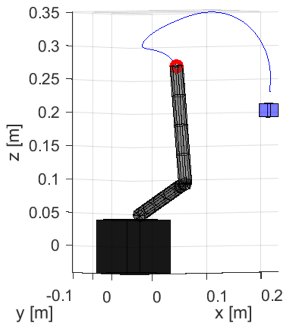


Figure 5.16: 3D-model of the Reach Alpha 5 with trace of the end effector when $\gamma = [1 \quad 1 \quad 1 \quad 1]$using alternative start and stop conditions.

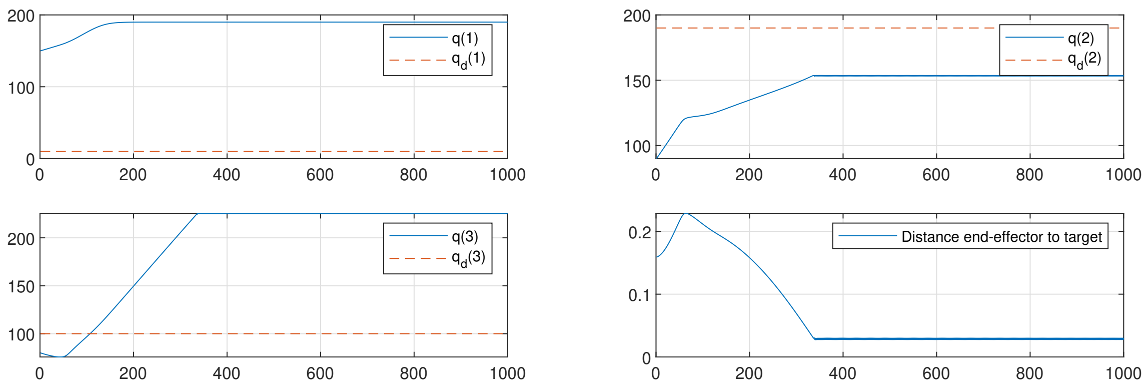


Figure 5.17: Joint variables $q(i)$ compared to the given reference $q_f(i)$ for $\gamma = [1 \quad 1 \quad 1 \quad 1]$.

Figure 5.18 plots the distance between each joint frame and their target positions throughout the simulation. None of the joint frames converge to their target destinations over time. The data for this secondary path has the same characteristics as the data from the original path.

Simulations 3 and 4 secondary path

The second simulation with weight on joint 1 lead to a similar result as all previous simulations. However, simulations 3 and 4 for the secondary path yielded different results. Figure 5.19 shows the trace of the end effector for simulation 3. The simulation terminated successfully after 373 steps.The figure shows that the end-effector reached the desired position at this time.

Figure 5.20 shows the manipulator joint values throughout the full simulation, as well as the

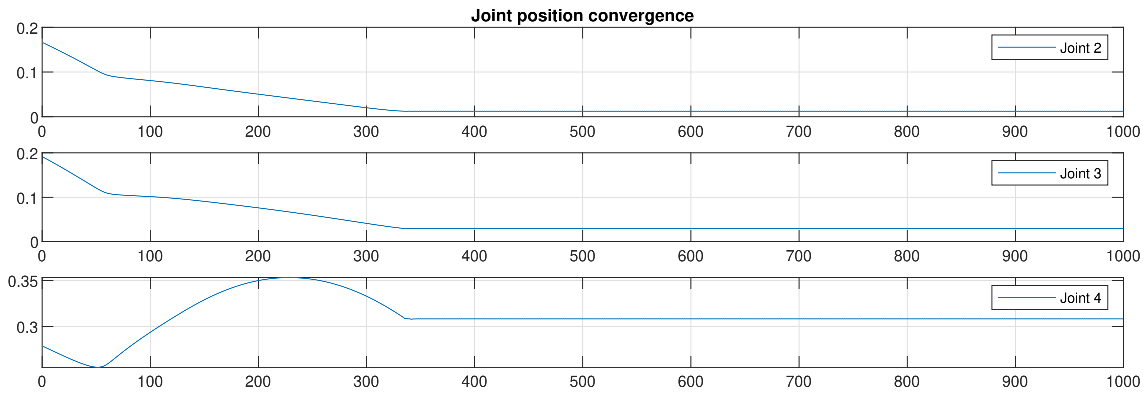


Figure 5.18: Distance between joint frame and target position for $\gamma = [1 \quad 1 \quad 1 \quad 10]$.

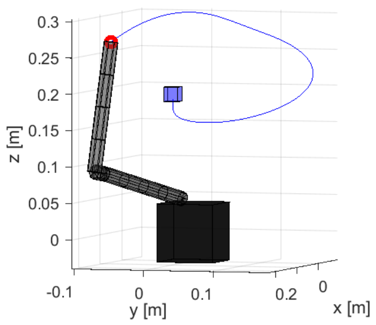


Figure 5.19: 3D-model of the Reach Alpha 5 with trace of the end effector when $\gamma = [1 \quad 10 \quad 1 \quad 1]$using alternative start and stop conditions.

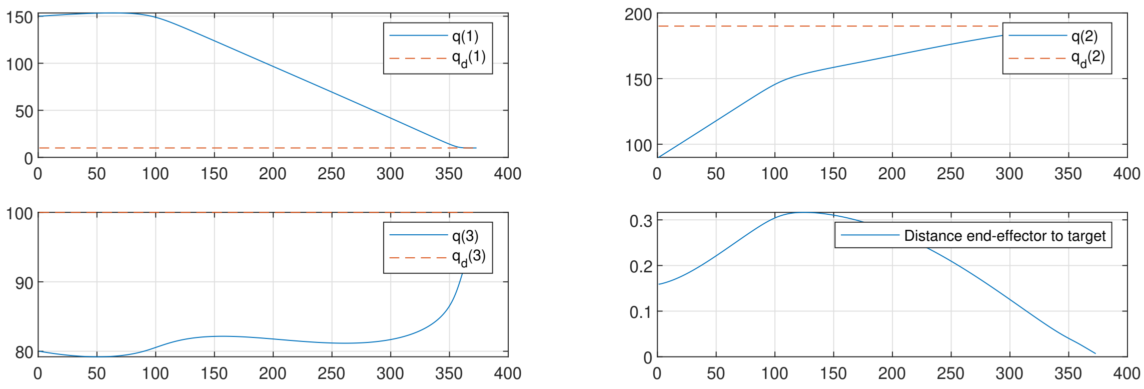


Figure 5.20: Joint variables $q(i)$ compared to the given reference $q_f(i)$ for $\gamma = [1 \quad 10 \quad 1 \quad 1]$.

distance between the end-effector and its target destination. Whereas no joints converged to the given reference in any previous simulation, all joints converged in this case.

Figure 5.21 plots the distance between each joint frame and their target positions throughout the simulation. Joint 2 and 3 converge to the target position, while joint 4 does not. This was later found out to be a bug when plotting the data. One can be sure that this is a visual bug, as the position of joint 4 must coincide with the target position by necessity when joint values q have been shown to converge to q_f.

Figure 5.22 shows the trace of the end effector for simulation 4. The simulation terminated successfully after 411 steps.The figure shows that the end-effector reached the desired position at this time.

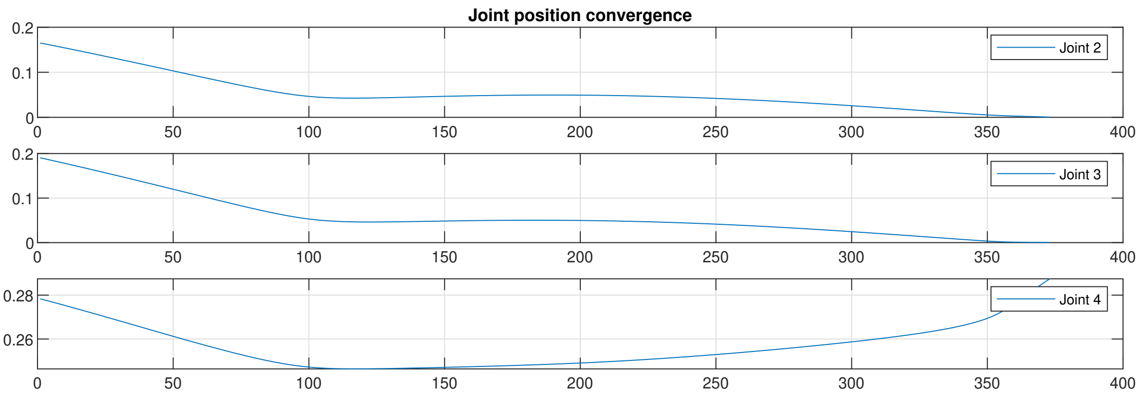


Figure 5.21: Distance between joint frame and target position for $\gamma = [1 \quad 1 \quad 1 \quad 1]$.

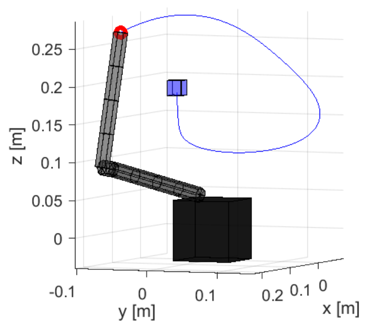


Figure 5.22: 3D-model of the Reach Alpha 5 with trace of the end effector when $\gamma = [1 \quad 1 \quad 10 \quad 1]$using alternative start and stop conditions.

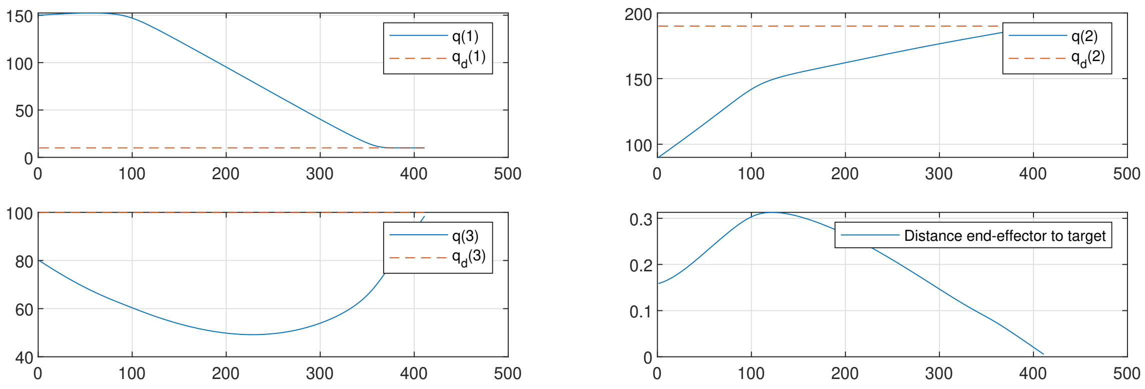


Figure 5.23: Joint variables $q(i)$ compared to the given reference $q_f(i)$ for $\gamma = [1 \quad 1 \quad 10 \quad 1]$.

Figure 5.23 shows the manipulator joint values throughout the full simulation, as well as the distance between the end-effector and its target destination. As with simulation 3, all joints converged.

Figure 5.24 plots the distance between each joint frame and their target positions throughout the simulation. All joint positions converge towards the desired target position. Once again the graph for joint 4 is bugged. The value was later validated as converging. All Matlab workspace data from these simulation was saved and will be submitted alongside this thesis.

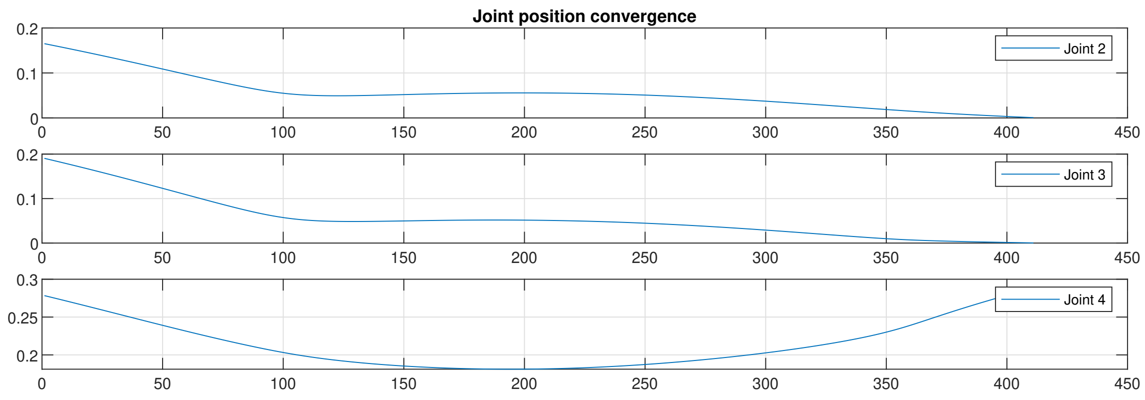


Figure 5.24: Distance between joint frame and target position for $\gamma = [1 \quad 10 \quad 1 \quad 1]$.

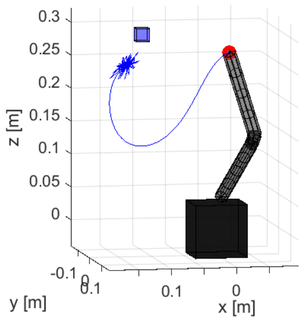


Figure 5.25: 3D-model of the Reach Alpha 5 with trace of the end effector when $q_v = 1°$ and $s = 4$.

Simulation 5 secondary path

Simulation 5 terminated after 1000 iterations. Like with most of the other simulations, the algorithm was caught in a local minima.

Observations

The secondary path presented here was one of many that were used in this experiment. It was included to show that results varied drastically for different q_s and q_f. While a large weight on joint 2 or 3 lead to successful path planning in this case, it was not a general trend. The algorithm would often get caught in a local minima, with no clear system based on the attractive gains

5.2 Random step

This section will present the results of the experiment outlined in section 4.2.

Simulation 1

The first simulation was executed with random walk parameters $q_v = 1°$ and $s = 4$. The initial an reference configurations were the same as in the attractive gains experiment. The simulation terminated after 1000 iterations. The algorithm was not able to generate a successful path. Figure 5.25 shows the trace of the end-effector during simulation. One can see the effect of the random walk when the algorithm encountered a local minima. Although the effects of the randomized walk move the end-effector nearly 5cm, the algorithm moves back into the local minima.

Figure 5.26 plots the manipulator joint values relative to their given reference. The randomized walk was not able to circumvent the local minima.

Simulation 2

The second simulation was executed with the same random walk parameters $q_v = 1°$ and $s = 4$ as simulation 1. The initial configuration and reference configuration were the same as in the secondary path from the attractive gains experiment. The simulation terminated after 1000 iterations. The algorithm was not able to generate a successful path. Figure 5.27 shows the trace of the end-effector during simulation. The new q_s and q_f did not change the outcome. The algorithm got trapped in a local minima and was not able to exit.

Figure 5.28 plots the manipulator joint values relative to their given reference for simulation 2.

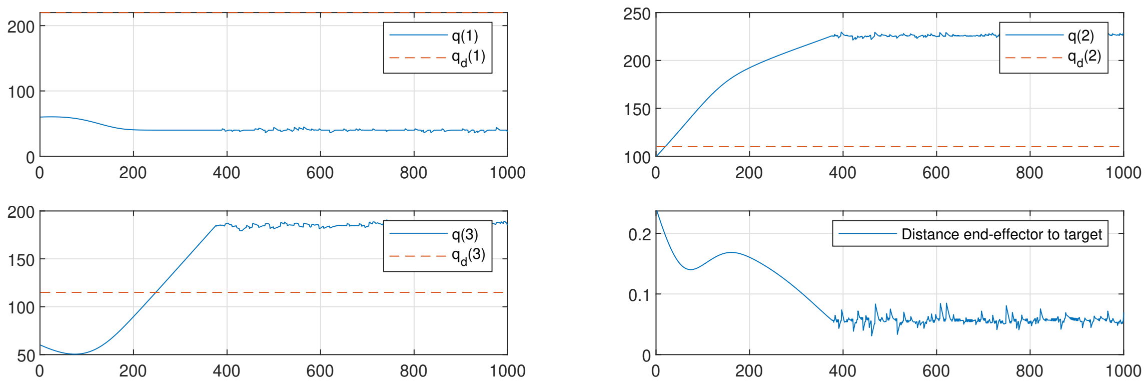


Figure 5.26: Joint variables $q(i)$ compared to the given reference $q_f(i)$ for $q_v = 1$ and $s = 4$.

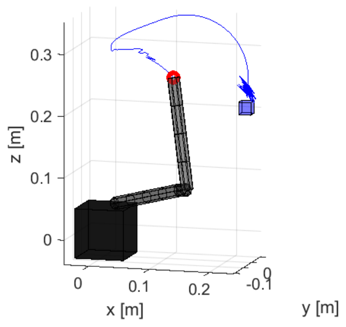


Figure 5.27: 3D-model of the Reach Alpha 5 with trace of the end effector when $q_v = 1°$ and $s = 4$ for secondary path.

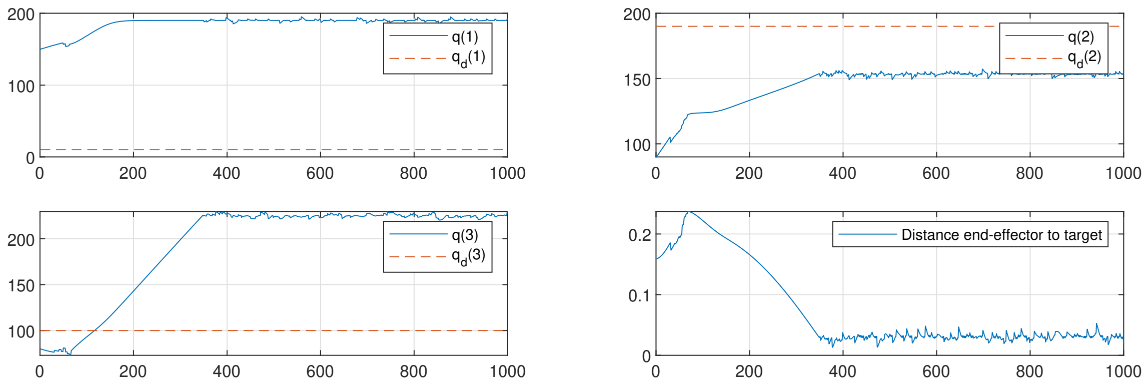


Figure 5.28: Joint variables $q(i)$ compared to the given reference $q_f(i)$ when $q_v = 1$ and $s = 4$ for secondary path.

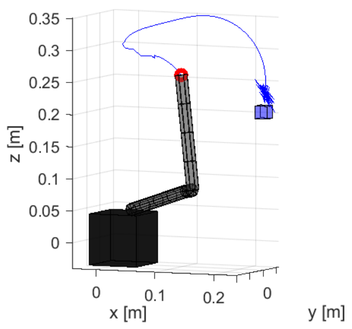


Figure 5.29: 3D-model of the Reach Alpha 5 with trace of the end effector when $q_v = 1°$ and $s = 4$ for new run of the secondary path.

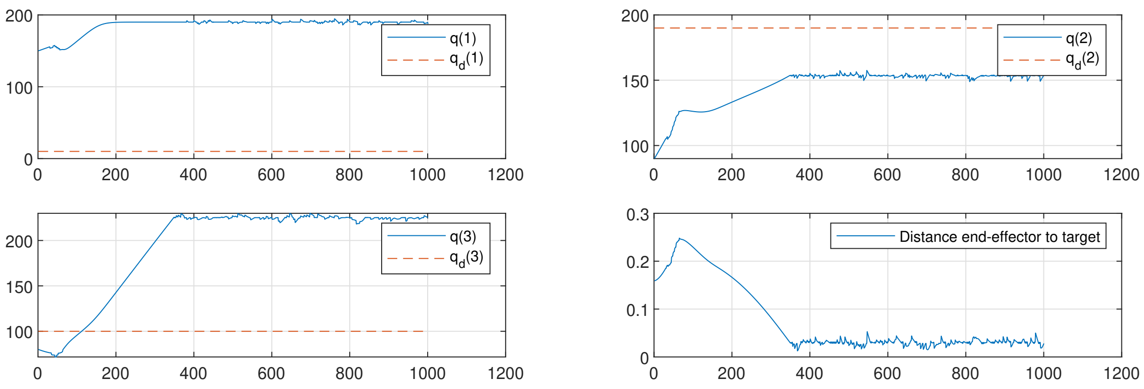


Figure 5.30: Joint variables $q(i)$ compared to the given reference $q_f(i)$ when $q_v = 1$ and $s = 4$ for new run of secondary path.

Simulation 2 repeated

Due to the random nature of the walk, it was possible that the first two simulations were outliers, and that the random walk would succeed in another instance. Simulation 2 was therefore repeated several times to ensure the validity of the result. None of these simulations were able to successfully generate a path from q_s to q_f. Presented here is the results from one of these simulations. Figure 5.29 shows the trace of the end-effector during simulation. In this case, the trace of the end-effector was smoother in the early stages of the path generation. This is a consequence of the random nature of the walk.

Figure 5.30 plots the manipulator joint values relative to their given reference for the repeated simulation.

Observations

The random walk was not able to pull the gradient descent algorithm out of the local minima. The walk parameters $q_v = 1°$ and $s = 4$ were originally assumed to be too large for practical use, as the end-effector could move far with each instance of the randomized walk. One can see from figures 5.25 and 5.27 that the end-effector moves as much as 5cm in different directions. Despite the large walks, no simulations succeeded. A new simulation was executed with an increased walk size to see if a complete path could be generated using the random walk method. The new walk was parameterized as $q_v = 4°$ and $s = 4$. In addition, the simulation was allowed to run for 10000 iterations before termination, as random elements with high variance could fail given a small sample. The joint configurations q_s and q_f were unchanged from simulation 2.

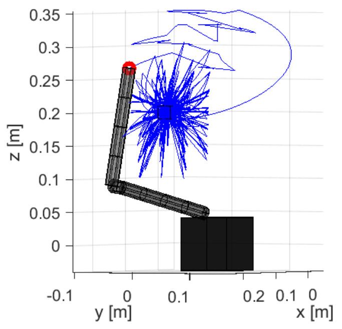


Figure 5.31: 3D-model of the Reach Alpha 5 with trace of the end effector when $q_v = 4°$ and $s = 4$.

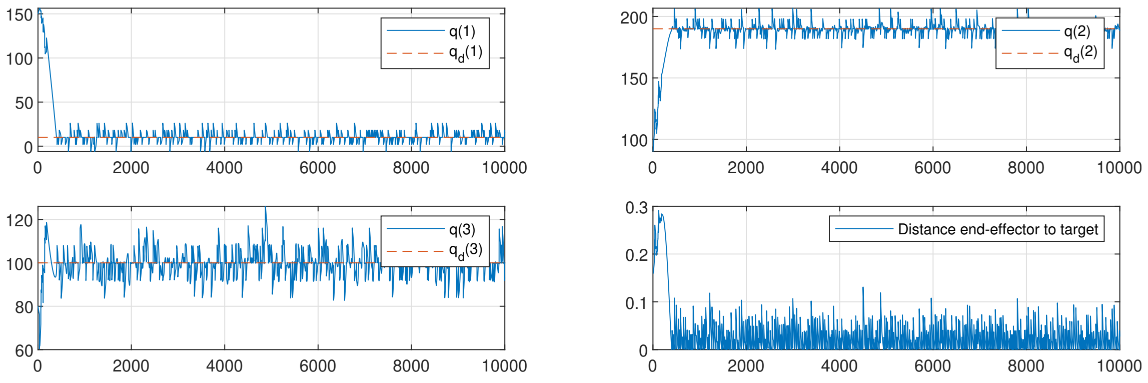


Figure 5.32: Joint variables $q(i)$ compared to the given reference $q_f(i)$ when $q_v = 4$ and $s = 4$.

Simulation 3

The third simulation executed with a modified randomized walk, where $q_v = 4°$ and $s = 4$. The simulation was terminated after 10000 iterations, not able to successfully generate a path from q_s to q_f. Figure 5.31 shows the trace of the end effector through the simulation. The algorithm could not escape the local minima, despite the large changes in configuration q^i after each randomized walk.

Figure 5.30 plots the manipulator joint values relative to their given reference for the simulation.

Simulation 3 repeated

The third simulation was also repeated multiple times to verify the result. While most cases lead to the gradient descent getting trapped in a minima, one case lead to a successful path. The results of that simulation are presented here. The simulation terminated successfully after 537 iterations. Figure 5.33 shows the trace of the end-effector during simulation.

Figure 5.34 plots the manipulator joint values relative to their given reference for the repeated simulation.

5.3 Wolfe conditions

This section will present the results of the experiment outlined in section 4.3.

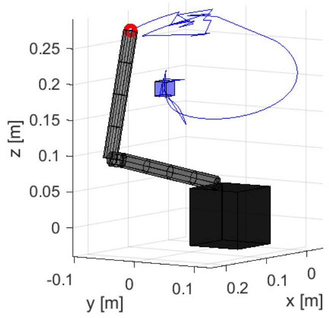


Figure 5.33: 3D-model of the Reach Alpha 5 with trace of the end effector when $q_v = 4°$ and $s = 4$ for new run of the same path.

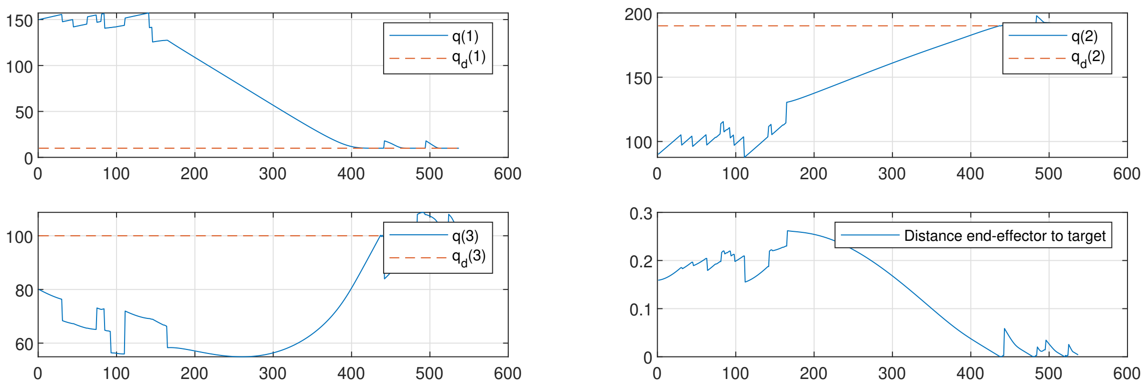


Figure 5.34: Joint variables $q(i)$ compared to the given reference $q_f(i)$ when $q_v = 4$ and $s = 4$ for new run of the same path.

Simulation 1

The path planner was executed with Wolfe conditions adjusting the step size α^i at each iteration of the search. The search was then forcefully terminated after some time, as the algorithm was stuck in a loop updating α^i indefinitely. A limit of 100 cycles was set before the loop would be forcefully terminated, returning the latest value for α^i. The simulation was executed once more, with termination condition after 100 iterations of the gradient descent search, due to excessive runtimes. Figure 5.35 shows the trace of the end effector during the search. The manipulator made large jumps which do not allow for a path to q_f.

Figure 5.36 shows the joint values at each iteration of the search.

The manipulator took excessively large steps in this simulation. Figure 5.37 shows the value for α at each iteration. The Wolfe conditions did not change alpha form its original value $\alpha_0 = 1$. Each step taken in the direction of the negative gradient was of size 1, resulting in the behavior from figure 5.35.

Observations

The initial step size $\alpha_0 = 1$ remains unchanged during the entire simulation. When the simulation was originally executed it had to be forcefully terminated. This indicates that α^i is not updated properly, resulting in exit conditions not being met. A second simulation was executed to verify that the initial step size was not causing the issue. A new initial step size $\alpha_0 = 0.1$ was selected.

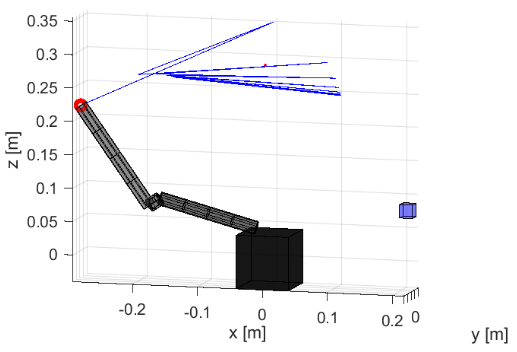


Figure 5.35: 3D-model of the Reach Alpha 5 with trace of the end effector when using Wolfe conditions. The step size α^i does not change from α_0.

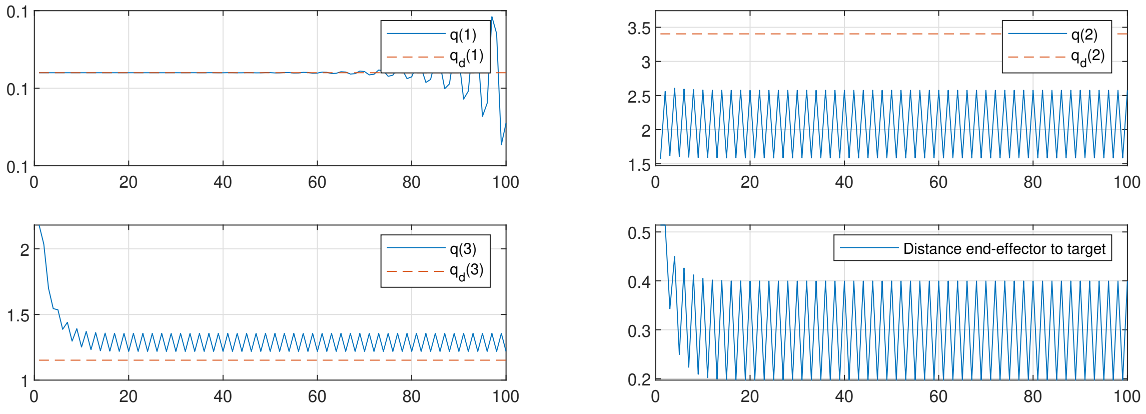


Figure 5.36: Joint variables $q(i)$ compared to the given reference $q_f(i)$ when using Wolfe conditions. The step size α^i does not change from α_0.

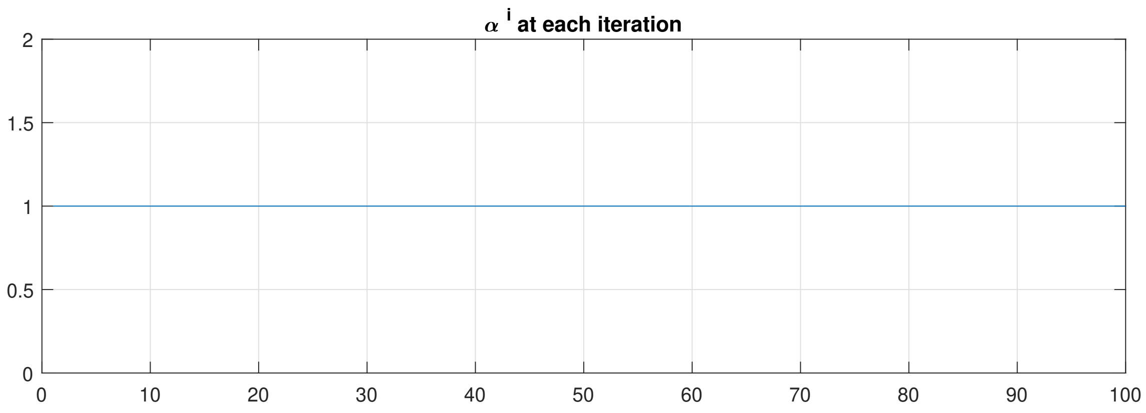


Figure 5.37: Value of α at every iteration of the search. The step size remains constant throughout the whole simulation.

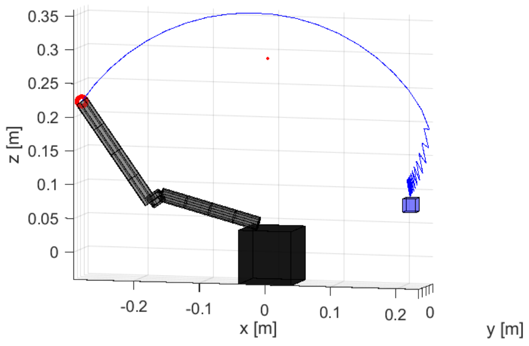


Figure 5.38: 3D-model of the Reach Alpha 5 with trace of the end effector when using Wolfe parameter. The step size α^i does not change from α_0.

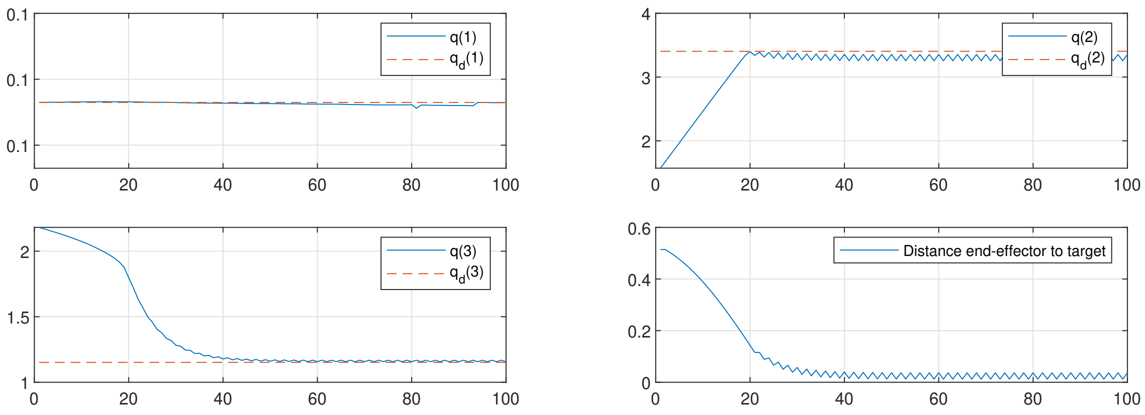


Figure 5.39: Joint variables $q(i)$ compared to the given reference $q_f(i)$ when using Wolfe conditions. The step size α^i does not change from α_0.

Simulation 2

The second simulation was executed between the same q_s and q_f, with an initial step size $\alpha_0 = 0.1$. Figure 5.38 shows the trace of the end-effector during simulation. The jittery motion towards the end of the path shows that the problem of constant step size still persists.

Figure 5.39 shows the joint values at each iteration of the search.

And figure 5.40 shows the value of α^i at each iteration of the search. This confirms that the issue with the algorithm is based in the implementation of the Wolfe conditions.

5.4 Obstacle avoidance

This section will present the results of the experiment outlined in section 4.4.

Simulation 1

The first simulation was executed without any obstacles. This would serve as a reference to subsequent simulations. Figure 5.41 shows the trace of the end effector for the full simulation. The algorithm terminated successfully after 215 iterations.

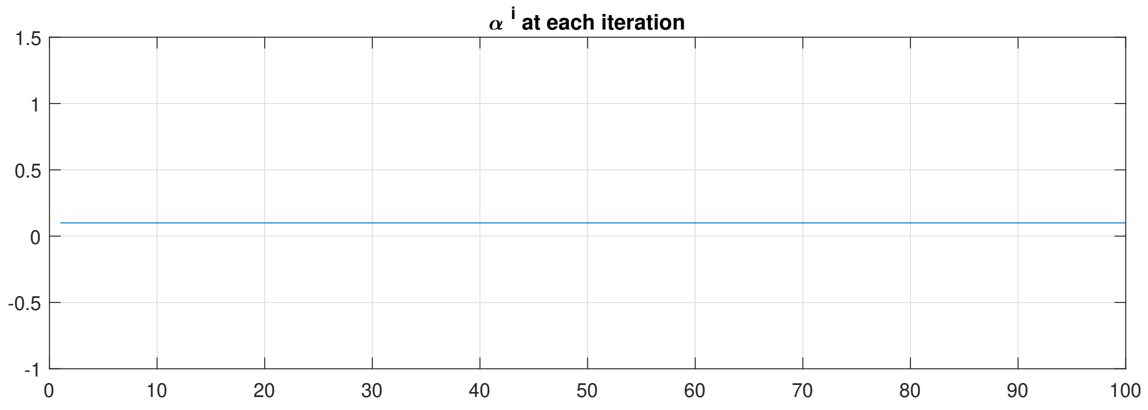


Figure 5.40: Value of α at every iteration of the search. The step size remains constant throughout the whole simulation.

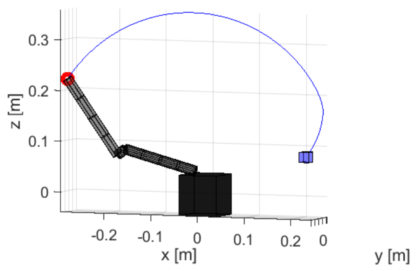


Figure 5.41: 3D-model of the Reach Alpha 5 with trace of the end effector. Motion planned by the baseline gradient descent algorithm.

Simulation 2

The second simulation was executed with a point obstacle in the direct path of the manipulator. The simulation ran for 3000 iterations until termination. A successful path from q_s to q_f was not found. Figure 5.42 shows the trace of the end-effector for the simulation.

The algorithm was able to avoid the obstacle, but got trapped in a local minima before completing the path. Figure 5.43 shows the manipulator prom the front.

Figure 5.44 shows the joint values through the search. The local minima was encountered much later in this simulation than any previous simulation.

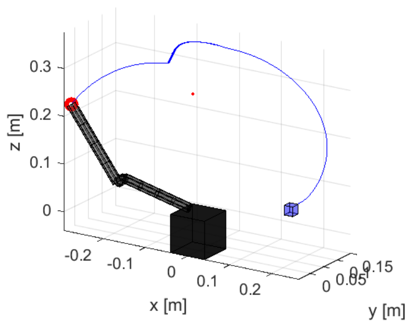


Figure 5.42: 3D-model of the Reach Alpha 5 with trace of the end effector. Motion planned by the baseline gradient descent algorithm with an obstacle in the workspace.

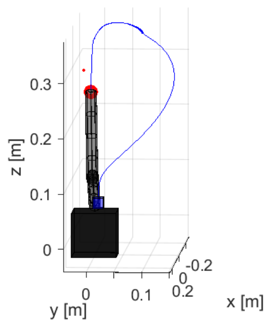


Figure 5.43: Front view of the manipulator. The end effector moves around the obstacle and avoids a collision.

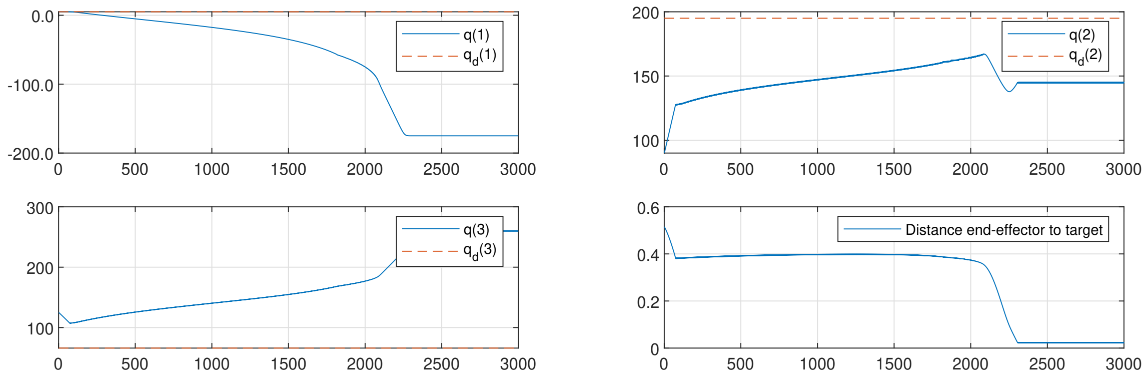


Figure 5.44: Joint variables $q(i)$ compared to the given reference $q_f(i)$ for obstacle avoidance.

Simulation 3

The third simulation was executed with a point obstacle in the direct path of the manipulator. The simulation ran for 3000 iterations until termination. A successful path from q_s to q_f was not found. Figure 5.45 shows the trace of the end-effector for the simulation.

This version of the gradient descent algorithm was also able to avoid the obstacle, but got trapped in the same local minima. The randomized step was not able to move the manipulator out of said minima.

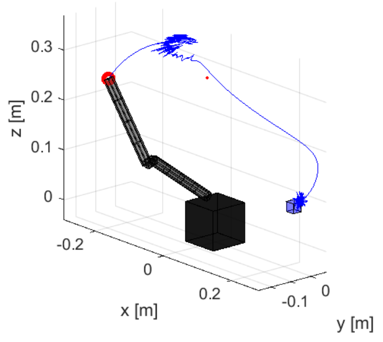


Figure 5.45: 3D-model of the Reach Alpha 5 with trace of the end effector. Motion planned by the random walk gradient descent algorithm with an obstacle in the workspace.

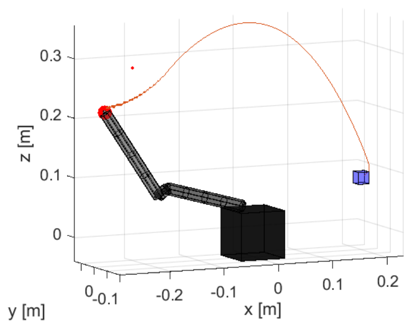


Figure 5.46: 3D-model of the Reach Alpha 5 with trace of the end effector. The obstacle has been moved to a different position closer to the initial position of the end-effector.

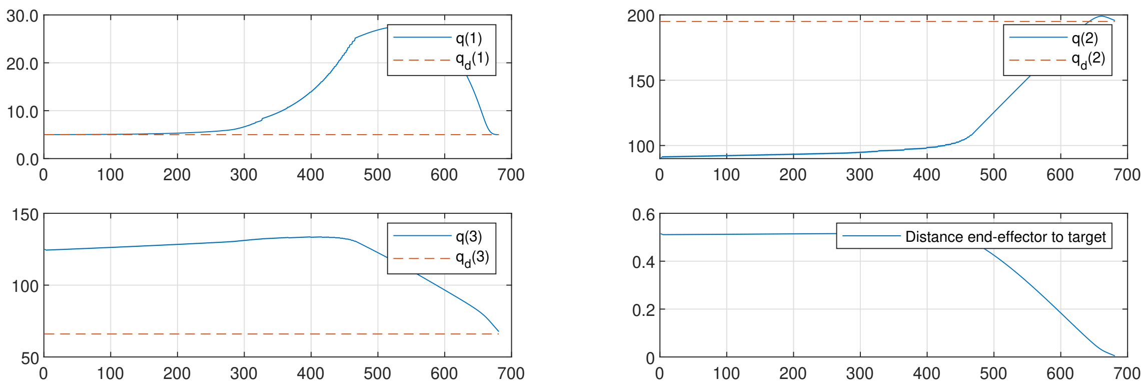


Figure 5.47: Joint variables $q(i)$ compared to the given reference $q_f(i)$ for obstacle avoidance.

Observations

As was originally predicted, local minima prevent the algorithms from completing the full path. However, the algorithms are able to avoid the workspace obstacle for the entirety of the motion. Because of this, a new simulation was conducted with a different obstacle position.

Simulation 4

The new position for the obstacle is $b = [-23e^{-2} \quad -2e^{-2} \quad 29.3e^{-2}]$. This places the obstacle much closer to the initial position of the end-effector. Randomized walk was not included in the algorithm for this simulation. The simulation terminated successfully after 681 iterations. Figure 5.46 shows the trace of the end-effector through the generated path. In this instance, the algorithm was able to complete the path without getting trapped in a local minima.

Figure 5.47 shows the joint values through the search.

5.5 Practical application of the path planning algorithm on the Reach Alpha 5

Two paths were chosen for the Reach Alpha 5. The first path is shown in figure 5.41. The second path is shown in figure 5.46. These paths were chosen as they were among few that successfully connected q_s and q_f.

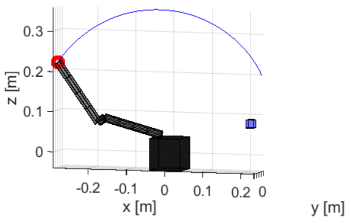


Figure 5.48: 3D-model of the Reach Alpha 5 with trace of the end effector.

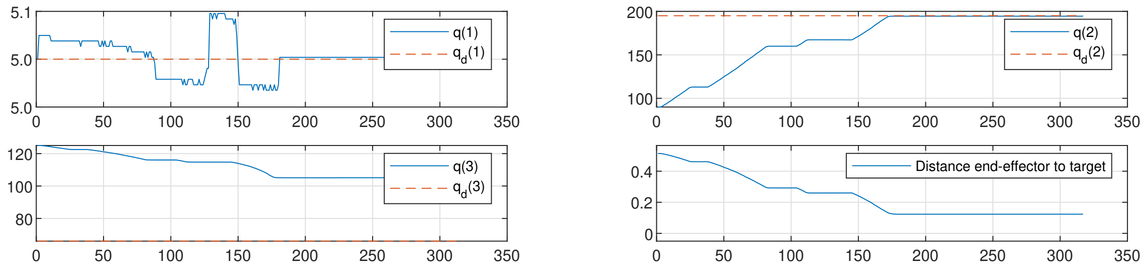


Figure 5.49: Joint variables $q(i)$ compared to the given reference $q_f(i)$ for Reach Alpha 5

Path 1

Path 1 consists of 215 set points. The full motion of Reach Alpha 5 returned 317 measurements of joint values q. Figure 5.48 shows the trace of the end-effector for the full motion. Even though the path was complete, the Reach Alpha 5 was not able to reach the target position. The figure shows this through the position of the end-effector.

Figure 5.49 shows the joint positions and final reference. Joint 1 and 2 converge to the reference, but joint 3 does not.

Path 2

Path 3 consists of 681 set points. The full motion of Reach Alpha 5 returned 405 measurements of joint values q. Figure 5.48 shows the trace of the end-effector for the full motion. Also in this instance, the Reach Alpha 5 was not able to reach the target position. The shape of the end-effector curve is still similar to that of the curve for the simulated manipulator.

Figure 5.49 shows the joint positions and final reference. Only joint 2 converges towards the final reference.

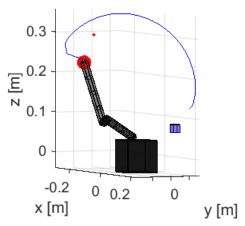


Figure 5.50: 3D-model of the Reach Alpha 5 with trace of the end effector.

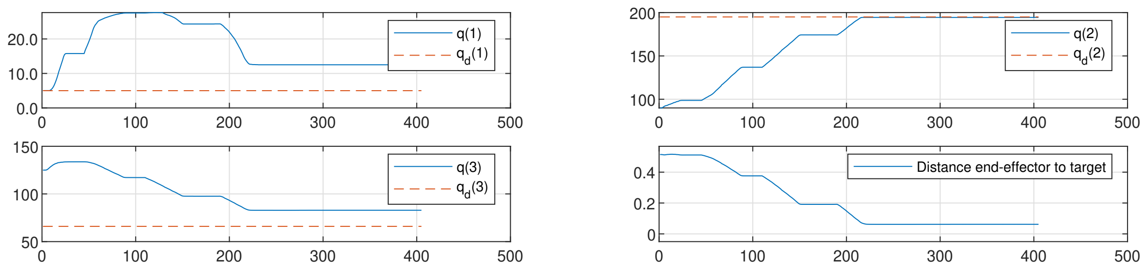


Figure 5.51: Joint variables $q(i)$ compared to the given reference $q_f(i)$ for Reach Alpha 5

Chapter 6

Discussion

Attractive field gains

One unexpected problem when using the path planning algorithm was the abundance of local minima. If q_s and q_f are arbitrarily chosen, there is a high likelihood that the search is caught in a local minima. The robotics textbook[22] mentions the problem of local minima, but did not make it clear how common of a phenomenon it is. They are mostly mentioned in reference to obstacles and certain configurations where the manipulator reaches around them. Figure 2.8 in section2.3.1.6 illustrated such a case. Based on experimentation, is seems like simple motions in a 2-dimensional plane are more likely to allow for path planning, while motions that move each joint through 3-dimensional space are plagued by local minima. It was based on this intuition that the joint configurations q_s and q_f used in the obstacle avoidance test were found.

A strong conclusion could not be drawn from the attractive gains test. The appropriate weighing of γ would shift based on the path being generated. The principal of using a higher weight on a single joint is expected to yield the best results if the issue of local minima were to be solved.

A noteworthy observation made in the attractive gains test was the tendency for joint 1 to converge to $q(1) = q_D(1) + 180°$. This would be expected if the position of the end-effector was used as a reference, since the inverse kinematics would solve for two different solutions. In this case, a configuration q_f was used as a reference, meaning that there was only one solution. This error implies that there is a mistake in the implementation, likely related to a trigonometric function. No such error was found. In addition, joint 1 was able to converge properly for certain paths. The cause of this is yet to be determined.

One way to expand on the current approach to attractive field gains is to implement adaptive gains. Changing gains during the search could improve performance by reducing weight on joints already close to their destinations. This could , however, exacerbate the problem of local minima. Any configuration of positive weights should theoretically lead to reduction in the field U, which implies that there is merit in implementing an adaptive scheme.

Random step

The randomized walk method was not able to reliably move the search algorithm out of a local minima, despite being the only method presented in the robotics textbook. In cases where the baseline algorithm got stuck, the randomized walk variant required numerous tries to succeed, if it succeeded at all. The values used in the test were already larger than what should be allowed on a real manipulator. A change of 4°in three joints could displace the end-effector by several centimeters. Figure 5.25 in section 5.2 illustrates this. This can be dangerous as the random walk does not take obstacles into account. If obstacles are taken into account by the random walk, it will not be able to escape the obstacle-created minima it was designed to move out of. Situations like the one illustrated in figure 2.8 cannot be escaped, as the manipulator would need to reverse too far backwards to see any alternative directions.

Due to the difficulty of escaping local minima, an alternative approach to path planning is recommended. The potential fields method was chosen due to being common in literature, as well as being thought in the robotics textbook which supplied most of the theory used in this thesis. There are, however, several other approaches one could take. One such approach is the *Rapidly-exploring random trees*(RRT)-method. This method is based around growing a tree in configuration space rooted at the initial configuration of the manipulator[3]. Branches of the tree are created between the root an randomly placed points in space. Variations of the RRT method are used in recent literature, and Matlab has built in functionality for this approach[12].

Wolfe conditions

Based on the results of the Wolfe conditions test, it is clear that the implementation was faulty. The step size α was not updated, likely due to a mistake in how the two inequalities were translated to this optimization problem. With proper implementation, better performance of the path planning algorithm is expected. However, no definitive conclusions can be drawn from this test.

An assumption was made in section 3.5 that a positive change in field U could be used as an indicator of a local minima. This was assumed after both the constant step algorithm and the Wolfe algorithm got the same result. Given that the Wolfe algorithm was faulty, the assumption does no longer hold for the general case. It did not matter in the case of this thesis, as only the constant step algorithm was used.

Obstacle avoidance

Although several simulations were not able to finish due to local minima, the repulsive fields effectively avoided collision between manipulator and obstacle. One improvement tho this implementation would be to use case-specific repulsive gains. A gain should be weighed less if the obstacle is located close to the final position of the manipulator. If the repulsive gains are static for all cases, there is an increased likelihood of encountering local minima close to the obstacle. The cutoff distance for the repulsive fields should also be chosen according to this principle.

Practical application of the path planning algorithm on the Reach Alpha 5

Despite following a complete path, the Reach Alpha 5 did not reach the goal position. Each set point was used as a reference to the manipulator, only changing reference when the manipulator was sufficiently close. Section 3.1 demonstrated that the internal PID-controllers are capable of following a reference to much higher precision. The error is likely caused by the threshold used for changing reference configuration being to large. This could have been tested, but the error was not discovered before access to the manipulator was gone.

Another noteworthy observation is the jagged motion of the manipulator. It can be seen in figure 5.51 in section 5.5, and it was very noticeable visually when conduction the test. This motion is caused by continuously feeding positional references with minimal changes to the PID-controllers. The motors perpetually activate and deactivate, causing the jagged motion. This would likely be damaging over time, so a different approach to controlling the manipulator is recommended. The motion shown in section 3.1 was smooth, which substantiates this assertion.

Chapter 7

Conclusion

This section will discuss how this thesis tackled the problems presented in the project statement, and give final remarks on the work done.

- **Software framework development:** Develop software for collision-free motion planning using the potential fields method.

This thesis has presented the theory and methods used in implementing a potential fields path planning algorithm. The algorithm was made using theory from the preceding project to this thesis, and expanded upon it by incorporating methods for handling local minima and inefficient optimization, as well as utilizing a more advanced structure for the repulsive field.

- **Virtual experiments to verify control system components:**
  1. Test basic motion planning capabilities in open air.
  2. Test of collision avoidance capabilities for avoiding obstacles in the manipulator workspace.

A set of experiments were conducted for testing the motion planning capabilities of the potential fields algorithm in an open workspace. The first test experimented with different weights in the attractive potential fields. The test was not able to determine what weight characteristic gave better results, as the path planning algorithm could not generate a complete path for several different sets of initial and final joint configuration. This was due to the algorithm getting trapped in local minima in the configuration space. The frequency at which the algorithm would encounter a local minima was unexpected. Another test was conducted in order to evaluate the effectiveness of the randomized walk, designed to circumvent local minima. The test showed that the randomized walk approach was not sufficient for escaping local minima. A third test was conducted in order to verify whether or not using Wolfe conditions would lead to a more efficient search. The test could not produce any results, as the Wolfe conditions had not been implemented correctly.

A test was also conducted for verifying that the path planning algorithm created the collision-free paths. The test showed that the manipulator was able to avoid an obstacle in the workspace, and in certain cases reach the desired destination. While the algorithm was not able to compete the path in some cases, it never collided with the obstacle.

- **Physical experiments with the Reach Alpha 5:** Conduct physical experiments using results from virtual experiments with the motion planning algorithm.

Paths generated by the potential fields path planing algorithm were given to the Reach Alpha 5, which then used those paths as references for joint positions. One path was taken from an open workspace, and the other path included a workspace obstacle. The Reach Alpha 5 failed to reach the final destination in both cases. This was due to the use of a threshold parameter which was not strict enough to bring the manipulator to the final destination. The way the path was used as a series of positional joint references led to an undesirable motion characteristic.

The thesis tackled every problem presented in the problem statement. The results from the tests that were conducted showed that this path planning algorithm is not able to perform the task or collision-free path planning consistently. The program used to operate the Reach Alpha 5 was implemented in such a way that it could likely damage the manipulator motors if used over an extended period if time. The Wolfe conditions were not implemented correctly according to theory. Despite all many flaws being caused by human mistakes, path planning approach itself has shown that it might not be a suitable option for such a complex task. Simple motion in free workspace with no restrictions on either joints or collisions still led to the algorithm getting trapped in local minima. Only 2-dimensional problems would yield a solution. If potential fields are to be used for path planning for the Reach Alpha 5, one would need to solve the problem of avoiding these local minima. One approach which likely leads to better performance is to utilise a different motion planning scheme altogether.

Bibliography

[1] Alexander Amini. *Spatial Uncertainty Sampling for End-to-End Control.* URL: https://www.researchgate.net/figure/Non-convex-optimization-We-utilize-stochastic-gradient-descent-to-find-a-local-optimum_fig1_325142728 (visited on 19/12/2021).

[2] Jason Brownlee. *Gradient Descent For Machine Learning.* URL: https://machinelearningmastery.com/gradient-descent-for-machine-learning/ (visited on 20/12/2021).

[3] Tim Chinenov. *Robotic Path Planning: RRT and RRT*.* URL: https://theclassytim.medium.com/robotic-path-planning-rrt-and-rrt-212319121378 (visited on 12/07/2022).

[4] Peter Corke. *Denavit-Hartenberg notation.* URL: https://robotacademy.net.au/lesson/denavit-hartenberg-notation/ (visited on 20/12/2021).

[5] Richard Scheunemann Denavit Jacques; Hartenberg. 'A kinematic notation for lower-pair mechanisms based on matrices'. In: (1955).

[6] Richard Scheunemann Denavit Jacques; Hartenberg. 'Kinematic synthesis of linkages'. In: (1965).

[7] Niklas Donges. *Gradient Descent: An Introduction to 1 of Machine Learning's Most Popular Algorithms.* URL: https://builtin.com/data-science/gradient-descent (visited on 20/12/2021).

[8] Martin Føre et al. *Precision fish farming: A new framework to improve production in aquaculture.* URL: https://www.sintef.no/publikasjoner/publikasjon/1520340/ (visited on 16/12/2021).

[9] Stephen J. Wright Jorge Nocedal. *Numerical Optimization.* Springer New York, 2006. DOI: 10.1007/978-0-387-40065-5. URL: https://doi.org/10.1007%2F978-0-387-40065-5.

[10] Sahdev Kansal. *Quick Guide to Gradient Descent and Its Variants.* URL: https://towardsdatascience.com/quick-guide-to-gradient-descent-and-its-variants-97a7afb33add (visited on 19/12/2021).

[11] Eleni Kelasidi. *Using robotics to drive up efficiency and minimise risks.* URL: https://blog.sintef.com/sintefocean/using-robotics-to-drive-up-efficiency-and-minimise-risks/ (visited on 20/12/2021).

[12] Mathworks. *Motion Planning with RRT for a Robot Manipulator.* URL: https://se.mathworks.com/help/nav/ug/motion-planning-with-rrt-for-manipulators.html (visited on 12/07/2022).

[13] MATLAB. *version R2021a.* Natick, Massachusetts: The MathWorks Inc., 2021.

[14] Oscar Nissen. 'Automating Tank Operations in Smolt Production - Control of an Underwater Manipulator'. In: (2021).

[15] Ahmed Sadiq, Firas Raheem and Noor F. Abbas. 'Robot Arm Path Planning Using Modified Particle Swarm Optimization based on D* algorithm'. In: *Al-Khwarizmi Engineering Journal* 13 (Oct. 2017). DOI: 10.22153/kej.2017.02.001.

[16] Hani Safadi. *Local Path Planning Using Virtual Potential Field.* URL: https://www.cs.mcgill.ca/~hsafad/robotics/ (visited on 23/11/2021).

[17] SINTEF. *Autosmolt 2025.* URL: https://www.sintef.no/globalassets/sintef-ocean/factsheets/aquanor/autosmolt-2025.pdf (visited on 16/12/2021).

[18] SINTEF. *Autosmolt2025.* URL: https://www.sintef.no/en/projects/2019/autosmolt2025/ (visited on 16/12/2021).

[19] Pål H. Skeide. 'Automating tank operations in smolt production – A concept study for automating tank cleaning using robotic arms'. In: (2020).

[20] Keith R. Solomon and Glen Van Der Kraak. ‘Organic Chemical Toxicology of Fishes’. In: (2013).

[21] Guido Van Rossum and Fred L. Drake. *Python 3 Reference Manual.* Scotts Valley, CA: CreateSpace, 2009. ISBN: 1441412697.

[22] Mark W. Spong, Seth Hutchinson and M. Vidyasagar. *Robot Modeling And Control.* John Wiley & Sons, Inc, 2006.

Appendix

.1 Project description

NTNU
Norwegian University of
Science and Technology

Faculty of Information Technology,
and Electrical Engineering,
Department of Engineering Cybernetics

MASTER THESIS

Name of the candidate: Henrik Baldishol

Discipline: Engineering Cybernetics

Project title (Norwegian): Baneplanlegging med kollisjonsunngåelse for 5-funksjons robotmanipulator for havbruksoperasjoner

Project Title (English): Path planning with obstacle avoidance for 5 function robot manipulator for aquaculture operations

Background:
There is a present trend in the aquaculture industry with operations and production methods becoming less reliant on manual labour and experience-based reasoning, and more based upon objective indicators, automation principles and decision support systems. This harmonises with the philosophy of Precision Fish Farming (PFF) where new technology and automation principles are employed to improve human control over the biological processes in different phases in the aquaculture production cycle. Although most of the weight gain in salmon production is achieved during the sea-based ongrowing phase, the success of the production also depends strongly on the quality of the fish when transferred to sea-cages. This is determined by how well the fish are managed during the smolt production phase that precedes the ongrowing phase. Smolt production is conducted in land-based tank facilities and covers the period from hatching until the fish undergo smoltification (i.e., a metamorphosis adapting them from living in fresh water to handling seawater).

Utilization of robotic manipulators with computer vision techniques have the potential to simplify or automate several different challenges in various industry segments. Within land-based rearing of smolt in rearing tanks, manipulators can be used for different applications. Specifically, recent research activities at SINTEF Ocean and NTNU have focused on developing control system components and algorithms for controlling a 5 function manipulator for such purposes. Moreover, NTNU recently acquired a 5 function robot manipulator (Blueprint Labs, Reach Alpha 5) that can be used as a physical platform for further developing and validating these methods.

This master project is linked with these activities, and builds on a preceding specialization project that was aimed at developing a software framework for virtual development of advanced control system components for conducting automated operations in tanks for smolt production. The master project will continue the development of the software framework, and expand it with new functionality where needed. Moreover, the project will set up and run specific simulation trials to test key functionality in the control system components, primarily collision avoidance approaches. All developments will be designed for the Blueprint Labs, Reach Alpha 5 manipulator. Once the functionality has been verified through simulation experiments, the same methods will be tested on the physical 5 function robot manipulator in physical experiments mimicking the simulation experiments.

The project will contain the following elements:

- Software framework development
    - Implement collision avoidance methods to prevent self-collision (i.e., collision between links), collisions with ground/fundament or obstacles
- Virtual experiments to verify control system components such as:
    - Basic tests of motion planning in air without obstacles
    - Collision avoidance to avoid self-collisions, ground collisions and dynamic/static obstacles
- Physical experiments with robotic manipulator:
    - Physical experiments mimicking the virtual experiments
    - Compare performance with simulation outcomes

Start date:	February 01, 2022
Delivery date:	July 12, 2022

Conducted at the Department of Engineering Cybernetics, NTNU

Main supervisor: Martin Føre

Co-supervisor(s): Bent Haugaløkken (SINTEF Ocean)

Trondheim, February 15, 2022

Supervisor

.2 Reach Alpha 5 datasheet

Reach System 1 Kinematic and Dynamic Properties

Blueprint Lab

Last Update: September 2019

1 Kinematic Properties

Link	**d (mm)**	$\boldsymbol{\theta}$	**a (mm)**	$\boldsymbol{\alpha}$
0	46.2	$\theta_0 + \pi$	20	$\pi/2$
1	0	$\theta_1 - \theta_a$	150.71	π
2	0	$\theta_2 - \theta_a$	20	$-\pi/2$
3	-180	$\theta_3 + \pi/2$	0	$\pi/2$
4	0	$-\pi/2$	0	0

Table 1: Standard DH Parameters for R5M with $\theta_a = \tan^{-1}\left(\frac{145.3}{40}\right)$

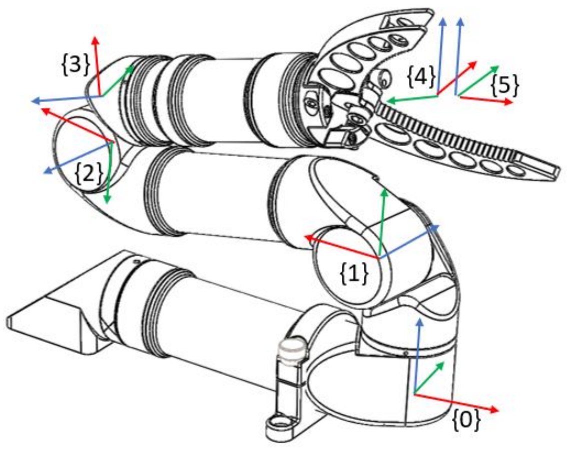


Figure 1: R5M joint frames (x,y,z)

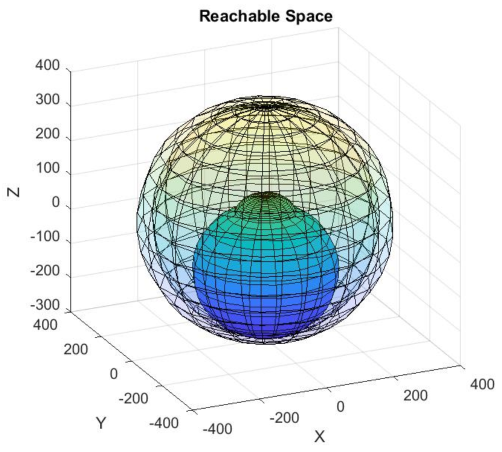


Figure 2: Reachable workspace without self collision showing inner and outer limits

1.1 Workspace

The outer reachable limits form a torus

$$(\sqrt{x^2+y^2}-a_0)^2+(z-d_0)^2 \leq (a_1+\sqrt{d_3^2+a_2^2})^2 \tag{1}$$

The inner reachable limit is the torus

$$(\sqrt{x^2+y^2}-a_0)^2+(z-d_0)^2 \geq ((39.94+a_2)^2+(145.3+d_3)^2) \tag{2}$$

The inner reachable limit with the arm in the downward position is the torus

$$(\sqrt{x^2+y^2}-a_0)^2+(z-d_0+145.3)^2 \geq (-d_3)^2 \tag{3}$$

These are shown in Figure 2.

1.2 Inverse Kinematics

From Figure 3 we can solve for the underarm solution

$$\theta_0 = tan2^1(\frac{y}{x})+\pi \tag{4}$$

$$R = \sqrt{x^2+y^2} \tag{5}$$

From Figure 4 we can solve for

$$l_1 = a_1 \tag{6}$$

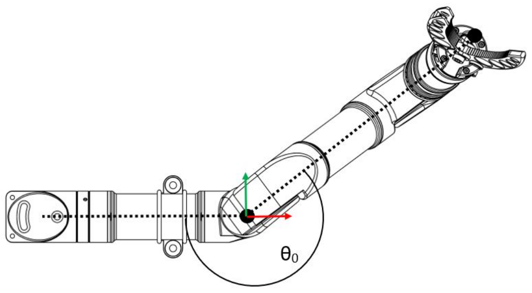


Figure 3: Calculation of θ_0

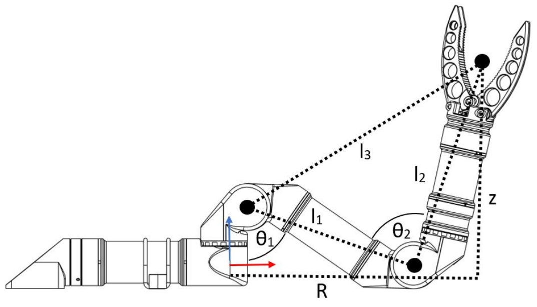


Figure 4: Calculation of θ_1 and θ_2

$$l_2 = \sqrt{a_2^2 + d_3^2} \tag{7}$$

$$l_3 = \sqrt{(R - a_0)^2 + (z - d_0)^2} \tag{8}$$

$$\theta_2 = cos^{-1}(\frac{l_1^2 + l_2^2 - l_3^2}{2l_1 l_2}) - sin^{-1}(\frac{2a_2}{l_1}) - sin^{-1}(\frac{a_2}{l_2}); \tag{9}$$

$$\theta_1 = \frac{\pi}{2} + tan2^{-1}(\frac{z - d_0}{R - a_0}) - cos^{-1}(\frac{l_1^2 + l_3^2 - l_2^2}{2l_1 l_3}) - sin^{-1}(\frac{2a_2}{l_1}) \tag{10}$$

Now we can also solve for the overarm solution from Figure 5

$$\theta_0 = tan2^1(\frac{y}{x}) \tag{11}$$

Finally from Figure 6

$$l_3 = \sqrt{(R + a_0)^2 + (z - d_0)^2} \tag{12}$$

$$\theta_2 = cos^{-1}(\frac{l_1^2 + l_2^2 - l_3^2}{2l_1 l_2}) - sin^{-1}(\frac{2a_2}{l_1}) - sin^{-1}(\frac{a_2}{l_2}); \tag{13}$$

$$\theta_1 = \frac{3\pi}{2} - tan2^{-1}(\frac{z - d_0}{R + a_0}) - cos^{-1}(\frac{l_1^2 + l_3^2 - l_2^2}{2l_1 l_3}) - sin^{-1}(\frac{2a_2}{l_1}) \tag{14}$$

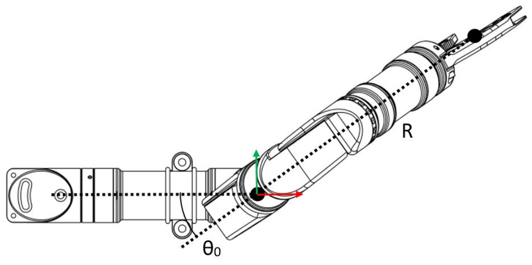


Figure 5: Calculation of θ_1 and θ_2

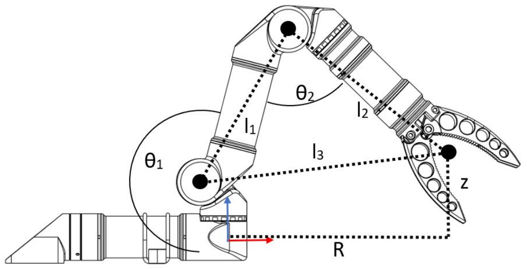


Figure 6: Calculation of θ_0

2 Inertial Properties

Link	**Mass (kg)**	**COM (mm)**	**I ($kg.mm^2$)**
0	0.341	$\begin{pmatrix} -75 & -6 & -3 \end{pmatrix}$	$\begin{pmatrix} 99 & 139 & 115 \\ 139 & 2920 & 3 \\ 115 & 3 & 2934 \end{pmatrix}$
1	0.194	$\begin{pmatrix} 5 & -1 & 16 \end{pmatrix}$	$\begin{pmatrix} 189 & 5 & 54 \\ 5 & 213 & 3 \\ 54 & 3 & 67 \end{pmatrix}$
2	0.429	$\begin{pmatrix} 73 & 0 & 0 \end{pmatrix}$	$\begin{pmatrix} 87 & -76 & -10 \\ -76 & 3190 & 0 \\ -10 & 0 & 3213 \end{pmatrix}$
3	0.115	$\begin{pmatrix} 17 & -26 & -2 \end{pmatrix}$	$\begin{pmatrix} 120 & -61 & -1 \\ -61 & 62 & 0 \\ -1 & 0 & 156 \end{pmatrix}$
4	0.333	$\begin{pmatrix} 0 & 3 & -98 \end{pmatrix}$	$\begin{pmatrix} 3709 & 2 & -4 \\ 2 & 3734 & 0 \\ -4 & 0 & 79 \end{pmatrix}$

Table 2: Inertial properties for R5M

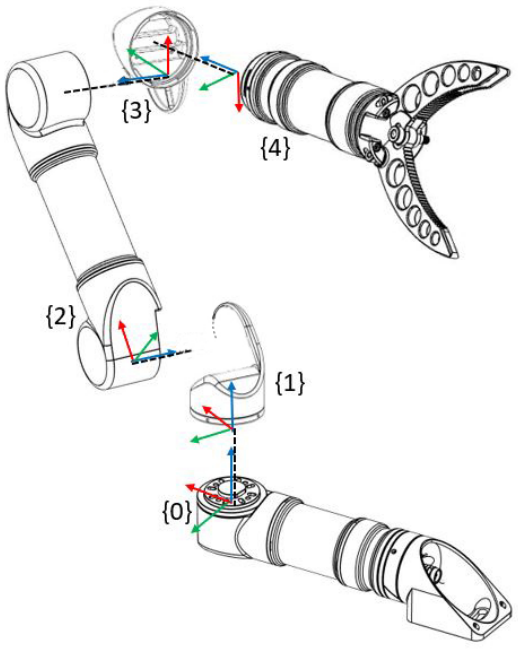


Figure 7: Inertial frames for each link (x,y,z)

3 Hydrodynamic Properties

3.1 Added Mass

We use standard equations for added mass assuming cylindrical sections with spherical ends.

Link	$(\boldsymbol{X_{\dot{u}}, Y_{\dot{v}}, Z_{\dot{w}}})(\boldsymbol{kg})$	$(\boldsymbol{K_{\dot{p}}, M_{\dot{q}}, N_{\dot{r}}})(\boldsymbol{kg.mm^2})$
0	$\begin{pmatrix} 0.017\rho & 0.189\rho & 0.189\rho \end{pmatrix}$	$\begin{pmatrix} 0 & 1414\rho & 1414\rho \end{pmatrix}$
1	$\begin{pmatrix} 0.032\rho & 0.032\rho & 0.017\rho \end{pmatrix}$	$\begin{pmatrix} 7\rho & 7\rho & 0 \end{pmatrix}$
2	$\begin{pmatrix} 0.017\rho & 0.201\rho & 0.201\rho \end{pmatrix}$	$\begin{pmatrix} 0 & 1716\rho & 1716\rho \end{pmatrix}$
3	$\begin{pmatrix} 0.032\rho & 0.017\rho & 0.032\rho \end{pmatrix}$	$\begin{pmatrix} 7\rho & 0 & 7\rho \end{pmatrix}$
4	$\begin{pmatrix} 0.226\rho & 0.226\rho & 0.017\rho \end{pmatrix}$	$\begin{pmatrix} 2443\rho & 2443\rho & 0 \end{pmatrix}$

Table 3: Added mass terms where $\rho \sim 1$ is the density in kg/L

3.2 Viscous Damping

We consider only quadratic drag assuming Reynolds numbers above the Stokes flow approximation at any significant velocity. We also consider only translational drag. We approximate the Reynolds number as

$$Re = \frac{\rho u L}{\mu} \sim \frac{10^3 * 0.1 * 0.1}{10^{-3}} \sim 10^{-4} \tag{15}$$

where ρ is the fluid density, u is the relative velocity, L is the characteristic length and μ is the dynamic viscosity, which gives a drag coefficient of $c_d \sim 0.5$. We now calculate the quadratic drag using

$$F_d = \frac{1}{2}\rho u^2 c_d A \tag{16}$$

where A is the cross sectional area.

The centre of drag is assumed to coincide with the centre of buoyancy

Link	$(X_{u\|u\|}, Y_{v\|v\|}, Z_{w\|w\|})$ $(N/\sqrt{ms^{-1}})$
0	$\begin{pmatrix} 0.3\rho & 1.5\rho & 1.5\rho \end{pmatrix}$
1	$\begin{pmatrix} 0.26\rho & 0.26\rho & 0.3\rho \end{pmatrix}$
2	$\begin{pmatrix} 0.3\rho & 1.6\rho & 1.6\rho \end{pmatrix}$
3	$\begin{pmatrix} 0.26\rho & 0.3\rho & 0.26\rho \end{pmatrix}$
4	$\begin{pmatrix} 1.8\rho & 1.8\rho & 0.3\rho \end{pmatrix}$

Table 4: Drag terms where $\rho \sim 1$ is the density in kg/L

3.3 Buoyancy

Link	**Volume (L)**	**COB (mm)**
0	0.202	$\begin{pmatrix} -75 & -6 & -3 \end{pmatrix}$
1	0.018	$\begin{pmatrix} -1 & -3 & 32 \end{pmatrix}$
2	0.203	$\begin{pmatrix} 73 & 0 & -2 \end{pmatrix}$
3	0.025	$\begin{pmatrix} 3 & 1 & -17 \end{pmatrix}$
4	0.155	$\begin{pmatrix} 0 & 3 & -98 \end{pmatrix}$

Table 5: Buoyancy terms with Centre of Buoyancy (COB)

4 Actuator Properties

$$torque = 90.6 * (current \pm 43.0) \tag{17}$$

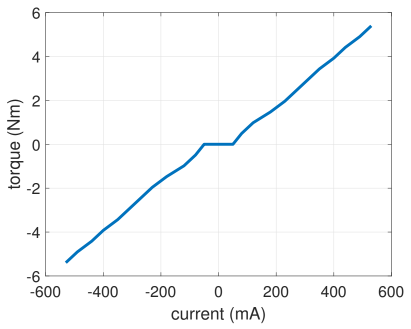


Figure 8: Plot of torque vs current for high torque joint