HAN_UNIVERSITY OF APPLIED SCIENCES

HAN Master Major Project

Major Project Report

Modelling and control of FANUC R-2000iC/210F as part of a production line for lightweight automotive parts

Student Number: 617931

Name: Karl Wallkum

Track: Master Control Systems Engineering

Company: HAN Smart Production Cell (IPKW)

Supervisors: Nguyen Trung(Company Supervisor)
Wesselingh Ellen(HAN Supervisor)
Jeltsema Dimitri(HAN Supervisor)

Date: 06/05/2020

STATEMENT OF OWN WORK

For exams (computer-based or otherwise)/assignments/reports/theses

Master: ☐ Automotive Systems ☒ Control Systems Engineering
Date of submission: 04.05.2020
Manner of submission: Email
Name: Karl Wallkum
Student number: 617931
Phone number: +49 157 7154 5726
E-mail: karl.wallkum@yahoo.de
Signature(s):  Digital signiert von Karl Wallkum DN: cn=Karl Wallkum, c=DE, email=karl.wallkum@yahoo.de Grund: Ich stimme den angegebenen Bedingungen durch meine digitale Signatur in diesem Dokument zu Ort: Arnhem Datum: 2020.04.17 15:40:52 +02'00'

By signing this form, I declare that the above-mentioned thesis that I have submitted (hereafter referred to as the 'document') was independently created by me without any external help and that I am aware of the rules concerning irregularities/fraud as set out in the degree statute.

Abstract

This project integrates a FANUC 210F 6 axis industrial robot arm into an experimental production line at the Smart Production Centre (SPC). This included setting up the robot and putting it to work as part of the production line. As this production line is set in a research environment, gaining a deeper understanding of all involved systems is desired. The structural definition describing the Cartesian position of the endeffector depending on the joint angles was obtained through a kinematic analysis. The dynamic relation of the forces acting on each joint due to movements and external forces is described by the equations of motion which are derived by a Lagrangian formulation. Values for this kinematic and dynamic model are gained through datasheets and assumed where necessary. This kinematic and dynamic model has been implemented in Matlab using the robotics toolbox created by Peter Corke. With this model, two types of control strategies, applied torque control and workspace control were tested. Results were compared in terms of point to point and tracking performance. Simulation results using Matlab showed, that workspace control has superior performance.

Figure 1: FANUC R-2000iC/210F 6-axis industrial robot arm at the SPC

Contents

List of Tables

List of Figures

List of Symbols

a_i	translation along x_i-axis
a_{xyz}	approach vector
α	rotation around x
α_i	rotation around the x_i-axis
β	rotation around y
c_i	$\cos(\theta_i)$
d_i	translation along z_i-axis
γ	rotation around z
h	height
I	Number of links
i	link iteration variable
$\mathcal{I}$	Innertia Matrix
J	Jacobian matrix
$T(q, \dot{q})$	Kinetic Energy
L	Lagrangian
k	length
m	link mass
N	Number of joints
n	joint iteration variable
n_{xyz}	normal vector
o_{xyz}	orientation vector
p	Position with orientation in euclidean space $p = (x, y, z, \alpha, \beta, \gamma)$
p_{xyz}	position vector
$V(q)$	Potential energy
q	joint angles in joinet space $q = (q_1, q_2, q_3, q_4, q_5, q_6)$
R	Rotation matrix
s_i	$\sin(\theta_i)$
T	Transformation matrix
θ_i	rotation around z_i-axis
Υ	actuator torque

v	Vector
w	width
x	x-position within frame
x_i	local coordinate frame x -axis
y	y-position within frame
y_i	local coordinate frame y -axis
z	z-position within frame
z_i	local coordinate frame z -axis

Acronyms

CAD Computer Aided Design

CNC Computer Numeric Control

DCS Dual Check Safety

DDS Data Distribution System

DH Denavit-Hartenberg

DOF degrees of freedom

DT Digital Twin

EOAT End Of Arm Tooling

ETCS European Train Control System

fig figure

FRP Fibre Reinforced Plastics

HMI Human Machine Interface

IPKW Industrial Park Kleevse Waard

MIC Mobility Innovation Center

MRAC Model Reference Adaptive Controller

OPC UA Open Platform Communications Unified Architecture

OS Operating System

PC Personal Computer

PLC Programmable Logic Controller

ROS Robot Operating System

SPC Smart Production Centre

USB Universal Serial Bus

VC Visual Components

VFC Variable Frequency controller

Glossary

atan2 The atan2(x, y)-function used in the inverse kinematic solution returns an angle between the positive x axis and a line pointing to $(x, y) \neq (0, 0)$. The function takes two arguments and returns a single value θ between $-\pi$ and π equivalent to the phase angle of complex numbers. While arctan can only solve in quadrant 1 and 4, arctan2 allows for calculation in all four quadrants because it differentiates between pos/neg x and y direction independently. It is mostly a question of implementation and solving method, so this new notation arose with programming languages.

closed-form solution Closed form solutions in the context of this thesis describe a solution in a closed form expression and are used complementary to numerical Solutions.

end of arm tool The EOAT refers to the equipment that interacts with the payload. Examples for this are a gripper or a welding torch.

endeffector The endeffector describes the end of the robotic arm. In the context of this thesis, this describes the end of the serial link mechanism without any EOAT.

extraneous root An extraneous root is introduced into an equation in the process of solving another equation, but is not a solution of the equation to be solved [30]. This test is done simply by substituting the solution into the original equation .

forward kinematics Forward kinematics (also called direct kinematics) describes the position of the endpoint of a kinematic chain in the operational space by the kinematic equations with the joint variables as input. The non-linear kinematic equations map the joint parameters to the configuration of the robot system. This results in a pure geometrical description of motion by means of position, orientation, and their time derivatives.

inverse kinematics Inverse Kinematics (also called backward kinematics) gives a set of solutions for a kinematic chain to reach a desired endpoint position. Depending on the type of kinematic chain, the number of solutions can be more than 1 or zero.

iPendant Wired I/O device for FANUC robots that connects to the robot controller (here R30iA). Provides a touchscreen, buttons, deadman switch and E-stop for manual control, programming, monitoring and configuration of the robot.

joint A joint is a connection between two links that allows for movement within certain contstraints.

link A link is a rigid body, defining the spatial relationship between two following axes. Links can be rotated or translated by joints, but don't deform themselves.

numerical solution Originally the term numerical solution describes a solution given in terms of numbers instead of an explicit closed form expression. In this work, the term "numerical solution" refers to numerical, iterative methods for solving the inverse kinematics problem by using a sequence of steps leading to incrementally better solutions for the joint angles. Examples for these methods are the jacobian inversion method, the optimization based method cyclic coordinate descent as shown by Lukas Barinka [6].

prismatic joint Prismatic joints slide along an axis like linear bearings. They provide a translation while resisting rotation.

revolute joint Revolute joints rotate arount an axis like door hindges and folding mechanisms. They provide a single-axis rotation function that does not allow translation or sliding linear motion.

robot Robots [39] can be defined as programmable movement automatons [4] that can perform tasks without human supervision and can be taught at least repetitive tasks. Increasingly, also ways to sense their surroundings are added to improve their movements according to the situation [45]. These sensors are placed additionally to the standard feedback-control sensors like pulse encoders and allow the robot to handle a wider range of situations without human intervention.

1. Introduction

1.1 Background

Production line The production line located at Industrial Park Kleevse Waard (IPKW) in the Mobility Innovation Center (MIC) building is being developed by the Smart Production Centre (SPC). The production line will produce steering knuckles made of Fibre Reinforced Plastics (FRP). To compete in the future develpements of "Smart Manufacturing", *the Smart Cell project investigates on an automated production cell for the realization of structural composite components which must be produced in large scale or mass produced* (see [1], Smart-Cell project description).

Origin of robots The term robot was first mentioned 1920 by Karel Čapek in a 1920 Czech science fiction play "Rossum's Universal Robots" and means serf labor but colloquially means hardwork or drudgery [14]. With the upscaling of production in the past century, industrial robots were developed to do work, that was too repetitive, dangerous or difficult for workers. The first patent for robots was filed in 1954 by George C. Devol. It was a mechanical arm with a gripper, mounted on a track. Motion sequences were encoded as magnetic patterns on predecessors of harddrives. Unimation, founded by Devol and Joseph Engelberger in 1956 installed their first industrial robot Unimate, shown in fig. 1.1, in 1961.

Figure 1.1: Unimate, the grandfather of industrial robots [24]

Simple robots CNC mills and 3D printers allow automated manufacturing with varying degrees of precision and speed. These are very simplistic robotic systems based on a feedforward control with stepper motors for position control. For starting a production process, these devices need to be calibrated and the position and orientation are taught by pointing the drill/printing head to the markerpoints manually. As no feedback control is usually available on these devices, loss of step control and other errors are possible and can go unnoticed until the end of production, when the part is inspected.

Current robots Highly developed automated platforms like the Fanuc 210F have feedback-sensors and -control for point to point as well as tracking motion, computer vision as seen on the deltarobot at SPC (see section 3.1 Deltarobot), digital and analog I/O, builtin- safety like the Dual Check Safety (DCS) motion safety system and high speed fieldbus connectivity like Profinet. With a teach pendant, these Robots can be configured

and programmed. Alternatively, Personal Computer (PC)-based proprietary manufacturer-specific software like Roboguide with a model based test cell can be used to program the robot.

Future robots Within research in materials and production, the automation environment system is of interest for this research. To manage the interaction between material but also the machines among each other, a Data Distribution System (DDS) couples the subsystems as a middleware as seen in figure (fig). 1.2. A middleware is "glue code" that helps to interface systems with each other to exchange commands and data to use it as a single robotic system. Robot Operating System (ROS) is a proposed middleware to model, simulate, and control the production line as a homogenious distributed system. Besides the classical publish subscribe pattern to simplify network programming, ROS provides an ecosystem of interchangeable nodes and drivers with a wide codebase for complex tasks.

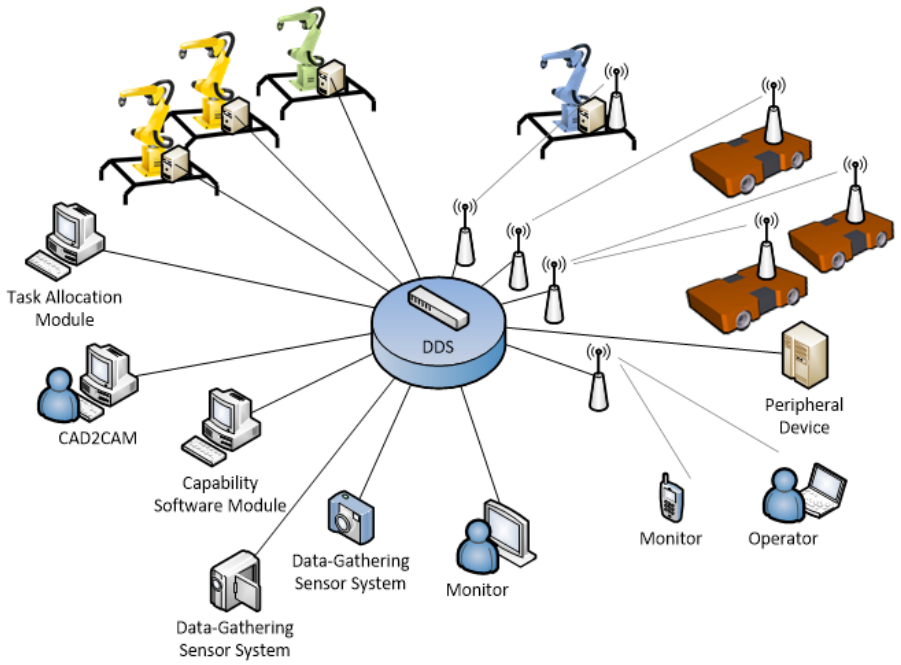

Figure 1.2: DDS as central information exchange platform

1.2 Problem Definition

For FRP part production, a robot arm can be used to load the press with raw material and unload the finished product. This allows for more flexibility in the production line than specialized low degrees of freedom (DOF) pick and place systems. For this task, a Fanuc 210F was obtained by SPC. The robot was delivered by IRSA Robotics. As the robot was not yet ready to carry out any task, setup and commissioning was necessary.

1.3 Objective

This thesis describes the setup, modelling and control of a 6DOF serial link industrial robot arm in a pick and place application for a production line. A combined kinematic and dynamic model of the Fanuc 210F is obtained. This model is then used to create a model-based control for the intended pick and place operation in the production line.

2. Literature Survey

In the project plan, it was stated, that "the master level will be demonstrated by understanding and simulating the dynamics of a 6 axis Robot arm." (see [55], sect. Master Level) This should be done by creating a model of the robot arm. This model can then be used to create a controller. To create the model of the robot arm, a literature review is necessary to lay out the best approach.

2.1 Methodology

To start the literature review, a set of first keywords was needed. Through an expert interview with the technical supervisor [33] who had already supervised other thesis projects in the domain of robotics, a list of keywords to start with was found in a quick discussion. Not all of these keywords were immediately clear, so it was necessary to find definitions for these. With the help of search engines and scientific databases, sources for these definitions could be found.

Keyword	Description	Source with search engine, website or database
6 axis robot	serial 6 degree of freedom robots	[52] with HANQuest
industrial robot arm	some form of jointed structure achieved by the linking of a number of rotary and/or linear motions or joint	[58] with Science Direct
inverse kinematics	Determination of joint variables in terms of the end-effector position	[27] with Springer Link search
Peter Corke robotics toolbox	Matlab toolbox for the study and simulation of classical arm-type robotics, for example such things as kinematics, dynamics, and trajectory generation	[13] with Google, yahoo, duckduckGo
motion planning	find a sequence of valid configuration of the joint that guides a robot towards a goal	[14] on website of Peter Corke
robot dynamics	relationship between the forces acting on a robot mechanism and the accelerations they produce	[22], with Scholarpedia
ROS	Robot Operating System - framework for writing robot software. It is a collection of tools, libraries, and conventions that aim to simplify the task of creating complex and robust robot behaviour across a wide variety of robotic platforms	in [41], About ROS, with Google ,yahoo, duckduckGo

Table 2.1: Keywords for literature review with definitions and sources

Spreading from these keywords, the literature can be extended with the help of HANQuest and Google scholar by the use of specific keyword combinations like "6DOF AND Industrial Robot AND Matlab model". Platforms like academia and researchgate help to find the right vocabulary for the field whilst also providing the papers and contact options with the authors. Some authors provide an extensive overview on their research, teachings and publications on their personal website. Wikipedia and other specialized wikis like scholarpedia break down the concepts and provide a summarizing view for different topics in the field of robotics. Individual phrases and keywords are best triangulated with different search engines like Google, DuckduckGo or yahoo to name a few. Databases like libgen and scihub are essential for scientific work, as not all universities can purchase access packages for all publishers. For applied knowledge, online manuals like the mathworks website or forums like

stackexchange and roboDK are a good source. Information about the robot model can be extracted from the manuals and other documents given by Fanuc.

2.2 Field of Study

As seen in [58], the FANUC 210F is an articulated robot arm, also called a jointed arm. It is a 6 axis robot that has six rotational joints, each mounted on the previous link. This type of robot has the ability to reach a point within the working envelope in more than one configuration or position with its final joint. As there are multiple configurations possible to reach the same position, path planning becomes an important topic. This means through inverse kinematics the motion of the joints needs to be determined without considering the local forces that cause them to move. As stated in the project plan, MATLAB was intended to be used. The robotics toolbox by Peter Corke was seen as a good tool for simulating these kinematics in MATLAB. When attaching the dynamics to this model, further simulations could be made to simulate the dynamic behaviour of the robot arm and to create a controller.

2.3 ROS

ROS is a flexible framework for writing robot software as seen on the "about ROS" page of the ROS-project. As pointed out by other engineers on the "ROS Answers" page, a project related forum [42], the role of ROS is not clearly defined. ROS shares characteristics with middleware, frameworks, but also has Operating System (OS)-like features. Robot Operating System (ROS) can take the role of a Data Distribution System (DDS), taking a central role in coordinating tasks and information on a distributed system of nodes.

2.4 Forward and Inverse Kinematics

As stated in the online manual of MathWorks, *Kinematics is the study of motion without considering the cause of the motion, such as forces and torques* [36]. Inverse kinematics is the logical opposite to Forward kinematics (See figure 2.1).
Mathworks [40] explains the relationship between forward and inverse kinematics. Source [46] gives a good overview on the topic of forward and inverse kinematics and gives an idea about the Denavit-Hartenberg notation, a convention to map the local coordinate systems within a kinematic chain as found in robot arms.
As seen in [38], chapter 2.2 , there are several other conventions besides the Denavit-Hartenberg (DH) notation used in the robotics research field like the the product of exponentials formulation . As most textbooks prefer a Denavit-Hartenberg (DH) formulation of the kinematics (see [38], ch. 3.1 Manipulation using single robots), this convention will be chosen in this work as well. The forward kinematic analysis can be obtained as seen in [10]. Also [61] gives a step by step guide how to determine the local coordinate frames for the links.
Solutions for forward kinematics are simple to obtain but solving inverse kinematics has been one of the main concerns in robot kinematics research. With more degrees of freedom (DOF), solutions get more complex as non-linear equations with transcendental functions need to be solved. For this set of inverse kinematics equations, no general algorithms (numerical solution) are available. Often algebraic, geometric and iterative methods for complex manipulators are used to find a solution to the inverse kinematic problem as stated by source [51].
To find a suitable method for solving the inverse kinematic problem, a definition for the solution is needed:

> *A manipulator will be considered solvable if the joint variables can be determined by an algorithm that allows one to determine all sets of joint variables associated with a given position and orientation. [...] The algorithm should find all possible solutions* [32]

With this definition of solvability, all systems with revolute joints and prismatic joints with 6 DOF in a single series chain are solvable with the current available research. [32]

As the goal is to find a suitable solution strategy for the inverse kinematic problem, it helps to map out the different types of methods. Solution strategies can be split into two classes as stated in [32]:

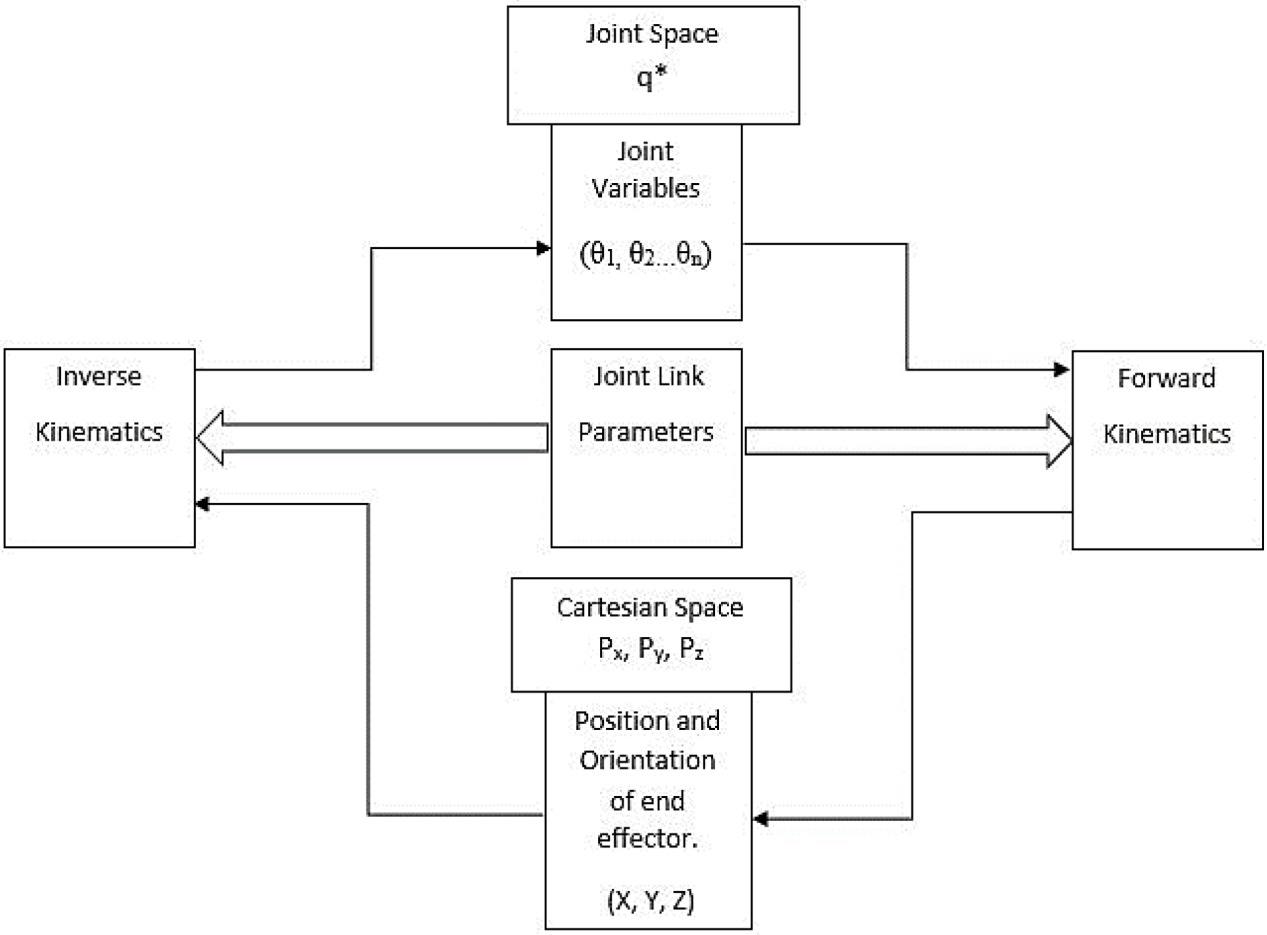

Figure 2.1: Relation between inverse and forward kinematics [26]

Closed form solutions	**Numerical solutions**
faster because analytical method	slower because of iterative nature
will find all solutions	can not always find all solutions
Two approaches:	
• algebraic approach	
• geometric approach	

Numerical solutions cannot always deliver all solutions and solve within an unknown number of operations. Also they depend on the users decision for accuracy [32], which is why a closed-form solution will be preferred.

Source [60] offers an algebraic method to solve the inverse kinematic problem for 6 axis robots. Other approaches are shown in [51] and [16].

As an alternative, geometric modelling can be done as seen in [9]. A completely different approach based on artificial neural networks is given in [3].
With one of these methods, a solution can be found for the inverse kinematic problem. This solution can then be verified with the robotics toolbox [13].

2.5 Dynamic Model

With this solution for the kinematics, a dynamic model of the robot can be created by attaching dynamics to the model as seen in [57] and [38]. A complete example of a dynamic simulation of a 6 DOF robot arm can be found in [14]. The dynamic model describes the behaviour of the manipulator with a set of equations, the equations of motion. The rigid-body equations of motion consist of five coupled differential equations in

matrix form. The Manipulator inertia matrix, Coriolis matrix and gravity term can be found with a Lagrangian formulation of the equations of motion for an open-chain manipulator as seen in [38] section 3.1. The force applied at the end-effector can be included as in [58], section 5.11, according to the principle of virtual work. A good overview on Coulomb and viscous friction can be found in [14], p. 252, section 9.1.2 .

2.6 Controller

A controller for trajectory tracking can then be created with the model as seen in [28]. This is a robot manipulator control based on plant inversion. In this thesis work, the controller will be based on [14] and [38]. This controller could then be plugged into the real robot as stated in [53], which aimsto achieve manipulator live control with Matlab. An example of this can be found in [25]. An overview on different control strategies for serial link manipulators can be found in [14], chapter 9 and [38], chapter 4.

3. Setup of the Physical Environment

The production line at Smart Production Centre (SPC) is planned to produce steering knuckles with Fibre Reinforced Plastics (FRP).

3.1 Steps in the Production Line

The planned production line consists of several steps that can be seen in figure 3.1. A visual representation from the simulation with Visual Components can be found in figure 3.2.

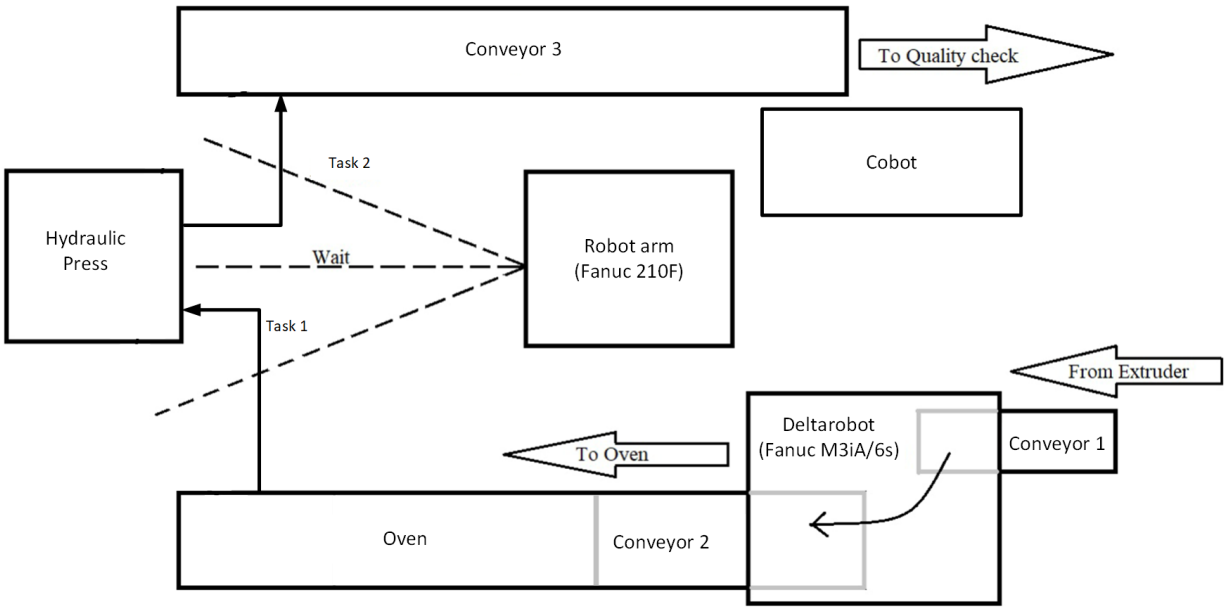

Figure 3.1: Schematic of the floorplan [34]

Extruder FRP raw material pellets are fed into an extruder. Alternatively, reprocessed material from steps down the production line is fed into the extruder. The extruder melts the raw material and releases it at its hot end. At the hot end of the extruder, the material is portioned into blops of 300g each by a guillotine.

Conveyor 1 The first conveyor belt transports the FRP material blops from the output of the Extruder to the Deltarobot. At the end of the coveyor belt is a proximity sensor to detect incoming material.

Deltarobot After a signal is given from the proximity sensor of Conveyor 1, a Fanuc M-3iA/6S 4-axis Deltarobot picks up the material and weighs it with a load cell built into the gripper of the robot. If the weight of the material is out of the desired range, it is discharged into a waste bin and recycled with the Extruder. If the weight of the material is within an acceptable range, it is placed on Conveyor 2. When placing the material on conveyor 2, the placement position and other data are communicated to processes further down the line. If needed, multiple blops of material can be stacked to the desired height while Conveyor 2 keeps moving.

Conveyor 2 The second conveyor belt transports the sorted and arranged FRP material from the Deltarobot through the Oven.

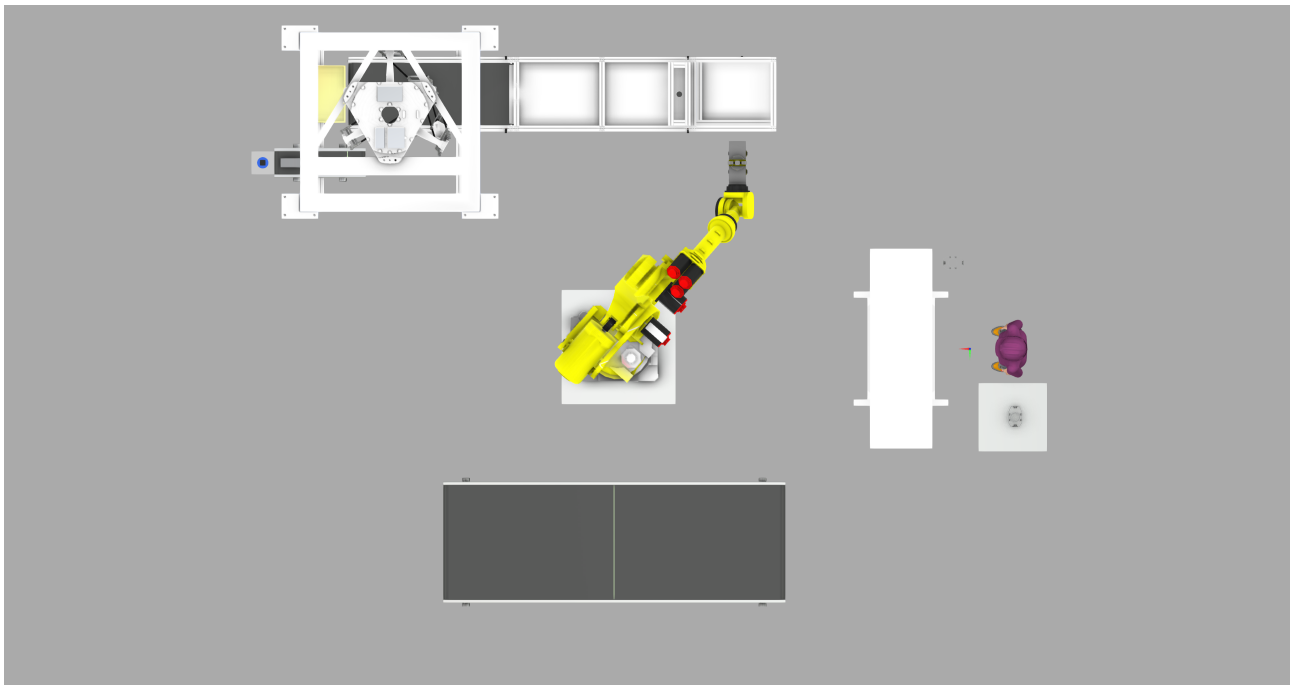

Figure 3.2: Top view of Production line layout, rendered with Visual Components [34]

Oven The oven takes in the blops of FRP material, and brings them up to the desired temperature with induction heaters. The material is then stored in a buffer zone separated with doors from the oven and the heating chamber where the material is kept at temperature. When the Hydraulic press is ready, a door in the buffer zone opens for the Robot arm to pick up the material.

Robot arm The robot arm has two tasks:

1. After a signal is given from the Hydraulic press, the Fanuc 210F 6axis robot arm picks up the hot FRP material stack from the buffer zone and places it in a Hydraulic press with a specialized gripper. After placing the part, the robot changes its gripper tool, and waits for its second task.

2. When the forming process in the Hydraulic press is finished, the mould opens, and the robot arm can use another gripper to pick up the FRP-part and place it on Conveyor 3. After placing the part, the robot changes its gripper tool, and waits for its first task

Hydraulic press After all material is placed and the Robot arm has completely withdrawn from the press in a safe distance, the press will close the heated mould to give the FRP material its desired shape with heat and pressure. After an estimated cycle time of 1 minute, the press opens and the finished part can be picked up.

Conveyor 3 The third conveyor belt transports the formed part from the press to the Cobot.

Cobot When the material arrives at the cobot, it is picked up, and presented to a camera system for analysation. If the part fulfils the desired properties, it is given to further processing steps and packing. In case, it does not fulfils the desired properties, it can be recycled with the Extruder.

3.2 Notes on Choice of Robot

The robot was chosen for this production line, as it provides a long reach. The arm without any End Of Arm Tooling (EOAT) can reach a full circle with a radius of $2655[mm]$ with its mounting plate. Also the overall weight of $1240[mm]$ allows for reasonably fast movements between $90 - 190[°/sec]$ depending on the axis while still giving a payload capacity of $210[Kg]$ at the mounting plate.[21] This high payload capacity is not needed directly, but it provides enough room for complex gripper designs, and provides room for future uses of the device.

3.3 Status of the Production Line at the Beginning of Project

At the start of this thesis work, the Hydraulic press was already fixed to the ground. Deltarobot,Conveyor 2, and the robot arm were roughly placed at their final positions. No gripper tools were available for either robots. The Extruder, Oven, Conveyor 1, Cobot and Conveyor 3 were not available. Electricity for the machines was not made available. A safety-fence was not set up.

3.4 Status of the Production Line at the End of Thesiswork

At the end of the project, the robots are positioned at a preferable position for their intended use. Interaction between press and robot arm were tested. Grippers were being developed in Computer Aided Design (CAD), and some standardized pneumatic grippers were tested on the robot arm for pick and place operations. Conveyor 2 was positioned, and fixed to the ground. Also, it was also tested with a Profinet-controlled variable frequency drive via Programmable Logic Controller (PLC). A heat resistant belt still needs to be mounted. An Extruder was delivered, but still needs to be commissioned, tested and modified. An Oven, Conveyor 1, Conveyor 3 and Cobot station are still missing. A preliminary safety fence with no electronic safeguard has been set up.

3.5 Set up Process of Robot Arm

3.5.1 Positioning

In order to integrate the robot arm into the production line, it needed to be mounted at its final position. This position was determined with a floorplan. To create the floorplan, the positions for the machines were worked out with two members of SPC responsible for the Deltarobot and the the Conveyor 2 with the Oven on top of it. At Qing, the production line was modelled in Visual Components (VC). With VC different positions for these machines could be tested. Simple pick and place programs could be simulated in that environment at Qing. These movements were inspected and varied to allow a smooth and fast transfer of material. Feedback and measurements for these simulations were provided. As the Hydraulic press was already fixed to the ground, all other positions were determined relative to the press. After estimating good positions via VC with respect to the movement range of the robot arm, the outlines of the machine feet were measured with a laser distance measurement system and marked on the floor. After placing the robots with a forklift by a member of the local fire brigade on the desired positions, the actual positions were measured again with the laser distance measurement system, documented in an excel sheet and tested again in VC. As the differences between actual and desired position were minimal, the simulations in VC did not show any problems. The robot arm and the other machines were then bolted down in the ground.

3.5.2 Commissioning

After the robot arm was positioned at its desired position, it could be plugged into the accompanying Fanuc R-30iA controller and was provided with electricity. This step was delayed a lot, as a safe and powerful enough electrical installation could not be provided by the Mobility Innovation Center (MIC) from the beginning of the project. After the robot and controller were running, the basic functions as found in the manuals could be tested. Manuals and configuration files were delivered by the previous owner, IRSA Robotics, on a set of Universal Serial Bus (USB)-sticks. These USB sticks had all kinds of manuals and configuration files for both robots without any context and order on them. An extensive search and ordering was needed to get the right manuals. Additionally, the previous owner had left a non-functional configuration on the robot, that was overcome by a complete factory reset with the help of Fanuc customer service. When these issues were overcome, the robots could be calibrated. As described in the manual, the robots have calibration markings. With these markings aligned, a calibration could be made with the iPendant.

3.5.3 Fieldbus and Network Setup

For the state of this thesis-work, the field bus and network- installation consists of following parts:

- PLC
- Human Machine Interface (HMI)-Display
- Variable Frequency controller (VFC) for conveyor belt
- R-2000 iA Mate Controller for Delta robot

- R-2000 iA Controller for Robot arm
- Several student notebooks for configuration, monitoring and control of the machines

Additionally following devices are Planned:

- Data Distribution System (DDS) (eventually consisting of several distributed nodes)
- cameras for process monitoring
- Controller for Cobot
- Controller for Extruder and Feeding system
- Part inspection system
- various sensors

The physical layer of the network and field bus is based on Ethernet. A basic connection to the PLC via Profinet as well as to the network via IP/Etherent at SPC was achieved.

The currently loaded configurations for network and field bus can be found in the iPendant.

3.6 Programming of Robot

For programming the robot, Fanuc's TP-language is used. The programming can either be done online with the iPendant or with an offline programming environment like Fanuc's Roboguide.

3.6.1 Online Programming

When programming the robot with the iPendant, the robot can be moved to the desired position. This position can then be directly saved into program code within the iPendant programming environment. More complex instructions like if/else, loops, or wait can be selected from a list provided in the iPendant programming environment. The iPendant can also be used to test programs written with an offline programming tool. It allows to simulate input states and run programs step by step in low speed to verify desired paths are taken.

3.6.2 Offline Programming

For offline-programming, programs can be created with various tools like Roboguide, a simple text editor or as in this project with VC.

Roboguide Roboguide is the offline programming environment provided by Fanuc. It comes with dynamic models of all Fanuc robots. Programs can be written and tested with these models. Besides that, the robot controller can be directly connected to a computer running Roboguide for configuration and program transfer. Program transfer to the controller can be done either with USB sticks, via FTP or via a Fanuc specific machine to machine communication protocol.

Visual Components VC is a software package that is used for layout planning, production simulation and offline programming. In figure 3.3 the toolchain for offline programming with VC can be seen. VC allows to simulate the robot together with other parts of the production line. This simulation can be used to program the robot in correspondence with all subsystems involved in the production line. The resulting TP program should then be loaded into Roboguide for verification, and if needed compiled. TP programs created with external tools can be compiled in Roboguide into ".ls" files that can be executed if no compiler is available on the robot controller. The R30iA controller then runs the program and supervises safe operation of the robot.

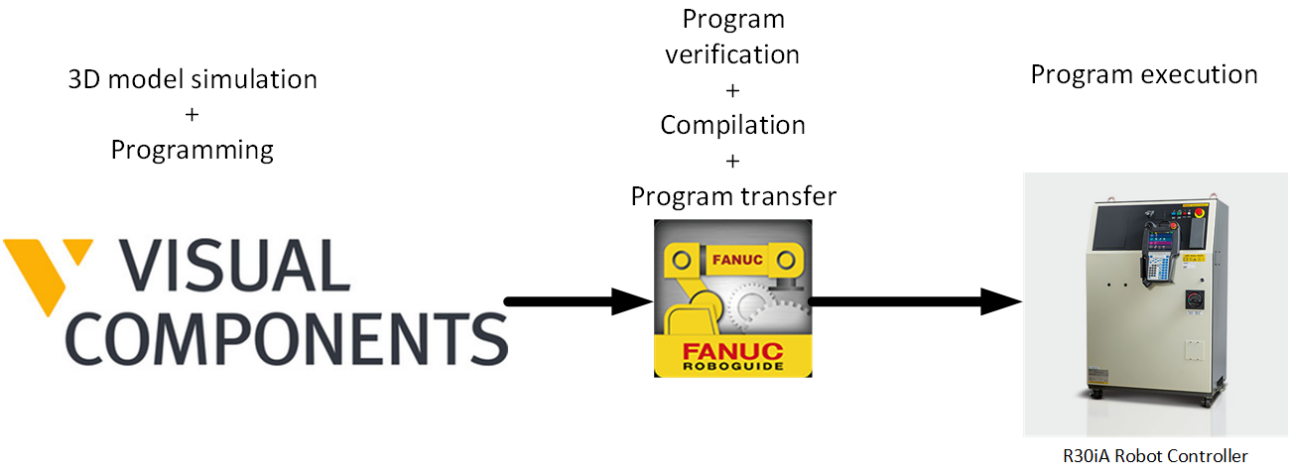

Figure 3.3: Steps for offline-programming with Visual Components

The simulation can be fed with live data from the running production line to create a digital shadow. On top of that, data can be exchanged via Open Platform Communications Unified Architecture (OPC UA) with a PLC that is connected to the robot controller. Through that, a high bandwidth communication with low latency can be established, that can be used to create a Digital Twin (DT).

3.6.3 Cooperation with Qing

Together with the company Qing, an event about DT was hosted at the SPC. In the preparation of this event, a Sequence for loading the press with the robot arm and unloading the finished product was developed. The pick and place sequence was created in Visual Components and then exported as a TP program. This program can be found in the appendix K TP Program.

3.6.4 Tasks Executed in the Machine Setup

Following tasks were executed in the machine setup process:

- Measuring in the factory hall for planning and verification of the positions
- Consulting Qing for good positioning of the machines
- Instructing the forklift driver for placing the robots
- Monitoring of safety on the factory floor
- Setting up parts of the electrical installation
- Setting up the robot controller with factory reset
- Testing of the robot controller and robot
- Requesting additional information, datasheets manuals and programs like Roboguide from Fanuc
- Configuring robot controller to the needs of SPC
- Developing TP program together with Qing
- Sifting through manuals and configuration files provided by IRSA Robotics and ordering them
- Providing help to the students setting up the oven and the PLC controlling the press
- Assisting other students for various tasks like moving equipment, program configuration or fitting of bearings
- Planning, ordering and installing the fieldbus and network equipment
- Consulting SPC in cybersecurity for network installation
- Cleaning the workshop regularly
- Presenting the production line with Qing and SPC at the event day

4. Building of a Model

4.1 Forward Kinematics

To model the movements of the end effector of a kinematic chain, a forward kinematic analysis is needed.

4.1.1 Five Step Approach

A serial-link manipulator consists of links in a chain connected by joints. For describing the serial-link mechanism geometry the Denavit-Hartenberg (DH) notation is one possible approach. It gives a description of a manipulator for kinematic solutions, Jacobians, dynamics, motion planning and simulation. This description can be obtained through a five step algorithm:[10]

1. Numbering of joints and links
2. Set up local coordinate reference frames
3. Establishing Denavit-Hartenberg (DH) parameters for each link
4. Calculate homogeneous transformation matrices
5. Compute Forward Kinematic Equations

4.1.1.1 Numbering of Joints and Links

A serial link robot with N joints has $N + 1$ links.

Numbering of Links The numbering scheme for links starts at (0) with the fixed grounded base and then increases sequentially up to (I) for the end effector.

Numbering of Joints The numbering scheme for joints starts at (1) with the joint connecting the first movable link to the base and then increases sequentially up to N.

Relation between Links and Joints Link (i) is connected to its

- lower link $(i - 1)$ at its proximal [31] end by joint (i)
- upper link $(i + 1)$ at its distal [31] end by joint $(i + 1)$

4.1.1.2 Set up Local Coordinate Reference Frames

With the DH convention, local coordinate frames can be attached to the links (i) and their accompanying joints $(i - 1)$. Each link (i) is described relative to the pose of the preceding link.

Assignation of z_i axes With the DH-notation, the z_i axes are assigned to link (i). By using the DH convention, both prismatic and revolute joints can be treated similarly:

revolute: z_i is the axis of revolution of joint $(i + 1) = n$

prismatic: z_i is the axis of translation of joint $(i + 1) = n$

This means, that joint n turns around axis z_i.

Direction of rotation With the direction of the z_i-axis, the direction of positive rotation around joint $(i+1)$ is also given by the right hand rule (see [56] and [47] at 10:35 for a visual representation). This means, the direction of positive rotation is counter-clockwise around the z_i-axis. With predictive choice of the direction of positive rotation, the direction of the z-axis can be chosen in order to minimize the DH-parameters.

Base frame The base frame $(i = 0)$ can be chosen nearly arbitrarily. The origin of the base frame can be any point on z_0. For simplicity, the origin of frame (0) can be put into joint $(n = 1)$. Usually, the x_0-axis of the base frame is chosen, so that it points in the direction of the end of arm tool in default position [61], but can be chosen in any convenient manner [48].

Assignation of frames (i) Starting from frame $(i = 1)$ in an iterative process, frame (i) can be set up using frame $(i - 1)$.
Three cases regarding the relationship of axes z_{i-1} and z_i need to be considered when setting up frames:

Non coplanar: Axes don't intersect and are not parallel. Line containing the common normal z_{i-1} to z_i defines x_i-axis and the point of intersection with z_i is the origin of frame (i), see C.1.

a) find common normal of the joint axes

b) put origin in intersection of normal with joint axis

c) put z_i axis in the joint axis

d) x_i points in direction of the common normal, facing away from frame $(i - 1)$

e) add y_i according to right hand rule

Parallel: Axes are parallel. Line containing the normal through origin of frame $(i + 1)$ and the z_i-axis defines the x_i-axis which is directed from origin of frame (i) toward the distal joint. The point of intersection of common normal with z_i-axis gives the origin of frame (i), see C.2.

a) look for normal originating from distal joint

b) put origin in the intersection of the normal with the joint axis

c) put the z_i axis in the joint axis of the link $(i + 1)$

d) x_i points follows the common normal with the distal joint, pointing towards the distal joint

e) add y_i according to right hand rule

Intersecting: Axes are intersecting. Line containing the normal to the plane formed by axes z_{i-1} and z_i gives the x_i-axis with positive direction chosen arbitrarily. Point of intersection of axes z_{i-1} and z_i is the origin of frame (i), see C.3.

a) put origin in intersection point of the axes

b) put the z_i axis in the joint axis of the link $(i + 1)$

c) x_i is perpendicular to both joint axes

d) add y_i according to right hand rule

Assignation of End Of Arm Tooling (EOAT) frame As there is no distal joint for the EOAT frame, the steps for this frame are different:

a) put origin on axis of proximal joint

b) axis z_i follows direction of z_{i-1}

c) x_i can be chosen arbitrarily, but is usually determined by screw holes

d) add y_i according to right hand rule

4.1.1.3 Establishing DH Parameters for Each Link

After assigning the local coordinate frames, the DH-Transformations to connect the coordinate frames have to be determined according to DH-convention. Further notes on the DH-convention can be found in appendix B DH-Convention.
The DH-parameters form the transformation matrices that connect serial link coordinate frames. They can be distinguished into parameters and variables. While constructive parameters are dependent on the construction of the robot and are constant, variables depend on the joint movement [51] [10] [61].

- α_i angle between z_{i-1} and z_i, measured in plane normal to x_i (constructive parameter)
- a_i distance between z_{i-1} and z_i, measured along x_i; parallel to $z_{i-1} \times z_i$ for intersecting axes (constructive parameter)
- d_i distance from the origin O_{i-1} of frame $i-1$ to the intersection of the x_i axis with z_{i-1}, measured along z_{i-1} (constructive parameter in revolute joints, variable in prismatic joints)
- θ_i angle between x_{i-1} and x_i, measured in plane normal to z_{i-1} (constructive parameter in prismatic joints, variable in revolute joints)

These parameters describe the relatively complex transformation of the coordinate system $i-1$ into the coordinate system i by four subsequent elementary transformations [46]:

$$T_{i-1,i} = ROT(z_{i-1}, \theta_i) \cdot TRANS(0,0,d_i)^T \cdot TRANS(a_i,0,0)^T \cdot ROT(x_i, \alpha_i) \tag{4.1}$$

$ROT(v, \alpha)$ is a simple rotation around vector v with the angle α and $TRANS(x,y,z)^T$ is a translational movement along the vector $[x,y,z]^T$. For a visual representation of these parameters see figure 4.1

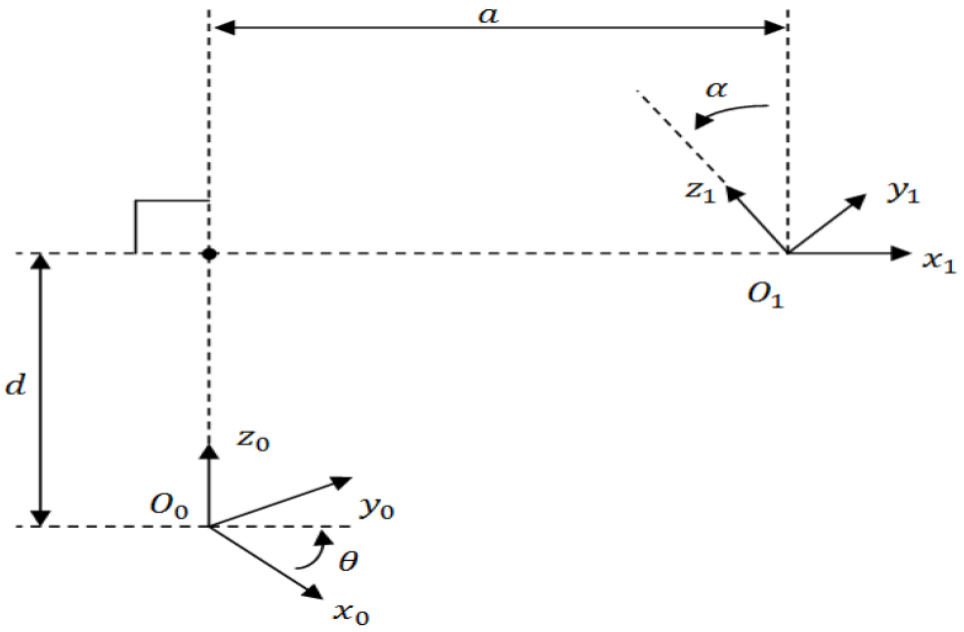

Figure 4.1: Visual representation of a DH-transformation

This visual representation gives a guide to attach the DH-parameters on serial-link mechanisms.

4.1.1.4 Calculate Homogeneous Transformation Matrices

Locating reference frame $(i-1)$ relative to (i) can be done by executing a series of transformations as given by equation 4.1. For simplification, they can be done with a 4×4 homogenous transformation matrix:

$${}^{i-1}T_i(\theta_i, d_i, a_i, \alpha_i) = R_z(\theta_i) \cdot T_z(d_i) \cdot T_x(a_i) \cdot R_x(\alpha_i) \tag{4.2}$$

Equation 4.2 can be expanded into:

$${}^{i-1}T_i = \begin{bmatrix} \cos\theta_i & -\sin\theta_i \cdot \cos\alpha_i & \sin\theta_i \cdot \sin\alpha_i & a_i \cdot \cos\theta_i \\ \sin\theta_i & \cos\theta_i \cdot \cos\alpha_i & -\cos\theta_i \cdot \sin\alpha_i & a_i \cdot \sin\theta_i \\ 0 & \sin\alpha_i & \cos\alpha_i & d_i \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{4.3}$$

4.1.1.5 Compute Forward Kinematic Equations

In Forward kinematics, the kinematic equations of a robot are used to compute the position of the EOAT from the given joint parameters.

Sum of transformations matrices With the transformation matrices from frame (i) to $(i-1)$, a 4×4 matrix for all N joints on I links can be established as in equation 4.4.

$$ {}^{0}T_{n} = \prod_{n}^{i=1} {}^{i-1}T_{i} \tag{4.4} $$

The resulting matrix has the form (see [60]):

$$ {}^{0}T_{n} = \begin{bmatrix} n_{xyz} & o_{xyz} & a_{xyz} & p_{xyz} \\ 0 & 0 & 0 & 1 \end{bmatrix} = \begin{bmatrix} n_x & o_x & a_x & p_x \\ n_y & o_y & a_y & p_y \\ n_z & o_z & a_z & p_z \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{4.5} $$

Short notation The short notations for rotation and position are defined as:

n_{xyz} normal vector

o_{xyz} orientation vector

a_{xyz} approach vector

p_{xyz} position

Transformation to Cartesian coordinates with Euler angles These vectors can be transformed into the $[x, y, z, \alpha, \beta, \gamma]$ notation. Vector $p = [p_x, p_y, p_z]$ gives $[x, y, z]$. The rotational approach $[\alpha, \beta, \gamma]$ can be calculated with the rotation matrix R as seen in 4.6.

$$ T = \begin{bmatrix} R & p \\ 0 & 1 \end{bmatrix} \tag{4.6} $$

The rotation matrix describes the relative orientation of two frames towards each other. As the columns $[n_{xyz}, o_{xyz}, a_{xyz}]$ are the unit vectors along the axes of one frame relative to the other reference frame, the relative orientation of frame b with respect to frame a is given by the rotation matrix as seen in 4.7 [10]

$$ {}^{b}_{a}R = \begin{bmatrix} {}_{a}x^{b} & {}_{a}y^{b} & {}_{a}z^{b} \end{bmatrix} = \begin{bmatrix} x^b \cdot x^a & y^b \cdot x^a & z^b \cdot x^a \\ x^b \cdot y^a & y^b \cdot y^a & z^b \cdot y^a \\ x^b \cdot z^a & y^b \cdot z^a & z^b \cdot z^a \end{bmatrix} \tag{4.7} $$

Transformation between frames The coordinates relative to the reference frame a, of a point p, of which the coordinates are known with respect to frame b with the same origin can then be calculated as in equation 4.8 (see [8], "Description of Position and Orientation) .

$$ {}_{a}p = {}^{b}_{a}R \, {}_{b}p \tag{4.8} $$

4.1.2 Forward Kinematic Equations of Fanuc 210F

4.1.2.1 Local Coordinate Reference Frames on the Fanuc 210F

As described in subsubsection 4.1.1.2 Set up Local Coordinate Reference Frames, the local coordinate reference frames can also be attached to the Fanuc 210F as seen in figure 4.2. For an overview how find the joints and links as well as the z-axis orientations see appendix E Joints and links. The assignment of individual frames can be found in appendix A Assignment of Individual Frames.

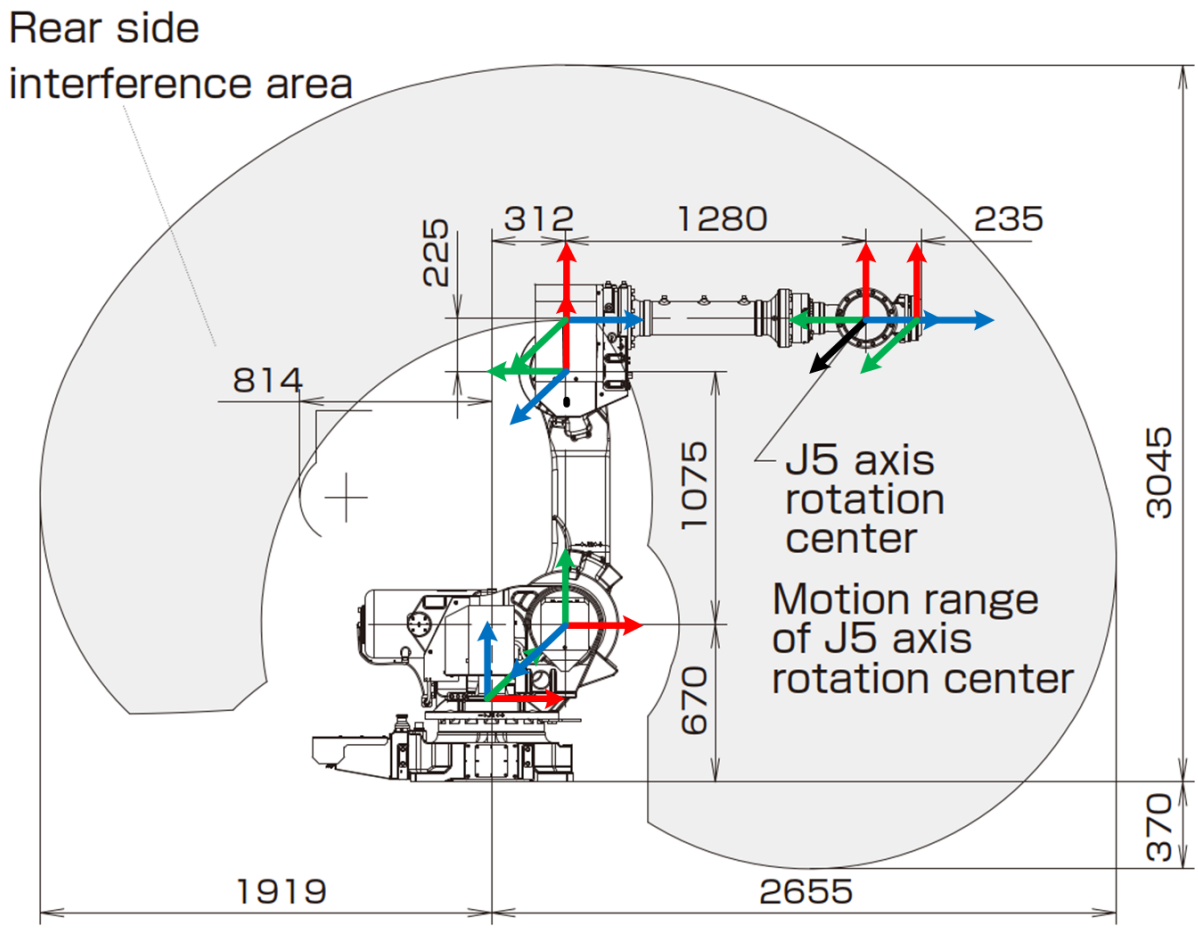

Figure 4.2: Coordinate reference frames for Fanuc 210F in standard (Normal) pose

4.1.2.2 Establishing DH Parameters on the Fanuc 210F

As described in subsubsection 4.1.1.3 Establishing DH Parameters for Each Link, the DH-Parameters can also be determined for the Fanuc 210F. These parameters can be found with the help of the assigned coordinate frames as seen in figure 4.3. For simplicity, only non-zero DH-parameters are shown in this figure.
To wrap this up, a summary of the D-H parameters for each link as follows from figure 4.3 can be found in Table 4.1 Denavit-Hartenberg (DH) Parameters for Fanuc 210F.

Link	θ_i	d_i	a_i	α_i
1	θ_1	d_1	a_1	α_1
2	θ_2	0	a_2	0
3	θ_3	0	a_3	α_3
4	θ_4	d_4	0	α_4
5	θ_5	0	0	α_5
6	θ_6	d_6	0	0

Table 4.1: Denavit-Hartenberg (DH) Parameters for Fanuc 210F

Numeric values of DH-parameters As can be seen from table 4.1, many DH-parameters are zero due to a good choice of pose and coordinate frames. All angles are plus or minus 90°.
Most of the frames have translational transformations in either a_i or d_i direction with link 1 and 5 being the exception. By lifting the base frame out of the joint, parameter d_1 could be set to zero as well (see Figure 4.3 Assignment of DH-Parameters on the Fanuc 210F). For better understanding, the origin of the frame was originally kept inside the joint (see Figure A.1 Coordinate reference frames for Fanuc 210F). Frame 5 shares the origin with frame 4, not requiring any translational movement. Most of the frames are also rotated by 90 degree around the x-axis with link 2 and 6 being the exception. Frame 6 was chosen with the same orientation as frame

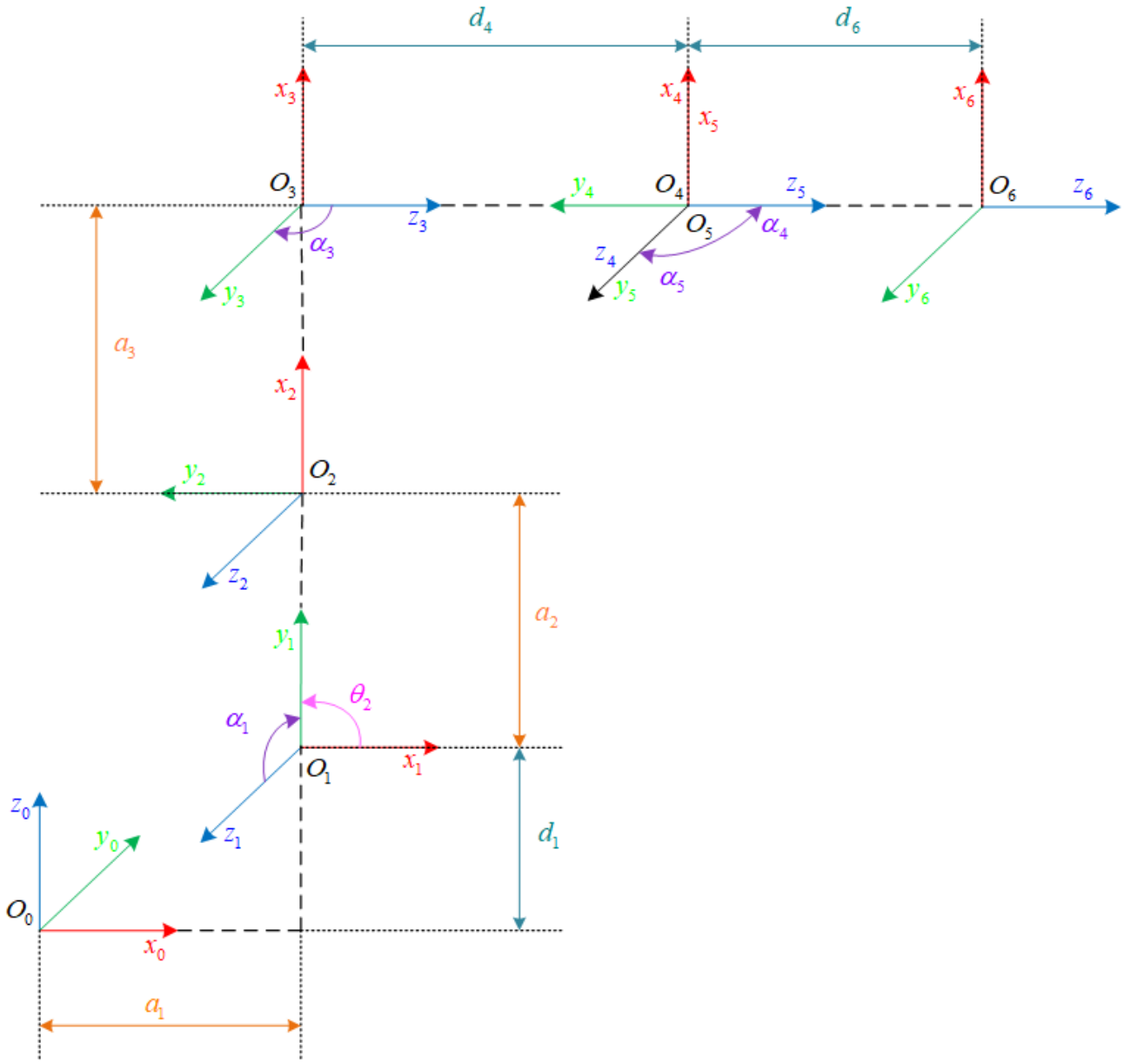

Figure 4.3: Assignment of DH-Parameters on the Fanuc 210F

5. Frame 2 has the same x-axis orientation as link 1 because joints two and three are parallel. There is just a rotational transformation around the z-axis of 90°. This is only due to the standard pose of the robot, as theta is a variable. For all coordinate frames theta is a variable, as all joints are revolute and none is translational.
Besides the angle values for α_i which were easy to determine from the graphical interpretation in figure 4.3, following numeric values need to be determined through measurements or CAD-Drawings:
d_1, d_4, d_6 and
a_1, a_2, a_3.
With figure 4.2 following DH-parameters can be obtained:

a_1= 312 $[mm]$

a_2= 1075 $[mm]$

a_3= 225 $[mm]$

d_4= 1280 $[mm]$

d_6= 235 $[mm]$

The parameter d_1 cannot be determined from this drawing. It can either be measured on the actual robot, or defined as zero, as frame zero can be freely moved along the z-axis. A value of zero would put the origin of

the base frame on the same height as frame 1, simplifying subsequent steps. It must be clear though, that all referencing in the forward and backward kinematics will be relative to this point. A change of this point of reference can be added later simply by adding another transformation matrix for translational movement along the z-axis as presented by Richard P. Paul [35].

This results in a table of numeric DH-parameters as seen in 4.2. The transformation ${}^{i-1}T_i$ depends only on a single variable θ_i as all of the other quantities stay constant, and all joints are revolute.

Link	θ_i	d_i	a_i	α_i
1	θ_1	0	312	$\pi/2$
2	θ_2	0	1075	0
3	θ_3	0	225	$\pi/2$
4	θ_4	1280	0	$-\pi/2$
5	θ_5	0	0	$\pi/2$
6	θ_6	235	0	0

Table 4.2: Numeric values of Denavit Hartenberg Parameters for Fanuc 210F

4.1.2.3 Calculate Homogeneous Transformation Matrices

For further details on how to set up $R_z(\theta_i), T_z(d_i), T_x(a_i) and R_x(\alpha_i)$ and form the transformation matrix in 4.3 see [35], chapter 1. The basic techniques for setting up the translation and rotation matrices as well as how to combine them are described in that chapter .

4.1.2.4 Forward Kinematic Equations on the 210F

As shown in fig.E.2, the Fanuc 210F is a 6-degrees of freedom (DOF) robotic arm with N=six revolute joints in series. Their orientation can be expressed in short form with

$$(R \perp R \parallel R \perp R \perp R \perp R). \tag{4.9}$$

With this, the complete transformation matrix can be derived in the form as described in subsubsection 4.1.1.5 Compute Forward Kinematic Equations. The intermediate steps, that lead to the forward kinematic equations, can be found in appendix G Intermediate Steps in Forming Forward Kinematic Equations. With matrix 4.5, the forward kinematic equations can be formed:

$$
\begin{aligned}
&n_x = \\
&\cos\theta_1(\cos(\theta_2+\theta_3)(\cos\theta_4\cos\theta_5\cos\theta_6-\sin\theta_4\sin\theta_6)-\sin(\theta_2+\theta_3)\sin\theta_5\cos\theta_6) \\
&\quad+\sin\theta_1(\sin\theta_4\cos\theta_5\cos\theta_6-\cos\theta_4\sin\theta_6) \\
&n_y \\
&= \\
&\sin\theta_1(\cos(\theta_2+\theta_3)(\cos\theta_4\cos\theta_5\cos\theta_6-\sin\theta_4\sin\theta_6)-\sin(\theta_2+\theta_3)\sin\theta_5\cos\theta_6) \\
&\quad-\cos\theta_1(\sin\theta_4\cos\theta_5\cos\theta_6-\cos\theta_4\sin\theta_6) \\
&n_z \\
&= \\
&\sin(\theta_2+\theta_3)(\cos\theta_4\cos\theta_5\cos\theta_6-\sin\theta_4\sin\theta_6)-\cos(\theta_2+\theta_3)\sin\theta_5\cos\theta_6 \\
&o_x \\
&= \\
&\cos\theta_1(-\cos(\theta_2+\theta_3)(\cos\theta_4\cos\theta_5\sin\theta_6+\sin\theta_4\cos\theta_6)-\sin(\theta_2+\theta_3)\sin\theta_5\sin\theta_6) \\
&\quad-\sin\theta_1(\sin\theta_4\cos\theta_5\sin\theta_6-\cos\theta_4\sin\theta_6) \\
&o_y \\
&= \\
&\sin\theta_1(-\cos(\theta_2+\theta_3)(\cos\theta_4\cos\theta_5\sin\theta_6+\sin\theta_4\cos\theta_6)-\sin(\theta_2+\theta_3)\sin\theta_5\sin\theta_6) \\
&\quad+\cos\theta_1(\sin\theta_4\cos\theta_5\sin\theta_6-\cos\theta_4\sin\theta_6) \\
&o_z \\
&= \\
&\quad-\sin(\theta_2+\theta_3)(\cos\theta_4\cos\theta_5\sin\theta_6+\sin\theta_4\sin\theta_6)-\cos(\theta_2+\theta_3)\sin\theta_5\sin\theta_6 \\
&a_x \\
&= \\
&\cos\theta_1(\cos(\theta_2+\theta_3)\cos\theta_4\sin\theta_5+\sin(\theta 2+\theta_3)\cos\theta_5)+\sin\theta_1\sin\theta_4\sin\theta_5 \\
&a_y \\
&= \\
&\sin\theta_1(\cos(\theta_2+\theta_3)\cos\theta_4\sin\theta_5+\sin(\theta 2+\theta_3)\cos\theta_5)+\cos\theta_1\sin\theta_4\sin\theta_5 \\
&a_z \\
&= \\
&\sin(\theta_2+\theta_3)\cos\theta_4\sin\theta_5-\cos(\theta_+\theta_3)\cos\theta_5 \\
&p_x \\
&= \\
&\cos\theta_1(a_1+a_2\cos\theta_2+a_3\cos(\theta_2+\theta_3)+d_4\sin(\theta_2+\theta_3)+d_6(\cos(\theta_2+\theta_3)\cos\theta_4\sin\theta_5+\sin(\theta_2+\theta_3)\cos\theta_5)) \\
&\quad+d_6\sin\theta_1\sin\theta_4\sin\theta_5 \\
&p_y \\
&= \\
&\sin\theta_1(a_1+a_2\cos\theta_2+a_3\cos(\theta_2+\theta_3)+d_4\sin(\theta_2+\theta_3)+d_6(\cos(\theta_2+\theta_3)\cos\theta_4\sin\theta_5+\sin(\theta_2+\theta_3)\cos\theta_5)) \\
&\quad+d_6\sin\theta_1\sin\theta_4\sin\theta_5 \\
&p_z \\
&= \\
&a_2\sin\theta_2+a_3\sin(\theta_2+\theta_3)-d_4\cos(\theta 2+\theta_3)+d_6(\sin(\theta_2+\theta_3)\cos\theta_4\sin\theta_5-\cos(\theta_2+\theta_3)\cos\theta_5)
\end{aligned}
\tag{4.10}
$$

In these equations, the numerical values were replaced with their parameter name. These numeric values can be found in section 4.1.2.2.
These kinematic equations can be used to simulate the movements of a robot with given angle-values for the joints.

4.2 Inverse Kinematics

With the forward kinematic equations, the position of the end effector can be determined relative to the base frame for given values of the joint variables. In the context of robotics, it is also necessary to determine these joint variables for a given end effector position. This is referred to as the inverse kinematic problem. Solving the inverse kinematic problem for a given serial link actuator allows to transform a motion plan of the EOAT into joint actuator trajectories for the robot.

4.2.1 Existence of Solutions

To determine the joint values for a EOAT-position it needs to be determined, if there exists a solution. Not all positions are equally reachable. There are two types of workspace([15], ch4, p.102):

dextrous workspace volume of space that the robot end-effector can reach with all orientations

reachable workspace volume of space, that the robot can reach in at least one orientation

A manipulator with less than 6DOF cannot reach all positions and orientations in 3D-space. Further limitations can be imposed by the axis alignment. A planar manipulator for example cannot reach out of the plane. [15]

4.2.2 Multiple Solutions

A 6DOF robot has multiple sets of inverse solutions. To understand why there are multiple solutions, it helps to have a look at fig. 4.4. The same end effector position and orientation can be reached in two ways by a 3-link manipulator.

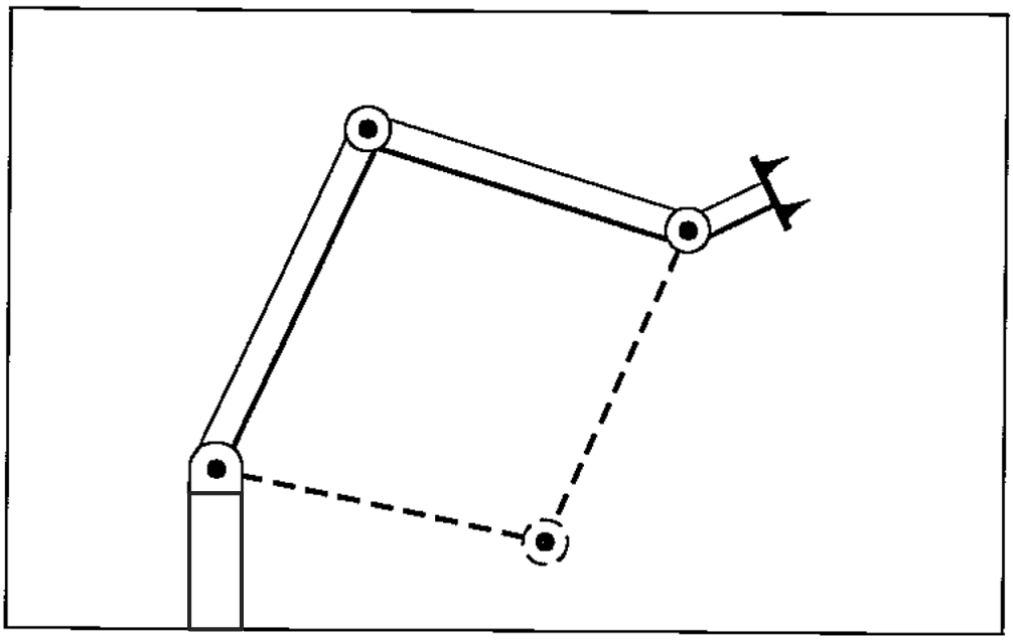

Figure 4.4: Three link manipulator with 2 solutions (see [15], p.103)

This can be extended to the 6DOF serial link manipulators. As stated by YanWu et al. there are 8 groups of inverse solution for most EOAT-positions within the manipulator's workspace [60].

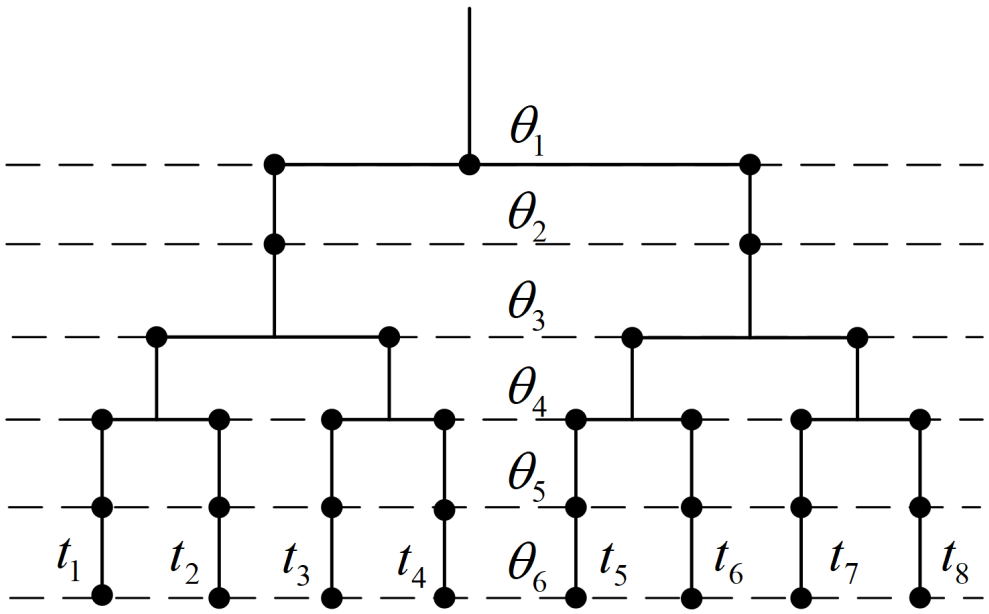

Figure 4.5: Tree of inverse solutions ([60], fig.2)

The tree of inverse solutions shown in figure 4.5 is a schematic representation of different possible solutions. It does not show actual positions. A set of the different joint configurations for inverse solutions in the Fanuc can be found in appendix D Robot Configurations.

4.2.3 Inverse Solution of 6 DOF Robot

As shown in chapter 4.1.1.5 the end effector position is defined by four 3×1 vectors $[n_{xyz}, o_{xyz}, a_{xyz}, p_{xyz}]$. With these known, the joint variables $(\theta_1, \theta_2, \theta_3, \theta_4, \theta_5, \theta_6)$ can be calculated. This problem is solved by the method of using the inverse transformation of unknown link pre-multiplication as presented by Yan Wu et al. [60]. This solution works only for 6 DOF serial link mechanisms with spherical wrist. For a distinction between spherical and nonspherical wrists see appendix **?? ??**.

Forward kinematic equation For a 6DOF Robot, the forward kinematic matrix equation from 4.4 and 4.5 can be reformulated into 4.11.

$$ {}^0_6T = \begin{bmatrix} n_x & o_x & a_x & p_x \\ n_y & o_y & a_y & p_y \\ n_z & o_z & a_z & p_z \\ 0 & 0 & 0 & 1 \end{bmatrix} = {}^0_1T(\theta_1){}^1_2T(\theta_2){}^2_3T(\theta_3){}^3_4T(\theta_4){}^4_5T(\theta_5){}^5_6T(\theta_6) \tag{4.11} $$

4.2.4 The Inverse Solution

With $[n_{xyz}, o_{xyz}, a_{xyz}, p_{xyz}]$ known, the angle values of the joint variables can be determined by separation of the joint variables. As stated in section 4.2.2, there are two solutions for each θ_i. Steps on how to obtain these equations can be found in appendix H Intermediate Steps in Forming Inverse Kinematic Equations.

θ_1:

$$ \theta_{1_1} = \text{atan2}(p_y - a_y d_6, p_x - a_x d_6) \qquad \theta_{1_2} = \text{atan2}(-p_y + a_y d_6, -p_x + a_x d_6) \tag{4.12} $$

θ_2:

$$ \theta_{2_{1/2}} = \text{atan2}(-v, u) \pm \text{atan2}(\sqrt{v^2 + u^2 - m^2}, m) \tag{4.13} $$

θ_3:

$$ \theta_{3_{1/2}} = \text{atan2}(-d_4, a_3) \pm A\tan(2(\sqrt{d_4^2 + a_3^2 - h^2}, h)) \tag{4.14} $$

θ_5:

$$\theta_{5_{1/2}} = \pm \operatorname{acos}[(-c_1 s_2 a_x - s_1 s_2 a_y - c_2 a_z)c_3 - (c_1 c_2 a_x + s_1 c_2 a_y - s_2 a_z)s_3] \tag{4.15}$$

θ_4:

$$\theta_{4_{1/2}} = \pm \arccos[\frac{c_1 c_2 a_x + s_1 c_2 a_y - s_2 a_z + s_3 c_5}{-c_3 s_5}] \tag{4.16}$$

θ_6:

$$\theta_{6_{1/2}} = \operatorname{atan2}(s_1 n_x - c_1 n_y, s_1 o_x - c_1 o_y) \pm \operatorname{atan2}(\sqrt{(s_1 n_x - c_1 n_y)^2 + (s_1 o_x - c_1 o_y)^2 - c_4^2}, c_4) \tag{4.17}$$

With 4.12,4.13,4.14,4.16,4.15 and 4.17 the values for $(\theta_1, \theta_2, \theta_3, \theta_4, \theta_5, \theta_6)$ can be calculated for all reachable end effector positions.

4.2.5 Optimization of Inverse Kinematic Solution

As seen in subsection 4.2.1 Existence of Solutions there exist eight sets of inverse solutions in the dextrous workspace. The optimal set of solutions can be found through an optimization function. This optimization function needs to find the set of solution that can reach the desired position fastest from the current position.

Cost function Equation 4.18 is the cost function to be minimized.

$\theta_i(k+1)$: target angle

$\theta_i(k)$: departing angle

k_i: power value

$$err = \min \sum_{i=1}^{6} k_i[\theta_i(k+1) - \theta_i(k)] \tag{4.18}$$

As can be seen, this cost function minimizes the sum of angles over all axes for the movement of the end effector from position $[n_i, o_i, a_i, p_i]$ to $[n_{i+1}, o_{i+1}, a_{i+1}, p_{i+1}]$ by using the inverse solution presented in subsection 4.2.4 The Inverse Solution.

Limitations of cost function This cost function not necessarily gives the fastest path, as it does not account for the actual movement speed of the different axes. A possible solution for this would require multiplying with a cost factor between 0 and 1 for each joint (1 for the slowest joint) with the respective angle error to account for the difference in angular velocity.

Description of implementation An implementation of this function receives the current position either in joint angles or Cartesian coordinates of the end-effector which could be given in form of a rotation matrix as seen in 4.5 or in form of euclidean space $[x, y, z, \alpha, \beta, \gamma]$ which could be transformed by the inverse of 4.6 into a rotation matrix.
This algorithm would then start the calculation to get all possible paths in the tree of inverse solutions (see figure 4.5) for position $i+1$.

Algorithm for minimal computational effort To minimize the computational effort, following algorithm is proposed by Yan Wu et al. [60]:

1. The calculation starts with θ_1 by finding the two solutions for equation 4.12.
2. When calculating θ_2:
   (a) The term $v^2 + u^2 - m^2$ should be > 0, otherwise the program can terminate the calculation of the subsequent steps of this group, return $err = inf$ or NaN , and provide a variable like $P = 1$ to determine the cause for the abortion of subsequent steps
   (b) If there is a solution for θ_2 it needs to be tested for extraneous roots

3. If the calculation has been continued, θ_3 can be determined. Again a test for extraneous roots is necessary
4. When calculating θ_5, test for extraneous roots
5. When calculating θ_4, test for extraneous roots
6. calculate θ_6
7. determine error as seen in 4.18 for all possible paths
8. find $\min(err(t_1^8))$

The resulting set of angle values is the output of the algorithm. It is the unique inverse solution, determined easiest to reach with the given cost function in 4.18.

4.3 Jacobian

When the position of the end-effector is $p = (x, y, z, \alpha, \beta, \gamma)$ and the joint angles are $q = (q_1, q_2, q_3, q_4, q_5, q_6)$, the derivative of p with respect to the joint variables q gives the Jacobian matrix as seen in 4.19.

$$\frac{dp}{dq} = J(q) \tag{4.19}$$

As a result, the Jacobian connects the velocities of the two state-spaces as seen in equation 4.20 (see [14] p.231, sec. 8.1). Further derivations like acceleration, Jerk, Snap an Crackle can be obtained similarly by dividing both sides through dt.

$$\dot{p} = J(q)\dot{p} \tag{4.20}$$

As the Jacobian matrix J connects the end-effector space in generalized coordinates with the joint space in angle coordinates, it can relate also forces between the state-spaces according to the principle of virtual work (see [58] page 157, section 5.11).
These are a few notable conversions, the Jacobian provides:
$\dot{x} = J(\theta)\dot{\theta}$
$\theta = J^{-}1x$
$\dot{\theta} = J^{-1}\dot{x}$
$\ddot{\theta} = J^{-1}\ddot{x} + \frac{d}{dt}(J^{-1})\dot{x}$

4.4 Dynamics

With the kinematic model in place, dynamics can be attached. The motion of the EOAT is a result of the motion of each link, which are moved by torques applied by the joints. These torques have to counteract friction, gravity and varying inertias. With the rigid-body equations of motion, the robot is described by a set of coupled dynamic equations. These give the joint torques necessary to move to and hold a certain pose.

4.4.1 Inverse Dynamics

For a serial link manipulator, the equations of motion can be expressed as a set of coupled differential equations in matrix form as seen in equation 4.21. These equations describing the manipulator rigid-body dynamics are called inverse dynamics. The inverse dynamics is described by following terms:

M Joint-space inertia matrix

C Coriolis and centripetal coupling matrix

F Friction force

G Gravity loading

W Wrench applied at the end effector

J Jacobian of the robot

The equations are given in generalized joint coordinates q, velocities $\dot{q}$ and accelerations $\ddot{q}$. The result is the vector Q of generalized actuator forces associated with the generalized coordinates q, that can be used to control the robot.

$$Q = M(q)\ddot{q} + C(q,\dot{q})\dot{q} + F(\dot{q}) + G(q) + J(q)^T W \tag{4.21}$$

4.4.2 Derivation of the Equations of Motion

The equations of motion can be derived with a Lagrangian approach as seen in [38], chapter 3.

4.4.2.1 Lagrange Equations for an Open-Chain Manipulator

The equations of motion for an open-chain robot can be formed by substituting the Lagrangian into the Lagrange equations. The Lagrange equations in vector form can be seen in eq. 4.22. They are given for a mechanical system with generalized coordinates q and with Υ representing the actuator torque.

$$\frac{d}{dt}\frac{\partial L}{\partial \dot{q}} - \frac{\partial L}{\partial q} = \Upsilon \tag{4.22}$$

With the Lagrangian in place, expanding the terms of partial derivatives and rearranging, these terms form 4.21 as seen in [38], section 3.2.

Lagrangian for an open-chain robot The Lagrangian L can be calculated with the potential $V(q)$ and kinetic $T(q,\dot{q})$ energy as seen in 4.23.

$$L(q,\dot{q}) = T(q,\dot{q}) - V(q) \tag{4.23}$$

Kinetic energy of the links For an open chain robot, the total kinetic energy can be defined as in 4.24.

$$T(\theta,\dot{\theta}) = \sum_{i=1}^{n} T_i(\theta,\dot{\theta}) = \frac{1}{2}\dot{\theta}^T M(\theta)\dot{\theta} \tag{4.24}$$

Potential energy of the links The total potential energy is defined similarly as in 4.25.

$$V(\theta) = \sum_{i=1}^{n} V_i(\theta) = m_i \cdot g \cdot h_i(\theta) \tag{4.25}$$

Forming the Lagrangian Substituting these into the Lagrangian gives 4.26.

$$L(\theta,\dot{\theta}) = \sum_{i=1}^{n}(T_i(\theta,\dot{\theta}) - V_i(\theta)) = \frac{1}{2}\dot{\theta}^T M(\theta)\dot{\theta} - V(\theta) \tag{4.26}$$

4.4.2.2 Inertia Matrix

"The joint-space inertia is a positive definite, and therefore symmetric, matrix". [14], section 9.2.2
The manipulator inertia matrix can be calculated with the link inertia Matrix and the Jacobian as seen in 4.27.

$$M(\theta) = \sum_{i=1}^{n} J_i^T(\theta) M_i J_i(\theta) \tag{4.27}$$

Link inertia matrix The link inertia Matrix can be created as in 4.28 with the Inertia Tensor $\mathcal{I}$and the link mass m.

$$M = \begin{bmatrix} mI & 0 \\ 0 & \mathcal{I} \end{bmatrix} \tag{4.28}$$

Inertia tensor The inertia Tensor $\mathcal{I}$ can be approximated with homogeneous simple forms like the cuboid or a cylinder as given in [44]. An example for the inertia Tensor can be found in [38].

Inertia matrix example An example for the inertia matrix is given in 4.29 for a homogeneous bar with width w, height h and length k. Other geometric shapes and non-homogeneous structures have different inertia matrices.

$$M = \begin{bmatrix} m & 0 & 0 & 0 & 0 & 0 \\ 0 & m & 0 & 0 & 0 & 0 \\ 0 & 0 & m & 0 & 0 & 0 \\ 0 & 0 & 0 & \frac{m}{12}(w^2+h^2) & 0 & 0 \\ 0 & 0 & 0 & 0 & \frac{m}{12}(l^2+h^2) & 0 \\ 0 & 0 & 0 & 0 & 0 & \frac{m}{12}(l^2+w^2) \end{bmatrix} \tag{4.29}$$

4.4.2.3 Coriolis Matrix

The Coriolis matrix can be calculated with 4.30 (see, [38], sec. 4.3, p. 172)

$$C_{ij}(\theta,\dot{\theta}) = \sum_{k=1}^{n} \Gamma_{ijk}\dot{\theta}_k = \frac{1}{2}\sum_{k=1}^{n}(\frac{\partial M_{ij}}{\partial \theta_k} + \frac{\partial M_{ik}}{\partial \theta_j} - \frac{\partial M_{kj}}{\partial \theta_i})\dot{\theta}_k \tag{4.30}$$

4.4.2.4 Friction

The friction can be determined through friction models as given in [14], section 9.1.2. As friction models are computationally very intensive, they will not be subject to this simulation of a robot arm.

4.4.2.5 Gravity Term

As $V(\theta)$ is the potential energy due to gravity, the forces acting on the joint due to gravity can be determined by $dV/d\theta_i$.

4.4.2.6 Wrench at Endeffector

A wrench W is a combination of translational force (f) and rotational moment (m) into a vector with six elements:
$W = (f_x, f_y, f_z, m_x, m_y, m_z)$ (see [14] page 69, section 3.2.2)
This wrench works on the endeffector, so it is given in the Cartesian workspace. As seen in section 4.3 Jacobian, these forces can be directly transformed into joint space with the Jacobian.

4.4.3 Forward Dynamics

To simulate the effects of the joint torques on the robot, the equations of motion 4.21 can be rearranged as seen in 4.31. This gives the resulting joint accelerations $\ddot{q}$ which can be used to calculate the movements of the links.

$$\ddot{q} = M^{-1}(q)(Q - C(q,\dot{q})\dot{q} - F(\dot{q}) - G(q) - J(q)^T W) \tag{4.31}$$

4.5 Matlab Implementation

With the help of the Robotics toolbox for Matlab, the robot can be implemented in Matlab as seen in appendix I Matlab Code.

4.5.1 Guideline to the Toolbox

The Robotics toolbox for Matlab, created by Peter Corke can be found on his website petercorke.com. This website gives an extensive manual how to install and use the toolbox and is best downloaded with the latest installation of the toolbox. The installation from a *.mltbx* file has been chosen. The accompanying book "Robotics Vision and control" gives the background information in physics and explains the examples delivered with the installer. Notable information on the toolbox can be found in the handbook "Robotics Toolbox for Matlab" that can be found in its latest release on the website. For better performance, part of the inverse Dynamics can be compiled locally in a faster language. Notes on that can be found in [14], page 266, "fastRNE".

4.5.2 Quick Start Guide to the Implementation

The implementation of the Fanuc210F in Matlab consists of several files:

Model_210F.m builds the Robot model and provides some tests to examine the behaviour

ComputedTorqueControl.slx Simulink file, that uses the robot model for a Computed Torque Control application

WorkspaceControl.slx Simulink file, that uses the robot model for a Workspace Control application

pickAndPlaceSequence.slx Simulink file, that uses the robot model for a pick and place operation with Workspace Control

As the .m file loads the robot model into work space, it needs to be run first. The program will pause several times, and asks via command line output to continue via enter. Then, the simulation in Simulink can be run. A copy of these files will be delivered with the written thesis-report.

4.5.3 Walkthrough on the Impelmentation

The Fanuc210F was implemented in Matlab with the help of [14] in Matlab. In that process, several decisions and simplifications had to be made. Subsequently, a walk-through of this implementation will be given.

4.5.3.1 Definition of Robot Model

The file Model_210F.m contains the definition of the Robot object **R**. Following steps are taken to create **R**.

Standard positions For simplicity of further work, four standard positions are provided:

qz Zero Angle

qr Ready, straight and vertical

qs Stretch, straight and horizontal

qn Nominal, arm is in a dextrous working pose

Kinematic definiton of links With the DH-parameters provided in table 4.2, the links are defined. As the toobox provides an offset parameter, link nr. 2 has an angular offset of $\frac{\pi}{2}$ to have the robot in the normal pose as seen in figure 4.2 at zero angle. Additionally, an estimation of the rotational freedom for the joints is estimated with the help of the data-sheet [21]. All these parameters are loaded into link-objects provided by the toolbox.

Definiton of manipulator With the **SerialLink** constructor, these link objects are combined into the robot arm, named "**R**". Additionally to the links, the base and a tool are defined. After running some test commands like testing the joint types, the kinematic definition is finished.

Testing of kinematics After the kinematic model is set up, the kinematic abilites of this model are tested. Forward kinematics can be simply tested by using the standard positions, and observing these with the plot command, provided with the toolbox, which gives a graphical representation of the manipulator. For the inverse kinematics, different solvers are available. The **ikine6s** function, an analytic solver has shown problems, as the robot does not take the intended positions, as can be observed in the graphical output. This is probably due to a bug in the implementation. Other solver functions like the **ikcon** function proved to be reliable in finding joint configurations for a desired endpoint coordinate within the workspace. If the position is outside of the workspace, and not reachable can be observed by the exit flag.

Trajectories With the **jatraj** and **ctraj** functions, paths can be created, either as shortest angular joint distance or as a straight endeffector line in cartesian space.

Dynamics definition One way to get the parameters, associated with the dynamics of the manipulator would be taking apart the robot, as it was done for the p560, delivered as a model with the toolbox. This is a very time- and labour intensive work which is expected to give the best results in accuracy.
Another way to receive the parameters would be the measurement of the electric power going into the joint motors. With the knowledge of a set of rudimentary parameters like the kinematic structure, and the parameters of the electric motors, other parameters could be discovered with techniques from systems identification. This process is time-and labour intensive as well. On the other hand it is minimally invasive, can be automated and once set up, can be applied to other robots easily.
These two techniques would have provided a very precise model of the robot. As they are too labour intensive, they will not be used in this work.

Innertia matrices To get a model of the robot quickly with the data available, the data-sheet [21] provides all necessary data. The whole structure is assumed to be of similar density. The links can be assumed as simple geometric structures like cylinders or cuboids. With the length, width and height of each link found in the data-sheet or estimated from the drawing, the volumes of the links can be calculated based on the volume model that best fits them. With the total mass of the machine is given in the data-sheet, the weight can be distributed by volume. Based on their shape, inertia tensors can be defined.

Center of mass With the drawings and data from the data-sheet, also the centre of masses can be put in the middle between the joints.

Friction and motor parameters Values for several parameters are hard to obtain or estimate. Because of this, the corresponding parameters from the Puma 560, a model delivered with the toolbox are taken. These parameters are:

- link friction
- motor viscous friction
- gear ratio
- motor inertia

The Puma 560 is a similarly sized robot as the 210F. These parameters will mostly not be used, but simply help to populate the model with values because the friction model is too computationally intensive and is spared from most dynamic simulations.

Payload As a robotic gripper was designed by another student, an estimation for weight and center of mass could be included in the model. According to the student responsible for the gripper design, the total mass of the gripper would not exceed 25Kg and would have a maximum length of 0.658m.

Testing of Dynamics To test the dynamics, the forces to move between positions and holding them can be calculated with the **rne** function. With the **fdyn** function, a collapse due to gravity with no forces applied at the joints can be simulated an observed.

4.5.3.2 Known Problems

Several problems have been identified with the toolbox. These are in the "ikine6s" function, the simulink block "tr2delta" and the ikine function.

ikine6s does not deliver correct solutions. By observing the graphical output in the program part "Inverse Kinematics", it can be seen, that the endeffector positions for all 8 solutions differ. The cause could not be identified. As an alternative, ikine suffices for most applications.

tr2delta in Simulink delivers wrong values for x and pitch. By observing the output with a scope, the error could be identified. The output can be corrected by multiplying these values with -1.

ikine fails to find any joint values sometimes. An error message "Warning: filed to converge: try a different initial value of joint coordinates" This behaviour has been observed on some machines but cannot be reproduced on others. According to a forum post [11], setting the initial estimate of joint angles to a reasonable estimate like $[0, 0, 0, 0, 0, 0]$ can help.

4.5.3.3 Dynamic Simulation

The file simulink files contain a simulation of the robot with control strategies. Further details will be discussed in the chapter about developing a control strategy.

5. Developement of a Control Strategy

With a dynamic representation of the robot implemented in Matlab, control strategies can be developed and implemented. Two types of model-based control that are of interest for serial link manipulators were picked.

5.1 Computed Torque Control

With computed torque control, the current position and velocity of the manipulator are measured. Then all non-linearities are cancelled so that only the torque needed to overcome the inertia of the actuator needs to be applied as seen in 5.1. ([38] sec.5.2, P.190) *It belongs to a class of controllers known as inverse dynamic control. The principle is that the non-linear system is cascaded with its inverse so that the overall system has a constant unity gain.* ([14], sect. 9.4.2, p.274)

$$\Upsilon = M(\theta)\ddot{\theta}_d + C(\theta,\dot{\theta})\dot{\theta} + N(\theta,\dot{\theta}) \tag{5.1}$$

To correct for the difference between initial position and velocity of the manipulator and its desired states, state feedback can be added as seen in 5.2. The position error is $e = \theta - \theta_d$ and $\dot{e}$ is calculated similar. K_v and K_p are the position and velocity gain matrices. [38]

$$\Upsilon = M(\theta)(\ddot{\theta}_d - K_v\dot{e} - K_p e) + C(\theta,\dot{\theta})\dot{\theta} + N(\theta,\dot{\theta}) \tag{5.2}$$

5.2 Workspace Control

Instead of controlling the end-effector in joint space, a control in the Cartesian operational space is also possible. With this control scheme, the computationally intensive inverse kinematic problem does not need to be solved at each instant in time to generate a desired joint angle trajectory path. Also the feedback gains can be chosen for the directions in Cartesian space rather than for the individual joints. Trajectories in response to inputs follow a straight line in Cartesian space rather than in joint space. ([38] sec.5.4, P.195)

With the Jacobian given in section 4.3 Jacobian, the dynamics can be mapped into Cartesian space by exchanging joint space coordinates for cartesian coordinates as seen in 5.3. After substituting suitable expressions, and pre-multiplying by $J^{-T} = (J^{-1})^{-T}$, following equation can be obtained:

$$J^{-T}M(\theta)J^{-q}\ddot{x} + (J^{-T}C(\theta,\dot{\theta})J^{-1} + J^{-T}M(\theta)\frac{d}{dt}(J^{-1}))\dot{x} + J^{-T}N(\theta,\dot{\theta}) = J^{-T}\Upsilon \tag{5.3}$$

By redefining the dynamics matrices as
$\tilde{M} = J^{-T}MJ^{-1}$
$\tilde{C} = J^{-T}(CJ^{-1} + M\frac{d}{dt}(J^{-1}))$
$\tilde{N} = J^{-T}N$
$F = J^{-T}\Upsilon$
The dynamics become

$$\tilde{M}(\theta)\ddot{x} + \tilde{C}(\theta,\dot{\theta})\dot{x} + \tilde{N}(\theta,\dot{\theta}) = F \tag{5.4}$$

.
This allows to extend the computed torque control to workspace coordinates as seen in 5.5. x_d is the desired workspace trajectory and $e = x - x_d$ is the workspace error.

$$F = \tilde{M}(\theta)(\ddot{x}_d - K_v\dot{e} - K_p e) + \tilde{C}(\theta,\dot{\theta})\dot{x} + \tilde{N}(\theta,\dot{\theta}) \tag{5.5}$$

As the motors are still controlled in joint space, the torques can be found with
$\Upsilon = J^T F$.

5.3 Implementation

These control schemes can be implemented in Simulink.

5.3.1 Computed Torque Control

In figure 5.1 the computed torque control together with a Simulink representation of the robot can be seen. The gains Kp and Kd have been determined by trial and error. The desired path is fed into the controller together with the states of the robot. The controller calculates the Torque and sends it to the robot.

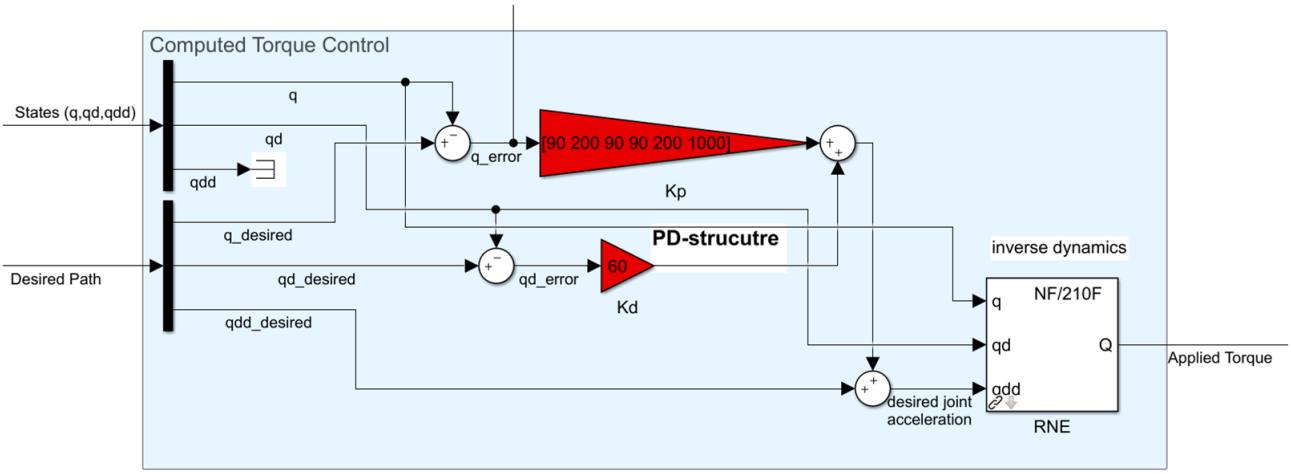

Figure 5.1: Simulink implementation of the computed Torque Control in Simulink

5.3.2 Workspace Control

In figure 5.2 the workspace control with a Simulink representation of the robot can be seen. The gains Kp and Kv have been determined by trial and error. The desired path is fed into the controller as a transformation matrix as well as in $[X, Y, Z, Roll, Pitch, Yaw]$ coordinates. For calculating the workspace error, a toolbox function is used. As this function has a bug, the X and $Roll$ error need to be multiplied by -1.

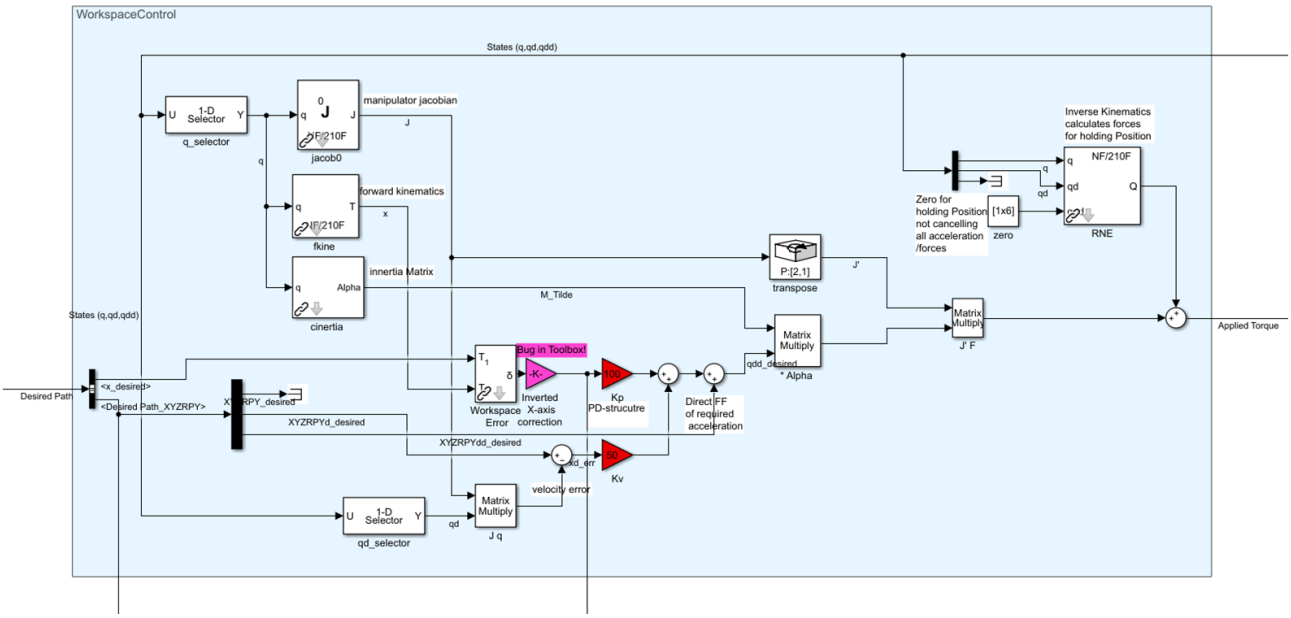

Figure 5.2: Workspace Control in Simulink

5.3.3 Robot model

The model created by the Matlab script is used by the ”Robot (model)” block as seen in figure 5.3. The zero-order hold block ensures a constant sample time of 5 milliseconds. The model implementation is the same for both control strategies.

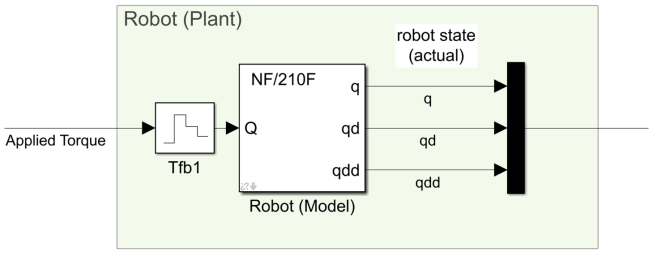

Figure 5.3: Robot model in Simulink

5.3.4 Path Generation

The input path to follow is generated by adjustable constants and function generators for circles, sine and others. The set values are close to normal pose with an offset of $1m$ in Z. These individual manipulations in all six degrees of freedom (DOF) need to be fused together into a transformation matrix. Additionally, the desired velocity and acceleration are calculated for better accuracy.

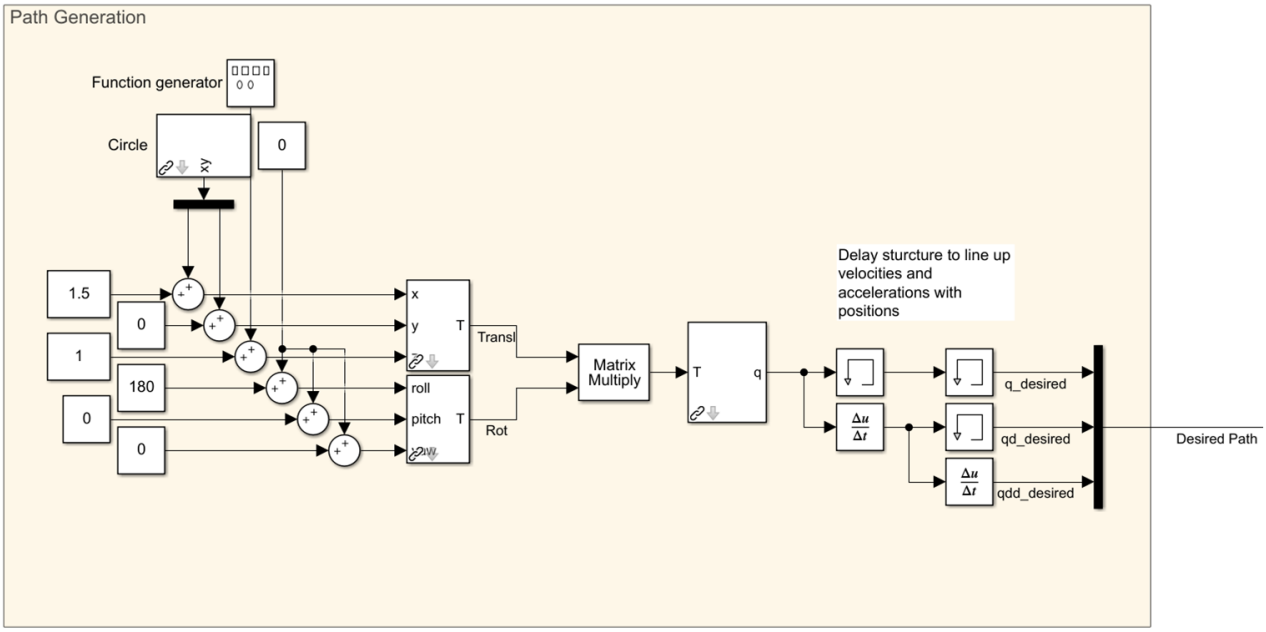

Figure 5.4: Path Generation in Simulink for Computed Torque Control

The path generation for workspace control is similar. Here, no inverse kinematics is needed as the desired Transformation matrix can be directly handed to the controller.

5.4 Performance of the Control

With the model and control implemented in MATLAB, the performance of the control schemes can be analysed. The performance of the two control strategies can be observed with different inputs. Following inputs are tested:

- Point to Point
- 3D-Path tracking

For the point to point motion, the robot is initialized in normal pose. Then, a desired position with zero offset in Z and a change in X and Y-position compared to normal pose is given to the controller. The 3D-Path tracking is observed on a circular path in the X/Y plane with a sinusoid on the Z-axis.

5.4.1 Computed Torque Control

Point to point The Path in the X/Y plane can be seen in figure 5.5. It can be clearly seen, that the robot takes a curved path to the desired end position, as the shortest path in joint space is taken.

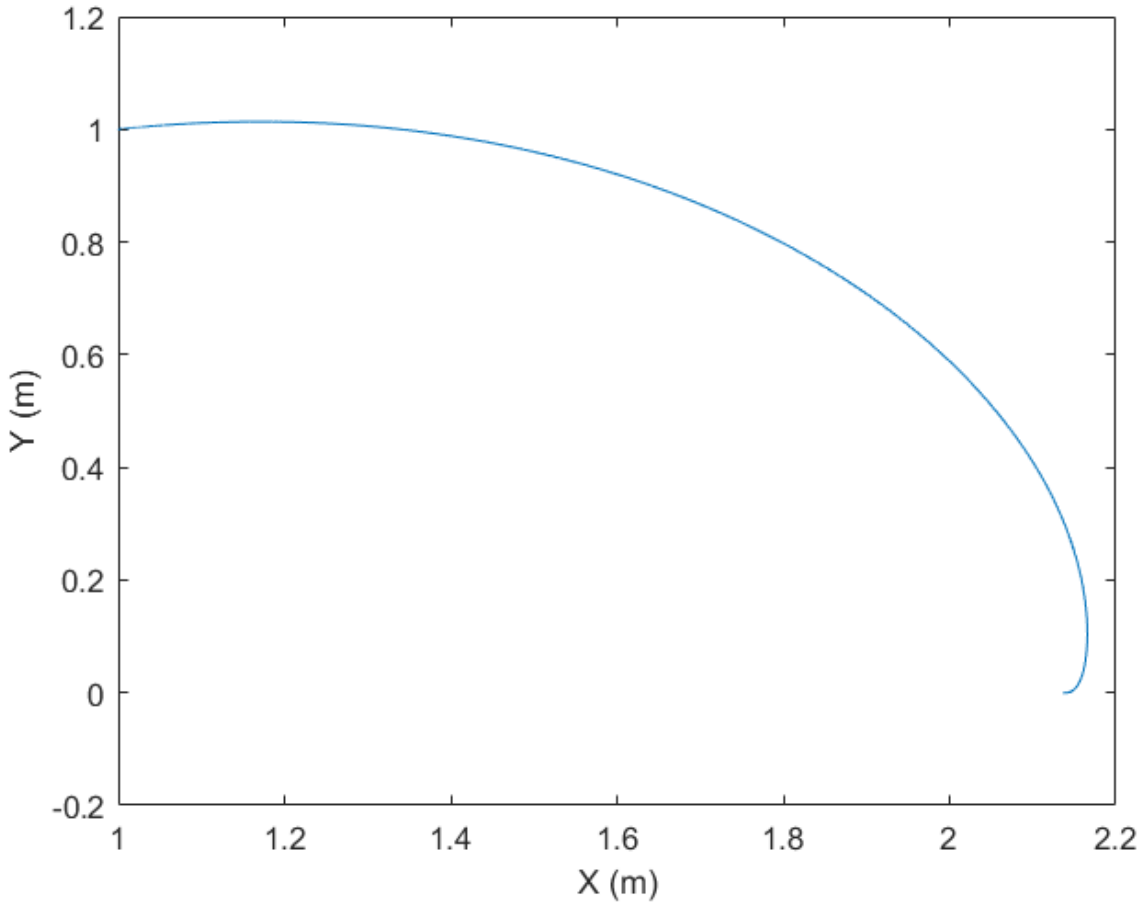

Figure 5.5: Path in X/Y plane for point to point motion with computed torque control

The errors in XYZ-position can be seen in figure 5.6. The endeffector reaches its destination straight with no overshoot. It shall be noted here, that the PD-tuning was chosen very low to avoid any overshoot and instabilities. The errors go below 1mm between 3.7 and 5.1 seconds.

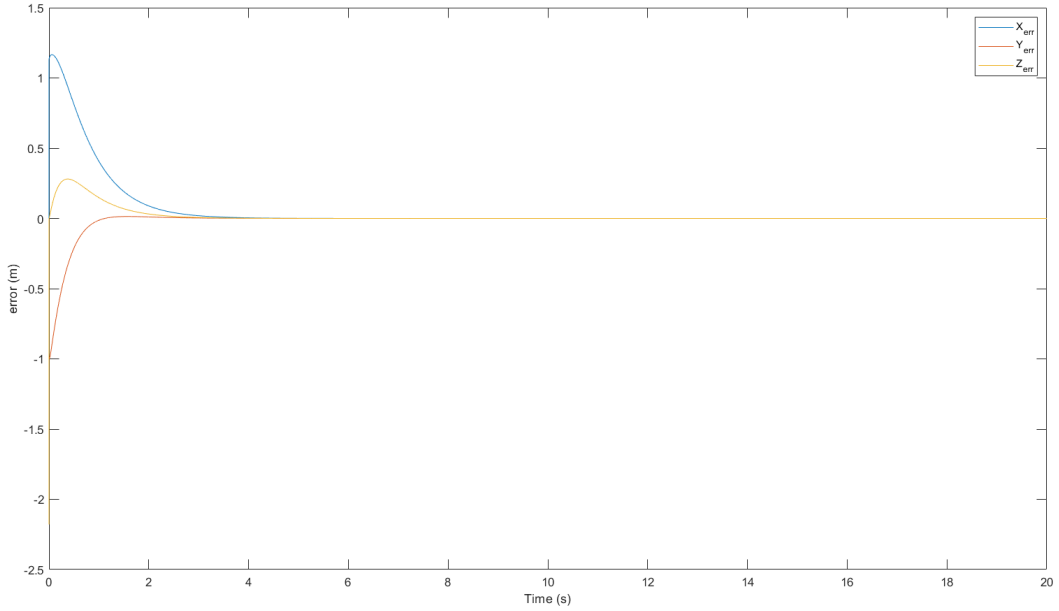

Figure 5.6: Errors in XYZ-position for point to point motion with computed torque control

3D-path The XYZ position vs the desired XYZ path can be seen in figure 5.7After a short Synchronization time due to difference in initial position, the end effector follows the desired trajectory.

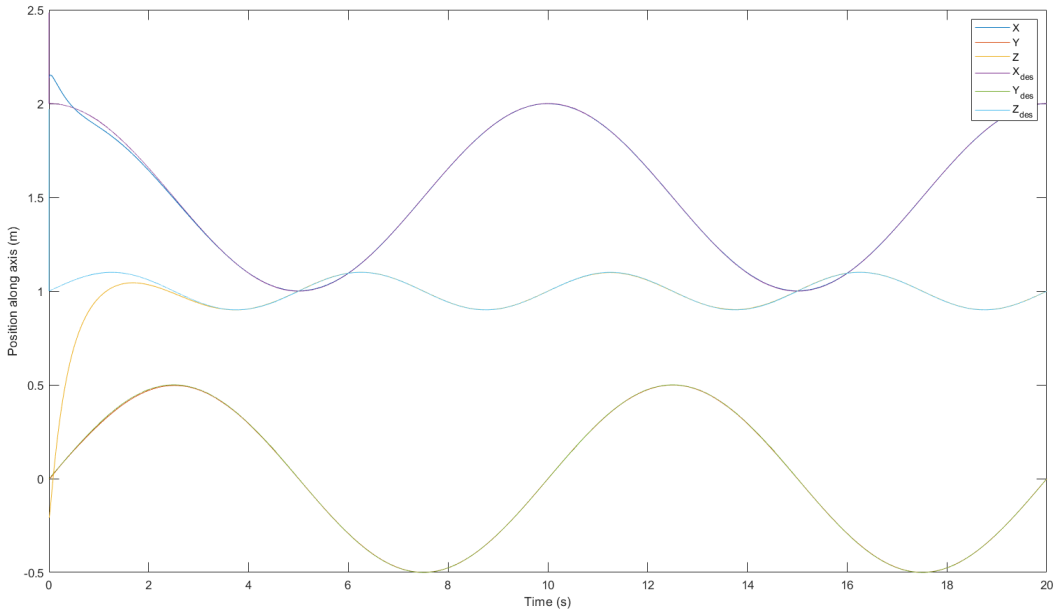

Figure 5.7: XYZ-position vs desired XYZ path with computed torque control

The errors in XYZ-tracking can be seen in figure 5.8 For the trajectory tracking, an overshoot in x-direction can be observed, that settles with the error in z-direction. Even after the synchronization period, very small tracking errors appear. These are results of the coupled non-linearities.

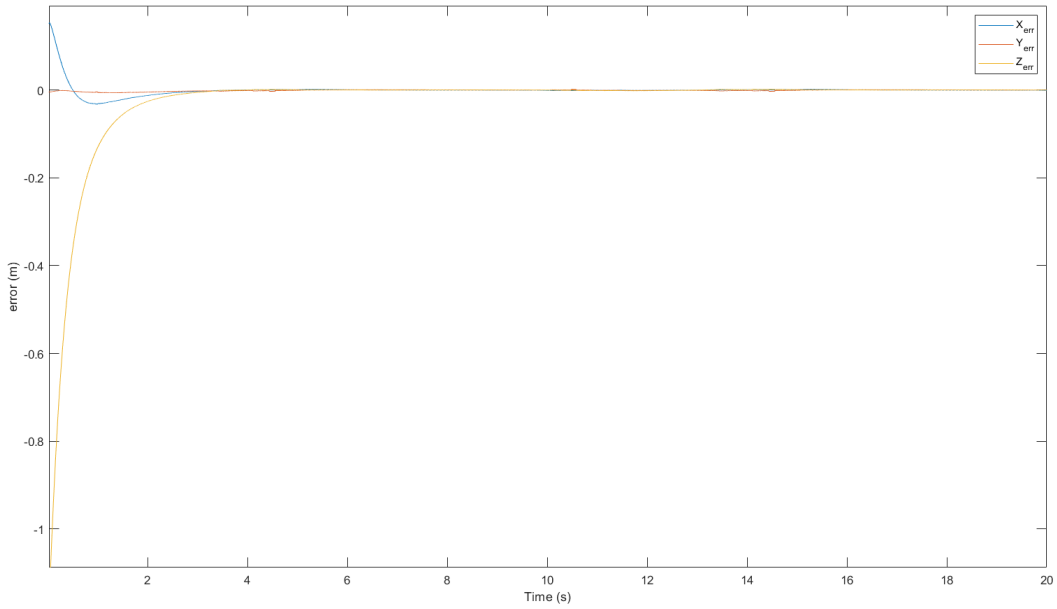

Figure 5.8: Errors in XYZ-position for 3D path tracking with computed torque control

These overshoots do not occur in the joint space as seen in figure 5.9.

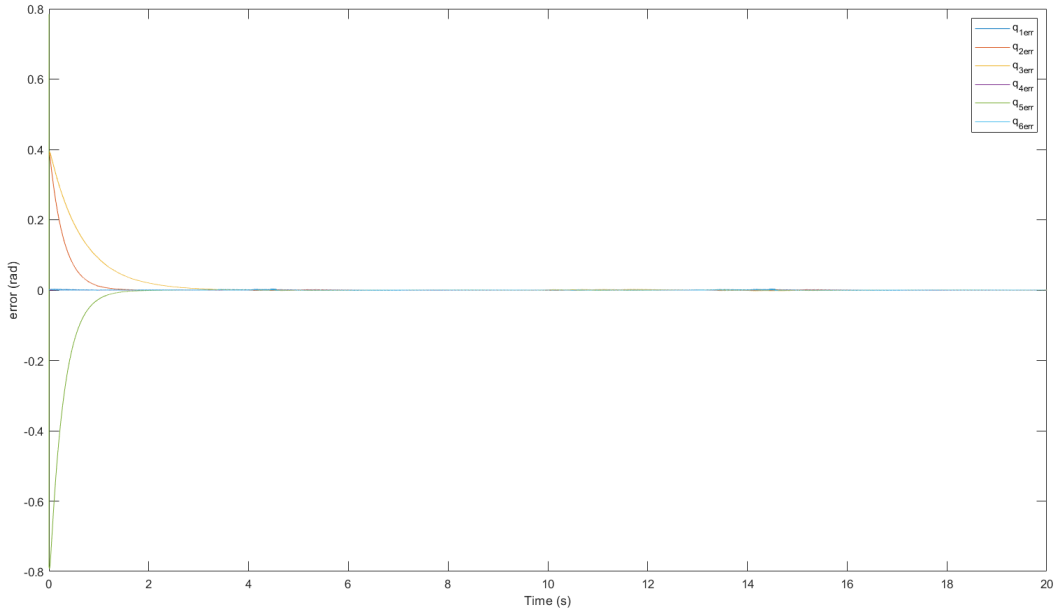

Figure 5.9: Errors in joint space for 3D path tracking with computed torque control

5.4.2 Workspace Control

Point to point The Path in the X/Y plane can be seen in figure 5.10. Besides a small movement to the side in the beginning due to the difference in initial conditions, the endeffector follows a straight path to the desired position.

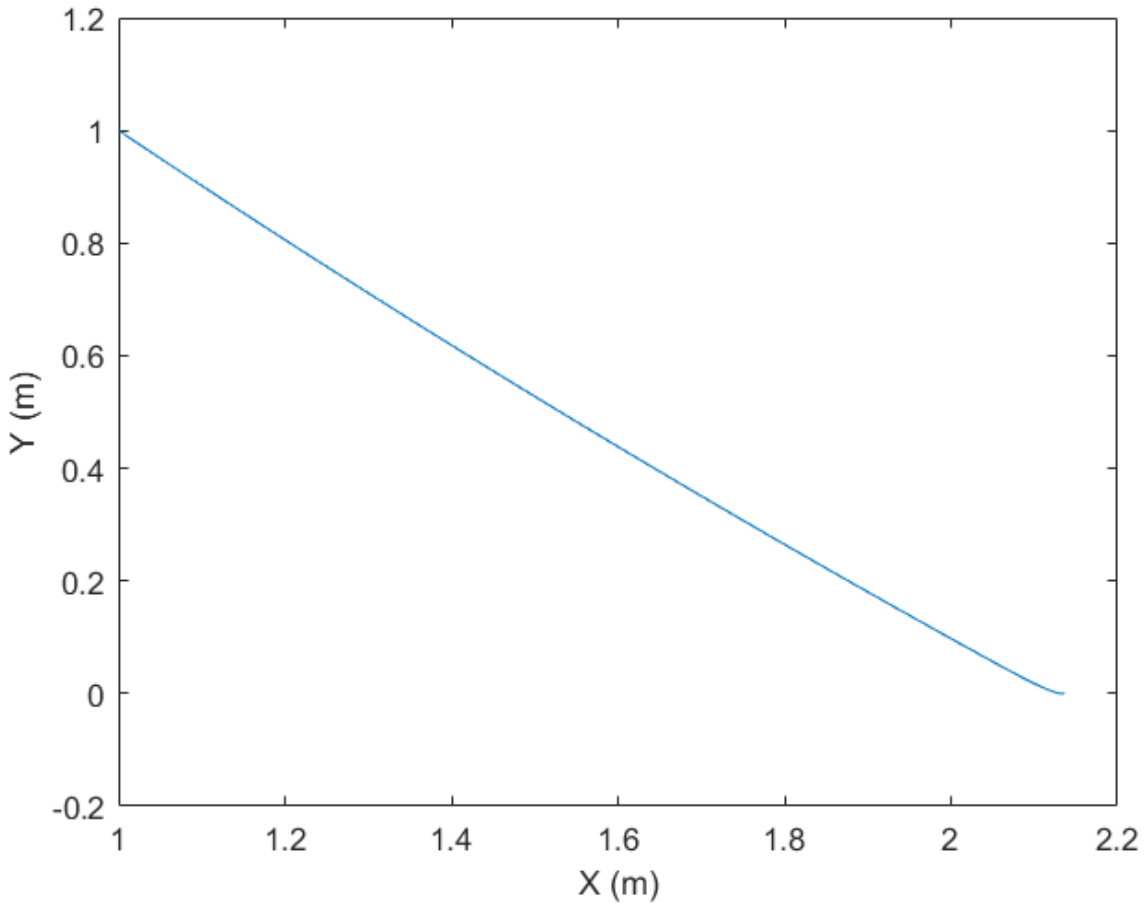

Figure 5.10: Path in X/Y plane for point to point motion with Workspace control

The errors in XYZ-position can be seen in figure 5.11. No overshoot occurs. The endeffector moves straight to the target.

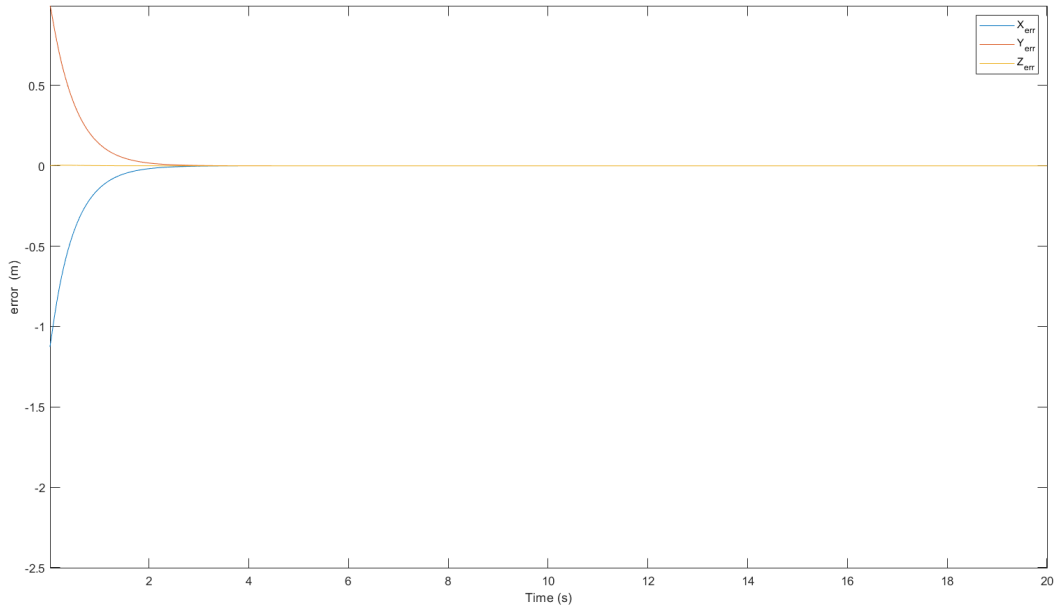

Figure 5.11: Errors in XYZ-position for point to point motion with Workspace control

For point to point tracking, it is advised to turn off the acceleration and velocity feedforward to avoid unreasonable spikes in joint torques.

3D-path The XYZ position vs the desired XYZ path can be seen in figure 5.12 After a short Synchronization time due to difference in initial position, the end effector follows the desired trajectory.

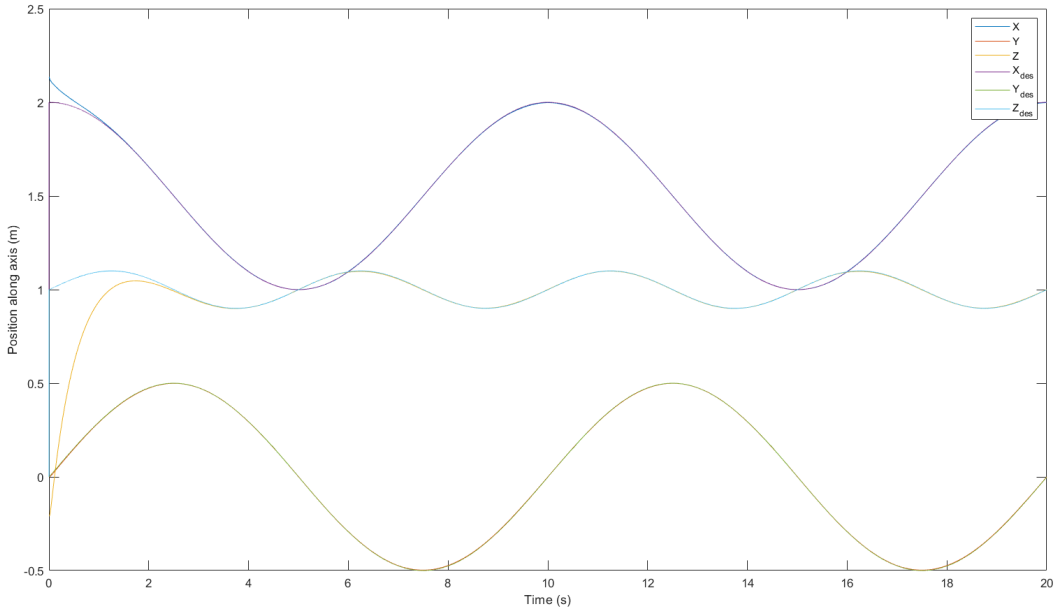

Figure 5.12: XYZ-position vs desired XYZ path with Workspace control

The errors in XYZ-tracking can be seen in figure 5.13 No overshoot can be seen. After the synchronization period, small tracking errors that vary with the path appear. These could probably be mitigated with a more aggressive PD tuning.

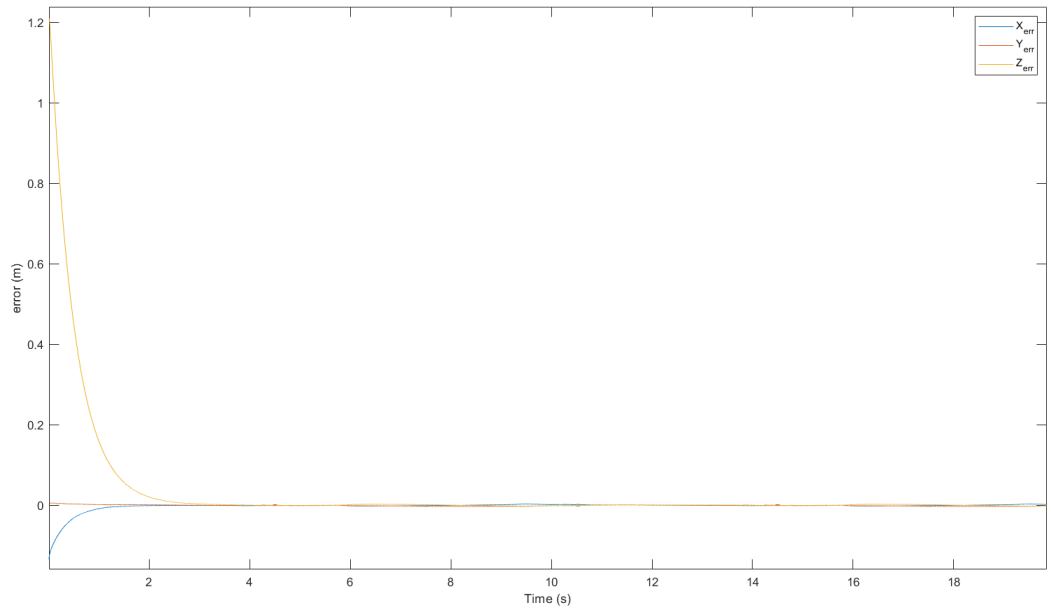

Figure 5.13: Errors in XYZ-position for 3D path tracking with Workspace control

5.4.3 Performance Comparison of Control Strategies

When comparing figure 5.5 with 5.10, it can be seen, that the computed torque control follows a curved line while the workspace control follows a straight line in euclidean space. This is a property of the two control strategies. While the computed torque control follows the shortest distance in joint space, the Workspace control follows the shortest distance in Euclidean space.

For point to point motion even with lower PD tuning factors, the Workspace control outperforms the joint space control. as seen in figure 5.14. The error in X-direction settles faster with workspace control. In Z-direction no visible error occurs with workspace control, while with the control in joint space, an overshoot from zero occurs. Only in Y-direction, the joint space control outperforms workspace control.

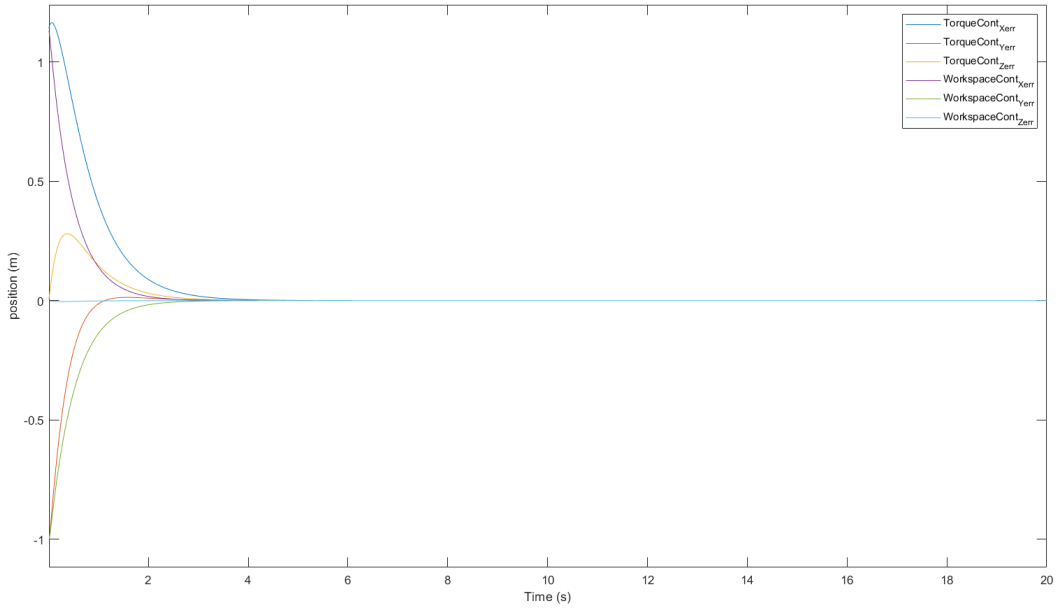

Figure 5.14: Errors in XYZ-position for 3D path tracking with Computed Torque Control vs Workspace control

The 3D tracking can be visualized isolated in the XY-plane as seen in figure 5.15. It can be clearly seen, that the workspace control joins the desired trajectory faster and without overshoot. This is especially of importance when it comes to collision avoidance and applications like Computer Numeric Control (CNC) milling, where overshoot can destroy the product and machine. As the desired path is initialized at zero position in workspace, a jump occurs, that leads to the yellow line in the middle.

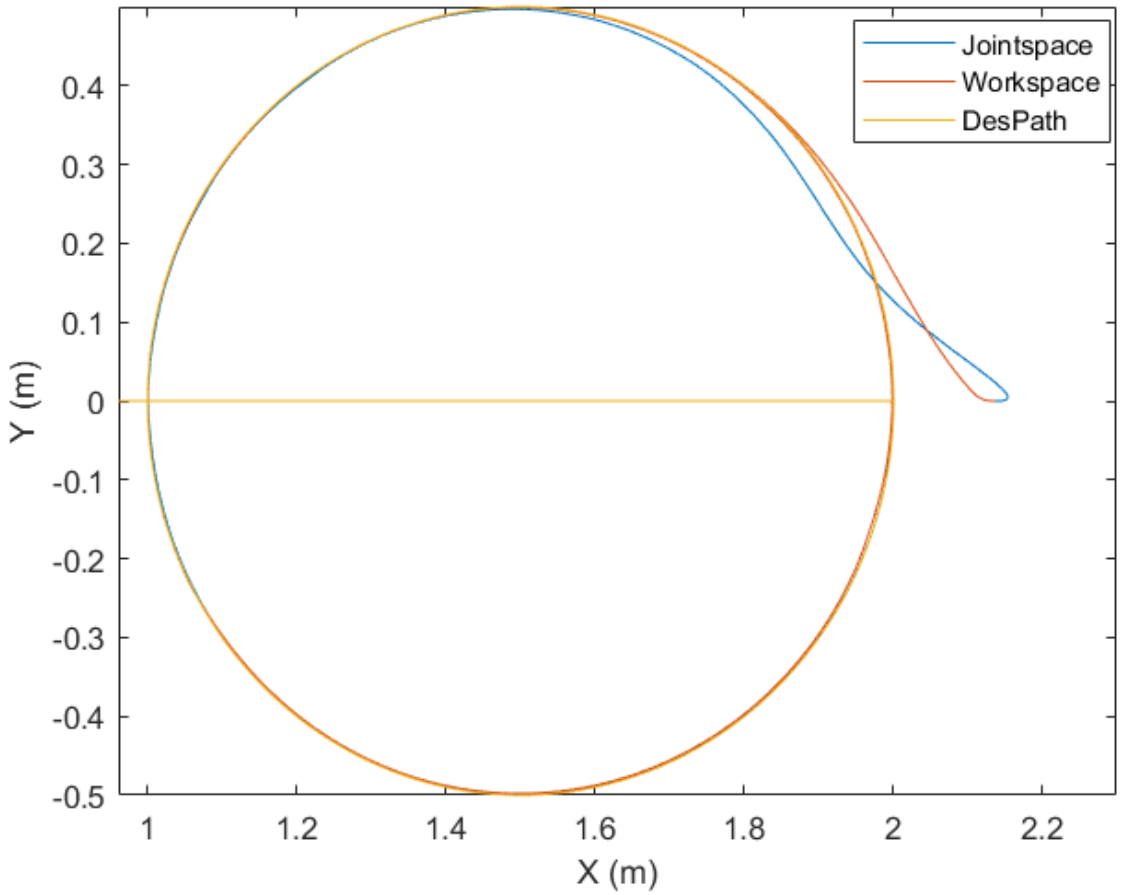

Figure 5.15: XY Path Joint Space control vs Workspace Control

Overall, the workspace control gives better performance. Also the computational performance is a lot better, as the inverse kinematic problem does not need to be solved.

5.5 Pick and Place

The goal of this thesis work is to develop a control strategy for the pick and place operation requested by SPC. A program that can be given to the robot controller can be found in appendix K TP Program. This program defines a sequence of positions. The robot follows these positions to achieve the desired pick and place operation. The $[XYZ]$ coordinates can be extracted from this program and fed into the Workspace control simulation.

5.5.1 Input Sequence

In figure 5.16 the desired sequence of positions can be seen, plotted over time. The sequence ends at second 32 and repeats after that. The positions are given to the controller in an interval of 2 seconds.

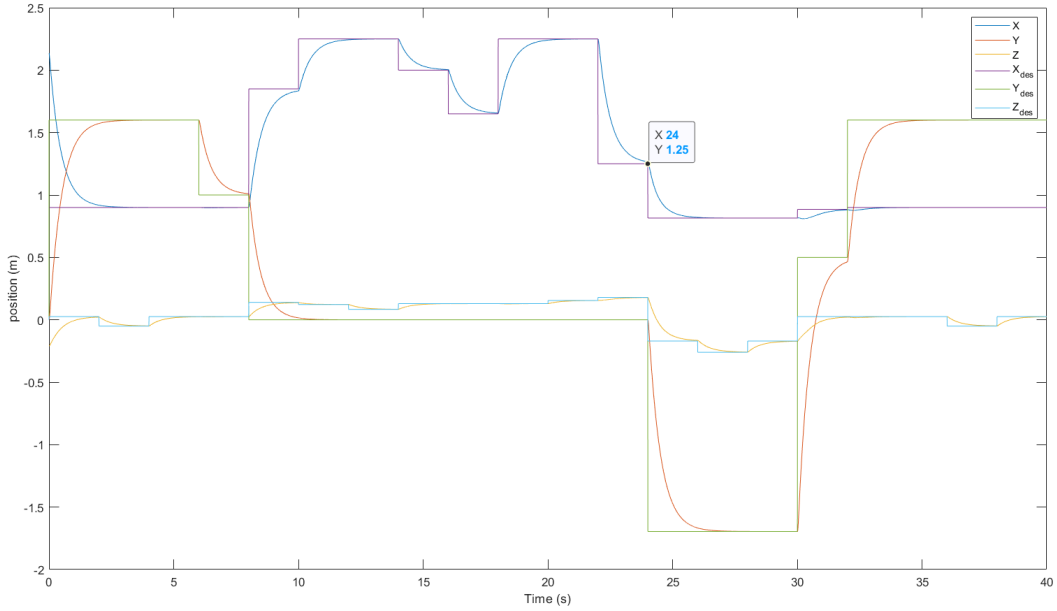

Figure 5.16: XYZ-position vs desired position sequence for pick and place with Workspace control

5.5.2 Point Tracking

As seen in fig. 5.16 the controller manages to follow the desired sequence of points with some difficulties. Certain points cannot be reached within a 2 second interval time. An example for this tracking error can be seen at second 24 . In fig. 5.17 an error in x-direction of 0.01605 can be seen at that point. This shows, that not all positions within the workspace of the robot can be reached in 2 seconds with the chosen PD tuning. Either the tuning needs to be adjusted, or the interval time needs to be increased.

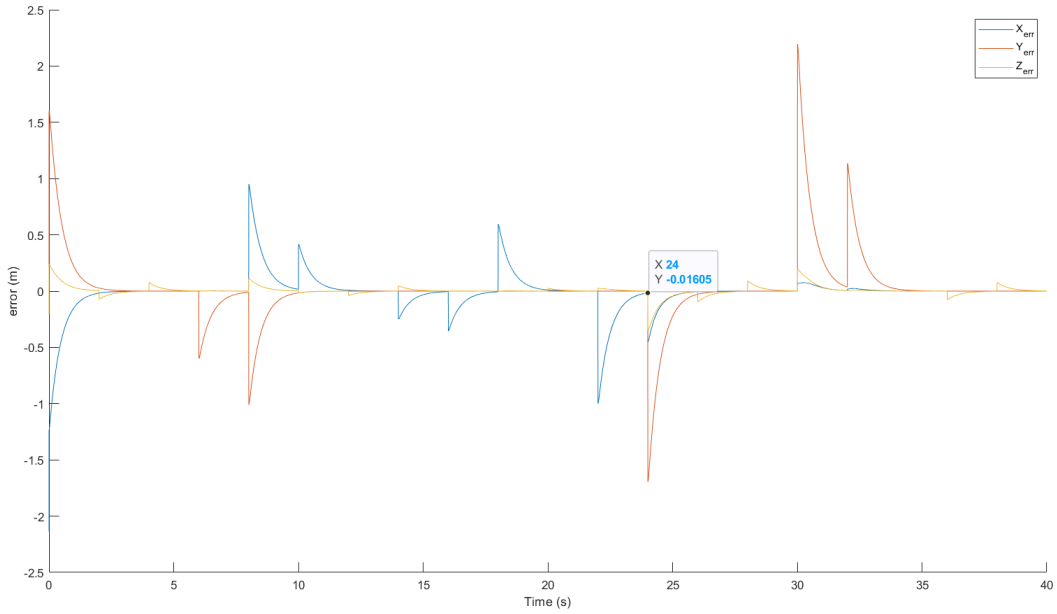

Figure 5.17: Errors in XYZ-position for pick and place with Workspace control

5.5.3 Improvement by Trajectory Tracking

Instead of handing over a set of points to the controller, trajectory tracking can be used to improve the performance for the desired task without making changes to the controller. For this, interpolated positions are

calculated between the desired endpoints. In fig. 5.18 the desired trajectory and the taken path can be seen.

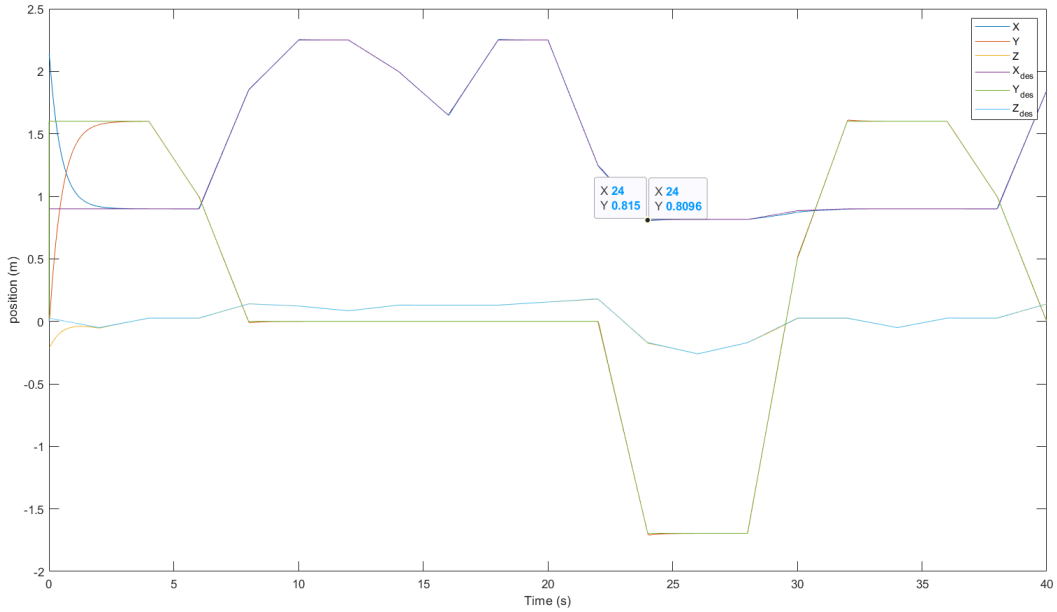

Figure 5.18: XYZ-position vs desired interpolated path for pick and place with Workspace control

With this addition, the endeffector manages to reach the desired points within 2 seconds, as velocity and acceleration can be used in the feedforward. The error for the desired position at second 24 reduces to an overshoot of 0.0054.

5.5.4 Taken Path in Workspace

In fig. 5.19, the path of the endeffector in the XY plane can be seen with an underlay of the robotic workcell at SPC. The time is represented by colour. As can be seen, the robot arm travels through itself. This is due to the limitations of this simulation. A collision detection and avoidance is not implemented, as it is computationally expensive.

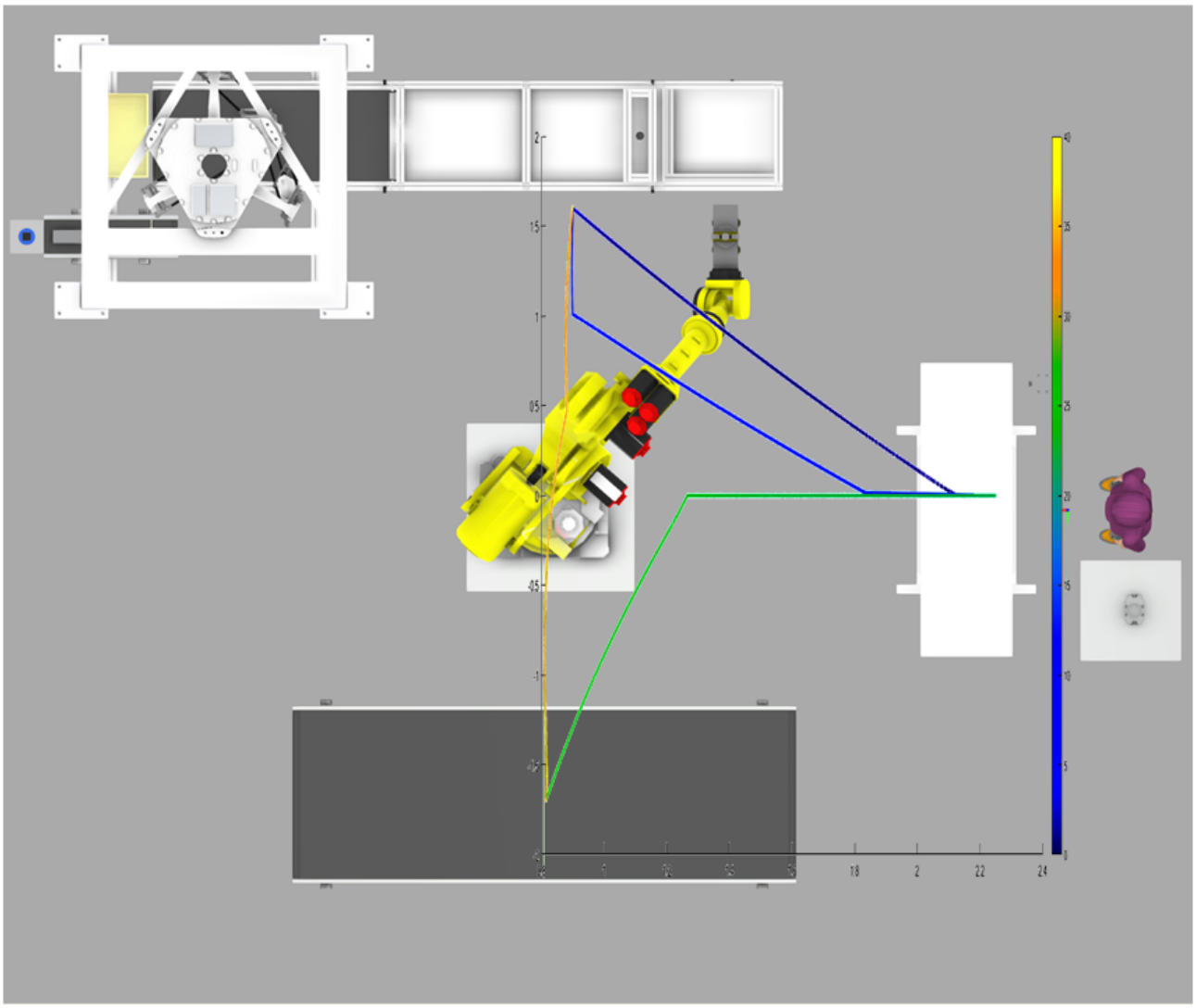

Figure 5.19: Path in XY-plane in the robotic work-cell at SPC

6. Conclusions and further work

6.1 Robot Programming

Programming of the R30iA robot controller was mainly done with Visual Components (VC). This Software package simplifies the offline programming as it shows the actions of the robot within the production line. Nevertheless, programs should still be verified with Roboguide. Program generation with VC has proven to be difficult in the beginning, as programs created with VC did not behave as expected or could not be executed. Also program transfer with Roboguide is faster and easier than with USB-sticks. Because of this, it is advised to Smart Production Centre (SPC) to purchase a licence of Roboguide for further work.

6.2 Model of 210F

The model created in Matlab is a good kinematic representation. Measurements could be directly taken from the datasheet. Due to difficulties in determining the dynamic parameters, the resemblance of the dynamics is limited. A further analysis of the manipulator by gathering movement data or complete disassembly and manual measurement are necessary. Like this, the model can be made more accurate, and the control can be fine-tuned for that machine.

6.3 Control of 210F

The controller shows good performance in point to point motion as well as trajectory tracking. Good accuracy can be achieved, and movements are executed in a reasonable timespan. A more aggressive PD- tuning could provide better results if necessary. This tuning could be either done manually or by using a tuning method like Ziegler-Nichols. Further improvements on the control strategy could be made by changing the gains depending on the position error. Ways to achieve this would be a sliding mode control or a fuzzy logic. Additionally, as the dynamic parameters of the robot are hard to determine, an Model Reference Adaptive Controller (MRAC) control could be added to compensate for that. As the inverse kinematics is computationally intensive in its current implementation, it could be replaced by the form presented in subsection 4.2.3 Inverse Solution of 6 degrees of freedom (DOF) Robot. Alternative approaches to that problem could be the use of an Artificial Neural network based structure to reduce computation time.

6.4 Validation of Model and Control

The model and control presented in the previous sections were not tested on the actual Robot. This was due to a lack of communication infrastructure for interfacing with the robot. For testing, a communication between the robot an MATLAB needs to be established. Robot Operating System (ROS) could be used here as a middleware to establish this real-time communication. As real-time performance is necessary, the program needs to run on a computer with high computational power.

6.5 Pick and Place Operation

The goal of this thesis-work was setting up the Fanuc 210F for pick and place operation. This goal has been reached. The robot was mounted at the best position for this task. A program has been developed and communication with other subprocesses within the production line was achieved. Further necessary steps lie in the detection of the Fibre Reinforced Plastics (FRP) material blops to be picked. This position could initially be provided by the delta robot. Further improvements could be made by including a camera detection system.

Acknowledgements

I want to thank everyone who has helped me with this thesis work. It was a great learning experience for me.

First of all, I would like to thank Peter Verschut, the project manager at SPC, who has made this project possible. By providing the SPC lab at MIC and letting me do experiments with the machines, I could get a lot of practical experience with the Fanuc210F and other machines. Also he connected me with experts in the field so I could make contacts for further work in the field of robotics.

Dimitri Jeltsema gave me a good hint how to model and control the robot based on the content of his lectures. Besides that, Ellen Wesselingh and Trung Nguyen have given me great advice on the writing and modelling of the robot. Without your guidance this journey would have been a lot harder. Your critical questioning and pushing in the right direction have helped me a lot. Thank you for your patience and effort.

Peter Corke has made the modelling and control of the robot arm a lot easier for me with the toolbox that he distributes for free. I hope, this toolbox will see many more iterations with additional functionalities and improvements.

Aliya Patel, a fellow master student working on another type of robot was a great help in the practical part of the project. Also besides the work, I am happy to have you as a friend. I wish you success in your career and I hope you will get to work in your field of interest.

Also Didier, Suzanne, Ivan and the other employees and students at SPC were always there, when I needed a helping hand. We were a great team.

I would also like to thank Bram and his colleagues from Qing for the great collaboration.

Special thanks goes to Karlijn who has kept me sane in times of isolation due to Covid-19. Thanks for the moral support!

Finally, I would like to thank my Mother and her partner Gerhard for making this master studies possible for me.

References

[1] URL: https://specials.han.nl/sites/automotive-research/smart-cell/index.xml.

[2] Riku Ala-Laurinaho. "Sensor data transmission from a physical twin to a digital twin; Anturidatan lähettäminen fyysiseltä kaksoselta digitaaliselle kaksoselle". en. G2 Pro gradu, diplomity. May 6, 2019, pp. 90 + 15. URL: http://urn.fi/URN:NBN:fi:aalto-201905123028.

[3] Ahmed R. J. Almusawi, Lale Dülger, and Sadettin Kapucu. "A New Artificial Neural Network Approach in Solving Inverse Kinematics of Robotic Arm (Denso VP6242)". In: *Computational Intelligence and Neuroscience* 2016 (Aug. 2016), p. 10. DOI: 10.1155/2016/5720163.

[4] *automaton*. URL: https://www.merriam-webster.com/dictionary/automaton.

[5] *Autonomous Robot Vehicles*. Aug. 6, 1991.

[6] Lukas Barinka. "Inverse Kinematics - Basic Methods". In: ed. by Ing. Roman Berka. Mar. 21, 2002.

[7] Fanuc BENELUX. *Fanuc Academy Manual*. Dutch. Robot hands on manual. FANUC.

[8] Jan Boddez. *Robotics_Kinematics_and_Dynamics*.

[9] Kamel Bouzgou and Zoubir Ahmed-Foitih. "Workspace Analysis and Geometric Modeling of 6 DOF Fanuc 200IC Robot". In: vol. 182. 4th WORLD CONFERENCE on EDUCATIONAL TECHNOLOGY RESEARCHES (WCETR-2014). 2015, pp. 703–709. DOI: https://doi.org/10.1016/j.sbspro.2015.04.817. URL: http://www.sciencedirect.com/science/article/pii/S187704281503092X.

[10] Daniel Constantin et al. "Forward Kinematic Analysis of an Industrial Robot". In:

[11] Peter Corke. *Google groups - The ikine Function used for the Puma560*. Apr. 17, 2018. URL: https://groups.google.com/forum/#!topic/robotics-tool-box/gtg_NHCyc_E.

[12] Peter Corke. *LESSON Denavit-Hartenberg notation*. EN. Ed. by editor. Format: Video lecture. Queesland University of Technology. Apr. 9, 2017. URL: https://robotacademy.net.au/lesson/denavit-hartenberg-notation/ (visited on 01/14/2020).

[13] Peter Corke. *Robotics Toolbox. Introduction*. EN. Homepage of Peter Corke. URL: https://petercorke.com/wordpress/toolboxes/robotics-toolbox (visited on 01/18/2020).

[14] Peter Corke. *Robotics, Vision and Control: Fundamental Algorithms in MATLAB*. 1st. Springer Publishing Company, Incorporated, 2013. ISBN: 3642201431.

[15] J.J. Craig. *Introduction to Robotics: Mechanics & Control*. Introduction to robotics : Mechanics & control / John J ... Craig. found at https://robotics.stackexchange.com/questions/10322/is-it-possible-to-get-all-possible-solutions-of-inverse-kinematics-of-a-6-dof-ar. Addison-Wesley Publishing Company, 1986. ISBN: 9780201103267. URL: https://books.google.nl/books?id=uDNSAAAAMAAJ.

[16] Mahidzal Dahari and Jian-Ding Tan. "Forward and inverse kinematics model for robotic welding process using KR16KS KUKA robot". In: *2011 4th International Conference on Modeling, Simulation and Applied Optimization, ICMSAO 2011* (Apr. 2011). DOI: 10.1109/ICMSAO.2011.5775598.

[17] *Digital twin*. URL: https://en.wikipedia.org/wiki/Digital_twin#Definitions.

[18] A. El Saddik. "Digital Twins: The Convergence of Multimedia Technologies". In: *IEEE MultiMedia* 25.2 (Apr. 2018), pp. 87–92. ISSN: 1941-0166. DOI: 10.1109/MMUL.2018.023121167.

[19] Khaled Elashry and Ruairi Glynn. "An Approach to Automated Construction Using Adaptive Programming". In: Mar. 2014, pp. 51–66. ISBN: 978-3-319-04662-4. DOI: 10.1007/978-3-319-04663-1_4.

[20] *European Train Control System (ETCS)*. URL: https://www.thalesgroup.com/en/european-train-control-system-etcs.

[21] *FANUC Robot R-2000iB series*.

[22] Roy Featherstone. *Robot dynamics*. Version revision Nr. 91723. Brain Corporation. Oct. 21, 2011. DOI: doi:10.4249/scholarpedia.3829. URL: http://www.scholarpedia.org/article/Robot_dynamics (visited on 01/25/2020).

[23] Michael Grieves. "Digital Twin: Manufacturing Excellence through Virtual Factory Replication". In: (Mar. 2015). URL: https://www.researchgate.net/publication/275211047_Digital_Twin_Manufacturing_Excellence_through_Virtual_Factory_Replication.

[24] IEEE. *Unimate IEEE*. Ed. by IEEE. URL: https://robots.ieee.org/robots/unimate/.

[25] Jamshed Iqbal, Muhammad Ul Islam, and Hamza Khan. "Modeling and analysis of a 6 DOF robotic arm manipulator". In: *Canadian Journal on Electrical and Electronics Engineering* 3.6 (Jan. 2012), pp. 300–306. URL: https://www.researchgate.net/publication/280643085.

[26] S. Jaladi and T.E. Rao. "An Inverse Kinematics Analysis of Space Station Remote Manipulator System (SSRMS) Using Genetic Algorithms". In: *International Journal of Mechanical and Production Engineering Research and Development* 9 (Feb. 2019), pp. 217–226. DOI: 10.24247/ijmperdfeb201921.

[27] Reza N. Jazar. "Inverse Kinematics". In: *Theory of Applied Robotics: Kinematics, Dynamics, and Control*. Boston, MA: Springer US, 2007, pp. 263–296. ISBN: 978-0-387-68964-7. DOI: 10.1007/978-0-387-68964-7_6. URL: https://doi.org/10.1007/978-0-387-68964-7_6.

[28] Pradeep Khosla and Takeo Kanade. "Experimental Evaluation of Nonlinear Feedback and Feedforward Control Schemes for Manipulators". In: *I. J. Robotic Res.* 7 (Feb. 1988), pp. 18–28. DOI: 10.1177/027836498800700102.

[29] Werner Kritzinger et al. "Digital Twin in manufacturing: A categorical literature review and classification". In: *IFAC-PapersOnLine* 51.11 (2018). 16th IFAC Symposium on Information Control Problems in Manufacturing INCOM 2018, pp. 1016–1022. ISSN: 2405-8963. DOI: https://doi.org/10.1016/j.ifacol.2018.08.474. URL: http://www.sciencedirect.com/science/article/pii/S2405896318316021.

[30] McGraw-Hill. *Dictionary of Scientific & Technical Terms*. 1970. URL: https://encyclopedia2.thefreedictionary.com/Extraneous+Root.

[31] Stephen Mraz. *The Difference between Medial and Lateral, Proximal and Distal, and Superior and Inferior (Biomechanics)*. June 22, 2016. URL: https://www.machinedesign.com/markets/medical/article/21834827/the-difference-between-medial-and-lateral-proximal-and-distal-and-superior-and-inferior-biomechanics (visited on 02/03/2020).

[32] Dr.-Ing. John Nassour. *Inverse Kinematics Serial link manipulators*. Humanoide Roboter (ehem. Robotik). Lecture in inverse Kinematics for serial link manipulators, TU Chemitz, Fakultät für Informatik. Technische Universität Chemnitz - Fakultät für Informatik. Nov. 22, 2016. URL: https://www.tu-chemnitz.de/informatik/KI/edu/robotik/ws2016/lecture-ik%201.pdf (visited on 01/18/2020).

[33] Trung Nguyen. Company Supervisor for project.

[34] Aliya Patel. "Using Fanuc M-3iA/6S (Delta-bot/Delta robot) for improved efficiency in FRC parts formation". MasterThesis. HAN, Mar. 3, 2020.

[35] Richard P. Paul. "Robot manipulators : mathematics, programming, and control : the computer control of robot manipulators". In: 1981.

[36] *Program inverse kinematics algorithms with MATLAB*. Mathworks online tutorial.

[37] Qinglin Qi et al. "Digital Twin Service towards Smart Manufacturing". In: *booktitle*. 2018.

[38] Li Ze-xiang Richard M. Murray S. Shankar Sastry. *A Mathematical Introduction to Robotic Manipulation*. CRC Press, 1994.

[39] *Robot*. URL: https://www.merriam-webster.com/dictionary/robot.

[40] *Roboterkinematik*. DE. Version 1. Wikipedia. May 28, 2005. URL: https://upload.wikimedia.org/wikipedia/de/a/ae/Roboterkinematik.png (visited on 01/15/2020).

[41] Open Robotics. Website of ROS project.

[42] *ROS Answers - What is Ros?* URL: https://answers.ros.org/question/12230/what-is-ros-exactly-middleware-framework-operating-system/.

[43] Benjamin Schleich et al. "Shaping the digital twin for design and production engineering". In: *CIRP Annals - Manufacturing Technology* 66 (Apr. 2017), pp. 141–144. DOI: 10.1016/j.cirp.2017.04.040.

[44] Raymond A. Serway and John W. Jewett. *Physics for Scientists and Engineers with Modern, Hybrid (with Enhanced WebAssign Homework and EBook LOE Printed Access Card for Multi Term Math and Science)*. 8th. USA: Brooks/Cole Publishing Co., 2011. ISBN: 113310360X.

[45] Matt Simon. *What Is a Robot*. Aug. 24, 2017. URL: https://www.wired.com/story/what-is-a-robot/.

[46] Wolfgang Smidt. "Verallgemeinerte inverse Kinematik für Anwendungen in der Robotersimulation und der virtuellen Realität". MA thesis. Institut für Roboterforschung Universität Dortmund, 1998.

[47] Angela Sodemann. *Robotics 1 U1 (Kinematics) S2 (Kinematic Diagrams) P1 (Denavit-Hartenberg Frames).* URL: https://www.youtube.com/watch?v=ut4uZ6Yzv6o&list=PLT_0lwItn0sDBE98BsbaZezflB96ws12b&index=7.

[48] Mark W. Spong. *Robot Dynamics and Control.* 1st. USA: John Wiley and Sons, Inc., 1989. ISBN: 047161243X.

[49] Jay Strybis. *One Robotics Website.* Ed. by ONE Robotics Company. URL: www.onerobotics.com/.

[50] Jay Strybis. *Robot Whispering.* Ed. by ONE Robotics Company. URL: https://www.onerobotics.com/robot-whispering/ (visited on 04/21/2020).

[51] Dr. Swet Chandan Tarun Pratap Singh Dr. P. Suresh. "Forward and Inverse Kinematic Analysis of Robotic Manipulators". In: *International Research Journal of Engineering and Technology (IRJET)* 4.2 (Feb. 2017).

[52] R. J. Urbanic. "Methods for Evaluating the Functional Work Space for Machine Tools and 6 Axis Serial Robots." In: *SAE International Journal of Materials & Manufacturing.* 9.2 (May 2016), pp. 465–473.

[53] Ashwin Kumar Venkatesan et al. *CONTROL OF A FANUC ROBOTIC ARM USING MATLAB MANUAL.* USC Viterbi / Aerospace and Mechnaical Engineering department.

[54] Ashwin Kumar Venkatesan et al. *CONTROL OF A FANUC ROBOTIC ARM USING MATLAB MANUAL.* USCViterbi School of Engineering - Aerospace and Mechanical Engineering department.

[55] Karl Wallkum. "Using FANUC R-2000iC/210F (6-axis robot) for improved efficiency in FRC parts formation". Major Project Plan. Nov. 2019.

[56] George Watson. *Right Hand Rule for Cross Products.* Ed. by Univ. of Delaware. 1998. URL: http://www.physics.udel.edu/~watson/phys345/Fall1998/class/1-right-hand-rule.html.

[57] Kong Wei. "Control and Safety Mechanisms for a 3 DOF Manipulator with Human Interaction". Supervisor: Prof.dr.ir. Aart-Jan de Graaf. MA thesis. HAN, Oct. 2014.

[58] Mike Wilson. *Implementation of Robot Systems.* An Introduction to Robotics, Automation, and Successful Systems Integration in Manufacturing. Butterworth-Heinemann, 2015. DOI: https://doi.org/10.1016/C2012-0-00795-8.

[59] William G. Wong. *What's the Difference Between a Simulation and a Digital Twin?* May 24, 2018. URL: https://www.electronicdesign.com/technologies/embedded-revolution/article/21806550/whats-the-difference-between-a-simulation-and-a-digital-twin.

[60] Yan Wu et al. "Inverse Kinematics Solution and Optimization of 6DOF Handling Robot". In: *Applied Mechanics and Materials* 635-637 (2014). DOI: 10.4028/www.scientific.net/AMM.635-637.1355.

[61] Prof. Dr. Klaus Wuest. *Denavit-Hartenberg-Konventionen.* Technische Hochschule Mittelhessen. Wiesenstrasse 14, 35390 Giessen, Deutschland, June 2018. URL: https://homepages.thm.de/~hg6458/Robotik/Denavit-Hartenberg.pdf.

Appendices

A. Assignment of Individual Frames

Abstract view of coordinate frames For a better overview, the drawing of the robot in figure 4.2 can be removed, which reveals the pure coordinate frames. In figure A.1 is described, in which relation the coordinate frames stand to each other.

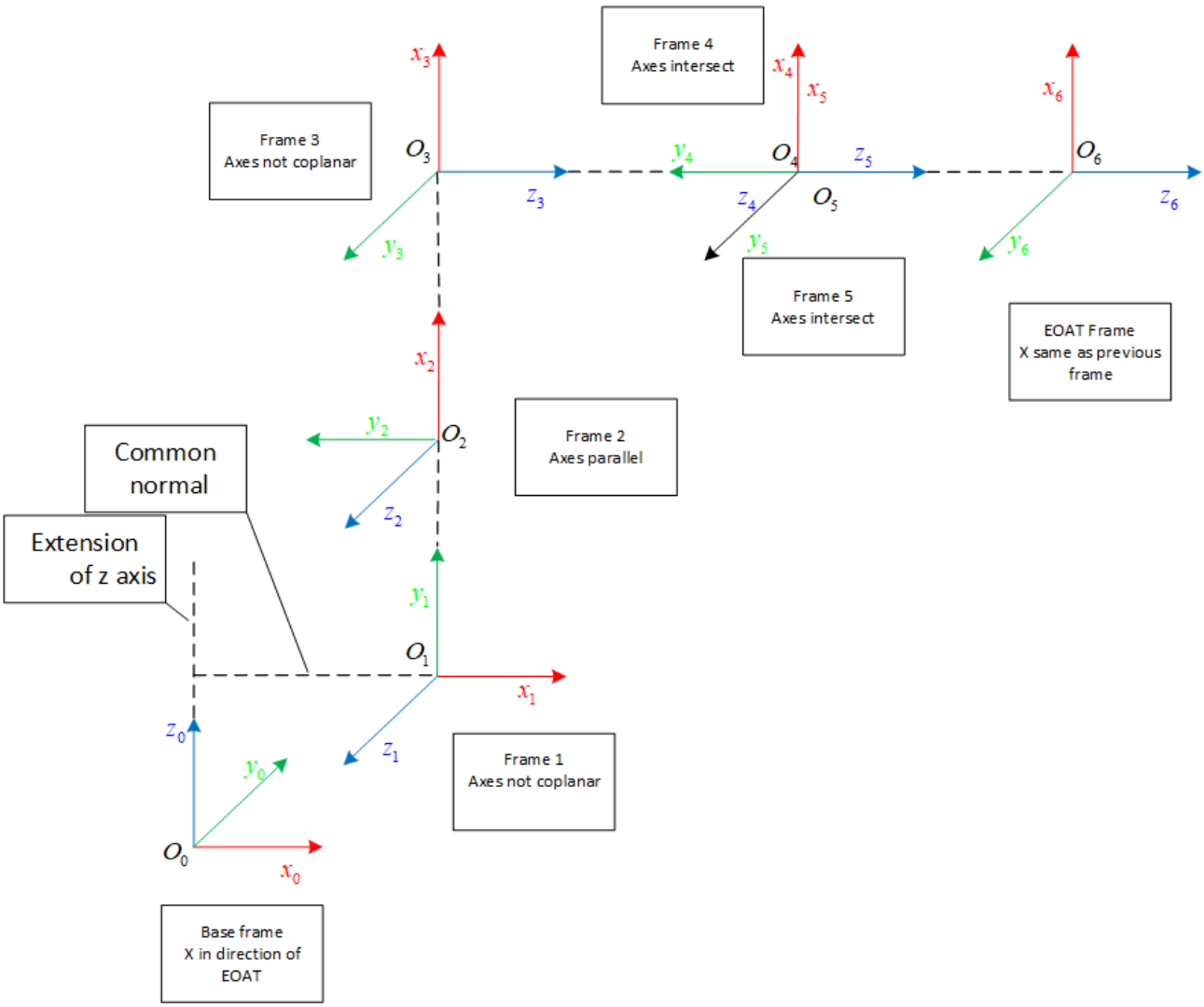

Figure A.1: Coordinate reference frames for Fanuc 210F

Steps to find individual frames The steps how to find each individual link-coordinate frame can be described as following:

Frame 0 (base) The frame mapping starts with the base frame. For the base frame the z-axis is given through joint j_1. Origin of frame 0 is put into joint 1. The x-axis is chosen to point in direction of of the End Of Arm Tooling (EOAT) in standard pose. For standard pose see 4.2 . This pose will also be chosen for alignment of all distal frames.
This assignment of the base frame can be optimized to minimize the Denavit-Hartenberg (DH)-parameters. By moving the base frame out of the joint on the same x/y-surface as frame 1, d_1 can be set to zero without

changing the kinematics. Simply an offset in the z-axis of the generalized world coordinates for the end effector is added.

Frame 1 For frame 1, the z-axes are not coplanar. Because of that, the z_0-axis is extended until a common normal can be found that intersects with z_i to define the origin O_1. x_1 departs from O_1 along the common normal. y_1 is added according to the right hand rule.

Frame 2 As seen, in image E.2, axes z_2 and z_1 are parallel. O_2 can be found at the intersection of the normal through O_1 with z_2. x_2 follows the common normal with joint 4. y_2 is added according to the right hand rule.

Frame 3 Joint 3 and joint 4 are not exactly aligned (see image 4.2), which prevents the z-axes to intersect. That's why axes z_3 and z_2 are not coplanar. z_3 runs parallel through the link 3, due to the orientation of the rotational axis. As the common normal between z_3 and z_2 defines x_3, which in turn gives O_3, the origin can be found far away from the physical position of the joint. y_3 is added according to the right hand rule.

Frame 4 Joint 5 lies in line with joint 4. As z_3 follows this line, the axes z_4 and z_3 intersect in the centre of joint 5, which gives O_4. x_4 leaves the plane spanned by z_4 and z_3 perpendicular. The positive direction of x_4 is chosen to be similar as in frame 3 for simplicity. y_4 is added according to the right hand rule and points in direction of frame 3.

Frame 5 As z_5 lies in line with O_4, the axes z_5 and z_4 intersect in O_4 which puts O_5 at the same position as O_4. x_5 leaves the plane spanned by z_5 and z_4 perpendicular. The positive direction of x_5 is chosen to be similar as in frame 4 for simplicity, which puts x_5 and x_4 on top of each other. y_5 is added according to the right hand rule and as z_5 is turned 90 degrees relative to z_4 around $x_{4,5}$, y_5 is turned 90 degrees relative to z_4 around $x_{4,5}$ as well.

Frame 6 (EOAT) z_6 lies in line with z_5 and is consequently parallel. as there are no distal joints, to reference x_6, it can be chosen arbitrarily. For simplicity, it is chosen to be similar as in frame 5. y_6 is added according to the right hand rule.

B. DH-Convention

The DH-Convention is a commonly used and simplifies the forward and backward transformation. It was named after Jaques Denavit and Richard Hartenberg who developed a general theory to describe a serial link mechanism. [12]
It consists of following parts:

- DH-Convention for establishing the coordinate systems
- DH-Transformation for generation of the coordinate systems
- DH-Parameters as a result form the transformations

Determining the coordinate systems is done according to set rules. Nevertheless, the choices of coordinate frames are also not unique, so different people will derive different, but correct frame assignments. This freedom of choice should be used to bring as many DH-Parameters as possible to zero. This simplifies subsequent equations and calculations. [61]
Each joint of the robot is described by four parameters.
This leads to a kinematic chain with each frame determined by the previous one [61].

Link0 - Link1 - Link2 - Link3 - Link4 - Link5 - Link6

Each DH-transformation consists of four elementary transformations[61]:

1) rotation around the x_i-axis with the amount of α_i

2) translation along x_i-axis with the amount of a_i

3) translation along z_i-axis with the amount of d_i

4) rotation around z_i-axis with the amount of θ_i

This shows that the DH robotic convention is a minimal line representation, as with four parameters, all possible lines in the Euclidean Space can be represented ([5], page 210).
Following from this, two pairs of parameters determine the joints and links [10]:

Links: represented by link length (a) and link twist (α)

Joints: represented by link offset (d) and joint angle (θ)

C. Assignation of Frames

To give a better understanding of the assignation of frames, a visual representation for the three cases can be found here:

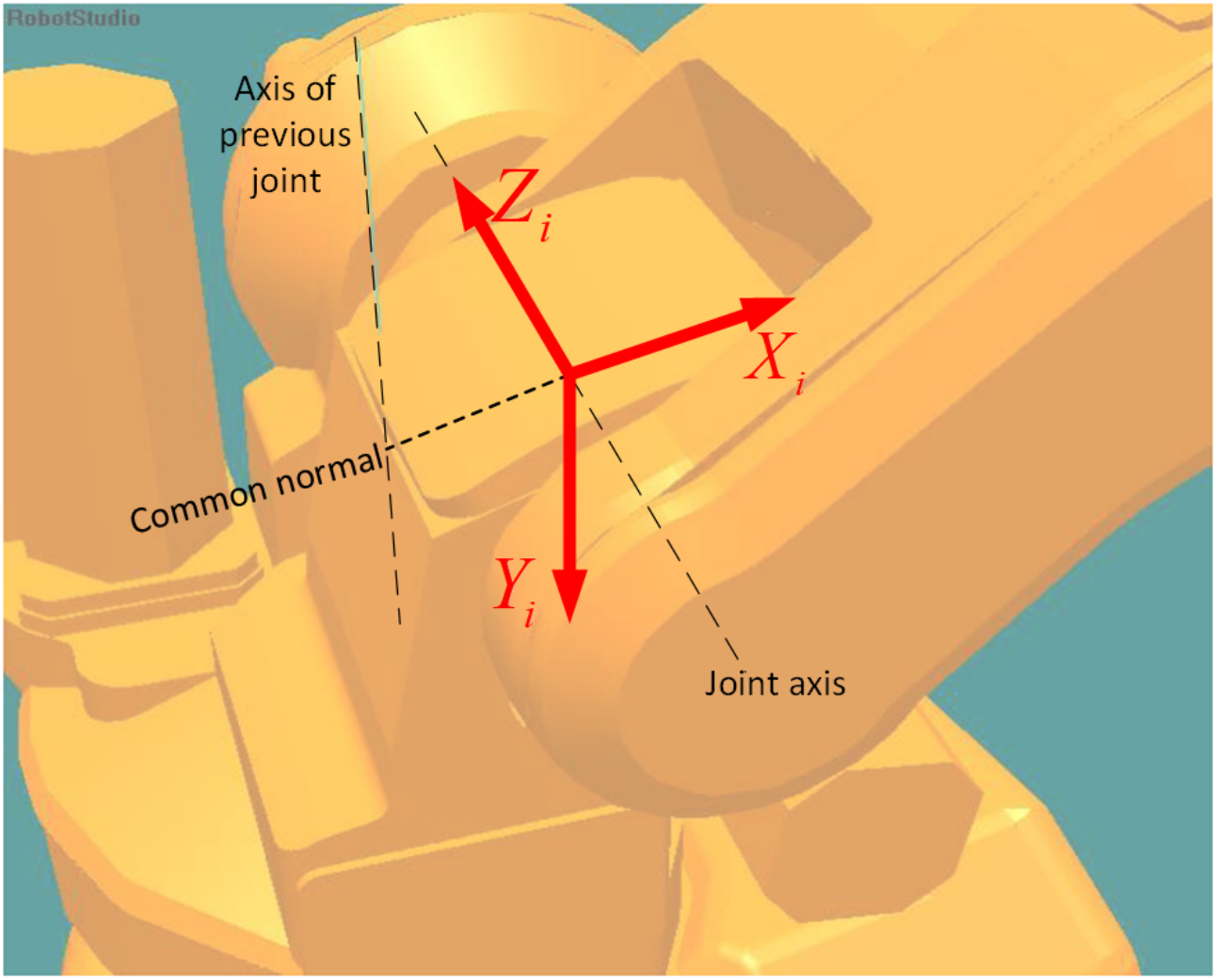

Figure C.1: Example for non coplanar axes [61]

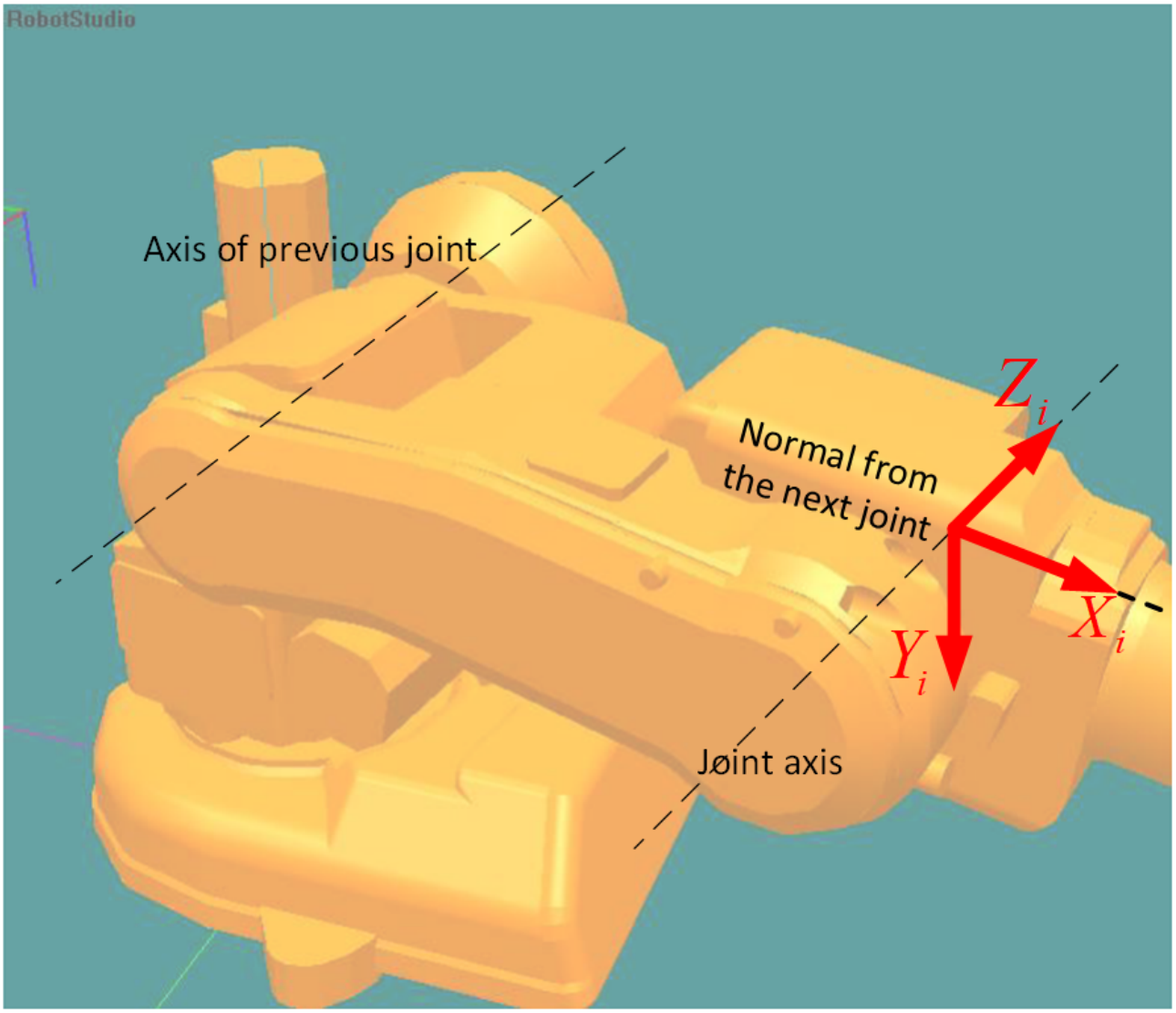

Figure C.2: Example for parallel axes [61]

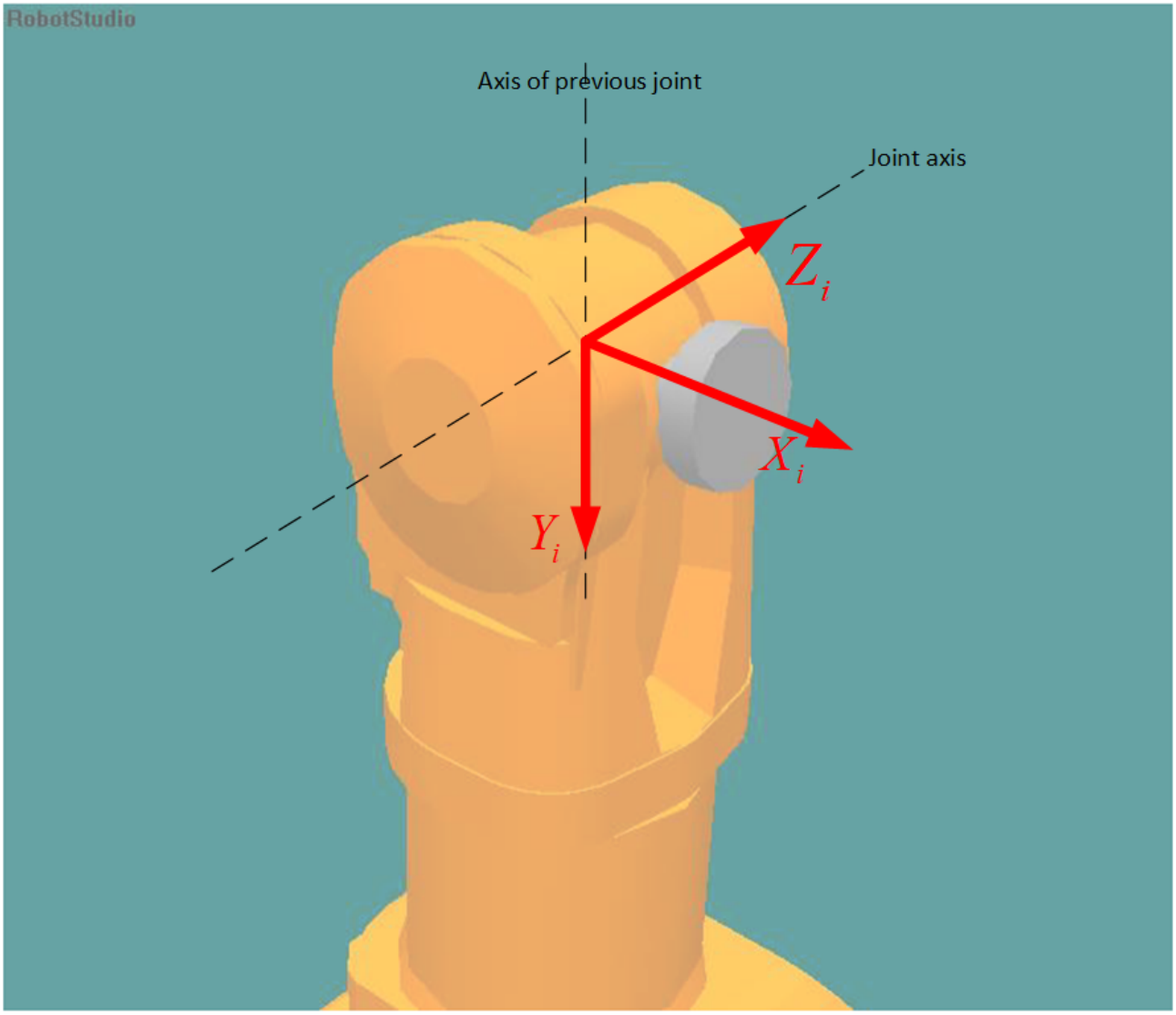

Figure C.3: Example for intersecting axes [61]

D. Robot Configurations

As mentioned before, a 6 DOF manipulator can reach the same position in multiple configurations. The different configurations in the solutions can be seen in the example of the FANUC robot arms (see figure D.1)

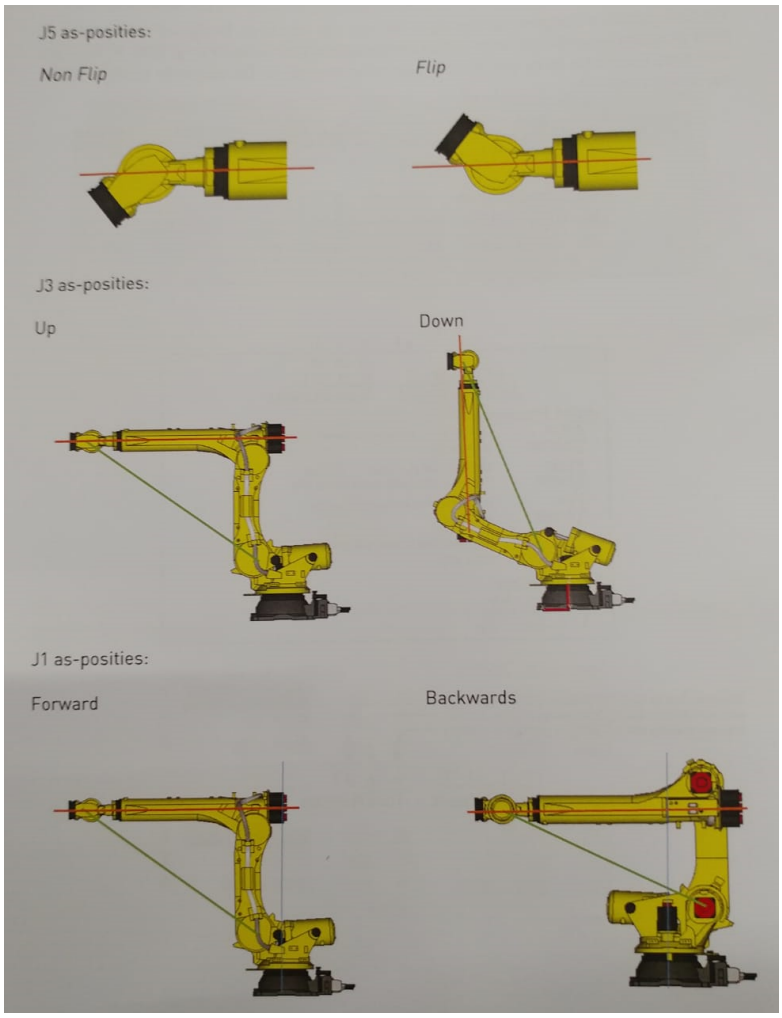

Figure D.1: 6 axis robot configurations on the example of a FANUC Robot [7]

There exist $2^3 = 8$ solutions for the robot arm.

E. Joints and links

Joint and link numbering in Fanuc 210F The numbering scheme presented in subsubsection 4.1.1.1 Numbering of Joints and Links can be applied to the Fanuc 210F (see fig. E.1).

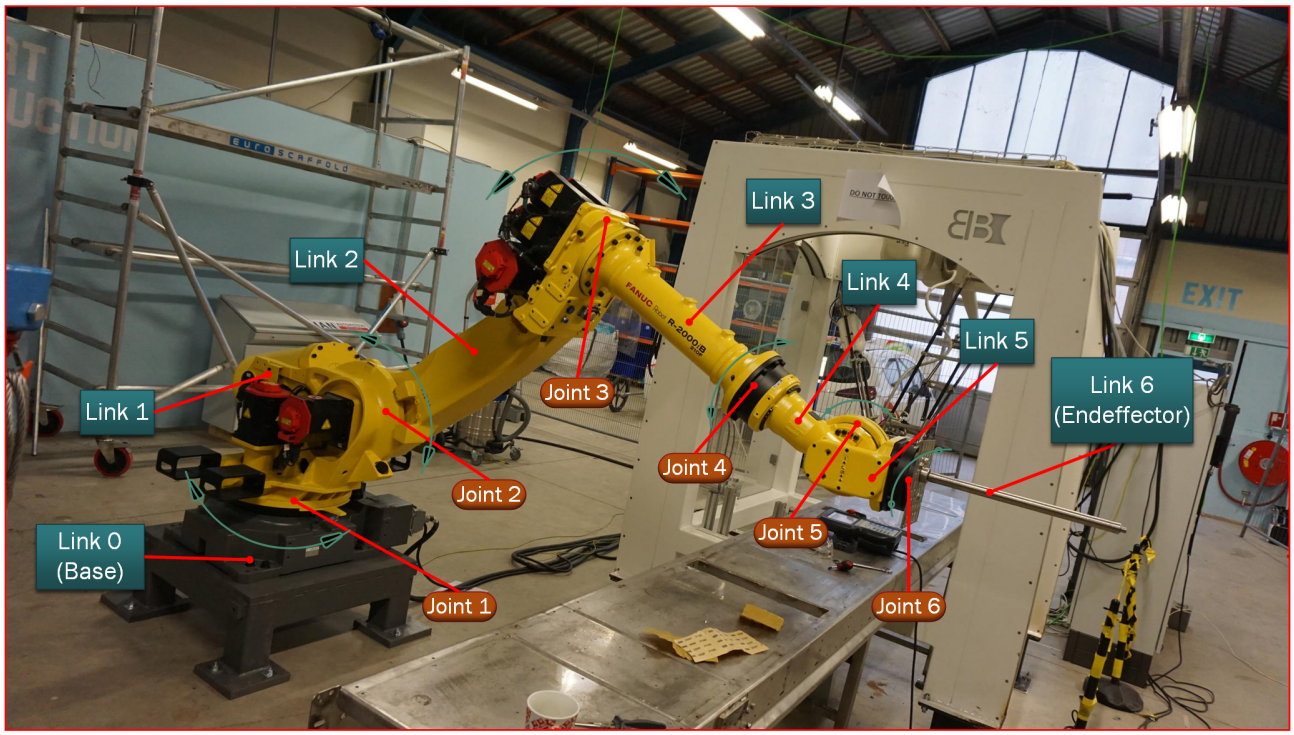

Figure E.1: Links (turquoise) and joints (orange) in the FANUC 210F

z_i axes in Fanuc 210F As described in section 4.1.1.2 Assignation of z_i axes, the z_i axes can be attached to the Fanuc 210F (see figure E.2).

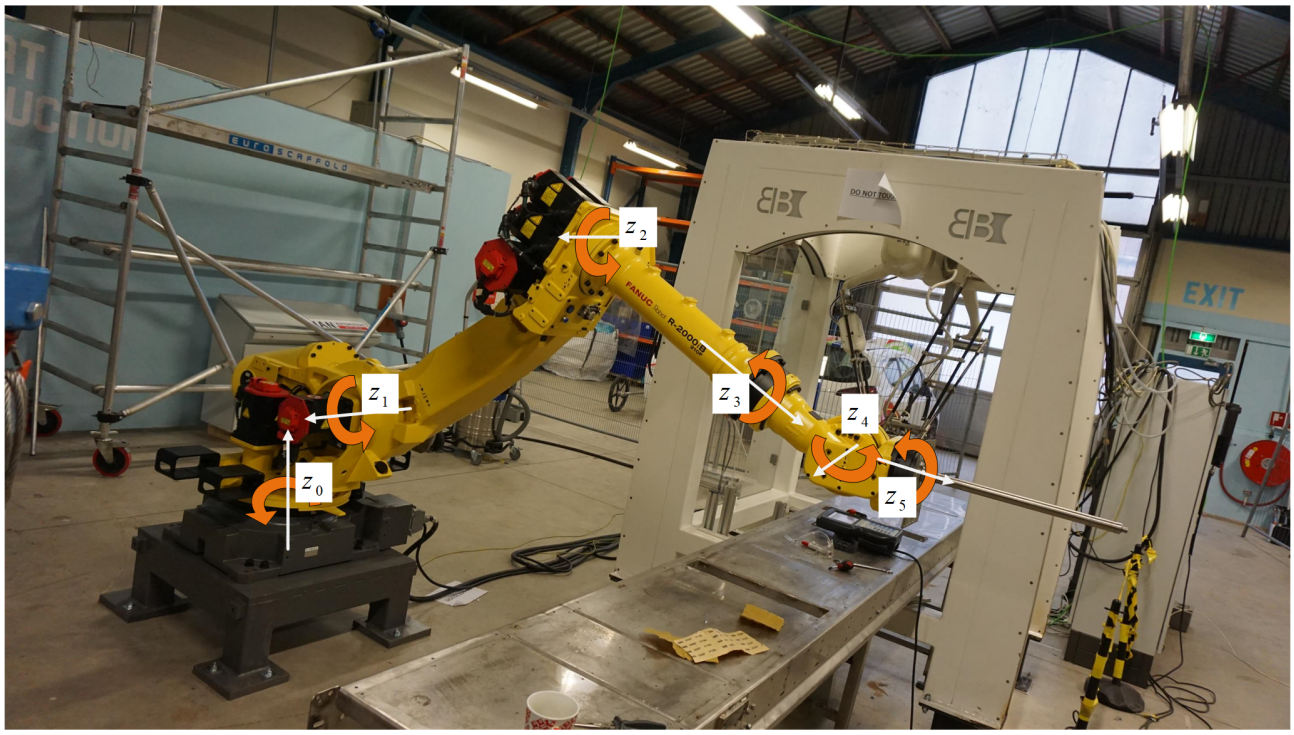

Figure E.2: z_i axes on the Fanuc 210F with the direction of positive rotation (orange)

The positive direction of the triples z_1, z_2, z_4 and z_0, z_3, z_5, was chosen to make sure, the x-axis would always have the same direction for parallel joints.

F. Sherical vs non-spherical wrist robots

Robots with a spherical axis configuration usually have three rotating wrist axes that intersect at one point while non-spherical configurations have a shifted wrist axis, see figure F.1. This leads to different solutions in the inverse kinematics. Spherical wrist robots have up to 8 solutions for the same endeffector position while non-spherical have 9. Spherical wrist robots have a simpler inverse kinematic solution, while suffering from more singularities. [19]

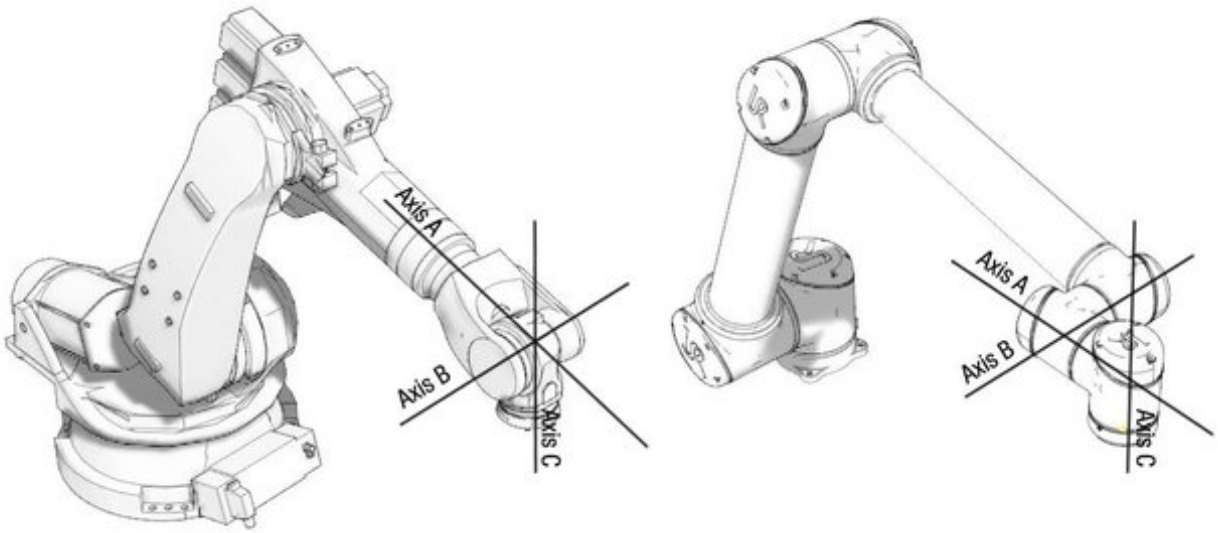

Figure F.1: Spherical vs non-spherical wrist robots [19]

G. Intermediate Steps in Forming Forward Kinematic Equations

The complete transformation matrix can be found with:

$$
{}^{0}T_{6} = \begin{bmatrix} \cos\theta_1 & -\sin\theta_1 \cdot \cos\alpha_1 & \sin\theta_1 \cdot \sin\alpha_1 & a_1 \cdot \cos\theta_1 \\ \sin\theta_1 & \cos\theta_1 \cdot \cos\alpha_1 & -\cos\theta_1 \cdot \sin\alpha_1 & a_1 \cdot \sin\theta_1 \\ 0 & \sin\alpha_1 & \cos\alpha_1 & d_1 \\ 0 & 0 & 0 & 1 \end{bmatrix} \cdot \begin{bmatrix} \cos\theta_2 & -\sin\theta_2 \cdot \cos\alpha_2 & \sin\theta_2 \cdot \sin\alpha_2 & a_2 \cdot \cos\theta_2 \\ \sin\theta_2 & \cos\theta_2 \cdot \cos\alpha_2 & -\cos\theta_2 \cdot \sin\alpha_2 & a_2 \cdot \sin\theta_2 \\ 0 & \sin\alpha_2 & \cos\alpha_2 & d_2 \\ 0 & 0 & 0 & 1 \end{bmatrix} \cdot \begin{bmatrix} \cos\theta_3 & -\sin\theta_3 \cdot \cos\alpha_3 & \sin\theta_3 \cdot \sin\alpha_3 & a_3 \cdot \cos\theta_3 \\ \sin\theta_3 & \cos\theta_3 \cdot \cos\alpha_3 & -\cos\theta_3 \cdot \sin\alpha_3 & a_3 \cdot \sin\theta_3 \\ 0 & \sin\alpha_3 & \cos\alpha_3 & d_3 \\ 0 & 0 & 0 & 1 \end{bmatrix} \cdot \begin{bmatrix} \cos\theta_4 & -\sin\theta_4 \cdot \cos\alpha_4 & \sin\theta_4 \cdot \sin\alpha_4 & a_4 \cdot \cos\theta_4 \\ \sin\theta_4 & \cos\theta_4 \cdot \cos\alpha_4 & -\cos\theta_4 \cdot \sin\alpha_4 & a_4 \cdot \sin\theta_4 \\ 0 & \sin\alpha_4 & \cos\alpha_4 & d_4 \\ 0 & 0 & 0 & 1 \end{bmatrix} \cdot \begin{bmatrix} \cos\theta_5 & -\sin\theta_5 \cdot \cos\alpha_5 & \sin\theta_5 \cdot \sin\alpha_5 & a_5 \cdot \cos\theta_5 \\ \sin\theta_5 & \cos\theta_5 \cdot \cos\alpha_5 & -\cos\theta_5 \cdot \sin\alpha_5 & a_5 \cdot \sin\theta_5 \\ 0 & \sin\alpha_5 & \cos\alpha_5 & d_5 \\ 0 & 0 & 0 & 1 \end{bmatrix} \cdot \begin{bmatrix} \cos\theta_6 & -\sin\theta_6 \cdot \cos\alpha_6 & \sin\theta_6 \cdot \sin\alpha_6 & a_6 \cdot \cos\theta_6 \\ \sin\theta_6 & \cos\theta_6 \cdot \cos\alpha_6 & -\cos\theta_6 \cdot \sin\alpha_6 & a_6 \cdot \sin\theta_6 \\ 0 & \sin\alpha_6 & \cos\alpha_6 & d_6 \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{G.1}
$$

This set of matrices can be filled with the numerical values from Table 4.2 Numeric values of Denavit Hartenberg Parameters for Fanuc 210F

$$
{}^{0}T_{6} =
\begin{bmatrix}
\cos\theta_1 & -\sin\theta_1 * \cos(\pi/2) & \sin\theta_1 * \sin(\pi/2) & 312 * \cos\theta_1 \\
\sin\theta_1 & \cos\theta_1 * \cos(\pi/2) & -\cos\theta_1 * \sin(\pi/2) & 312 * \sin\theta_1 \\
0 & \sin(\pi/2) & \cos(\pi/2) & 0 \\
0 & 0 & 0 & 1
\end{bmatrix} *
\begin{bmatrix}
\cos\theta_2 & -\sin\theta_2 * \cos(0) & \sin\theta_2 * \sin(0) & 1075 * \cos\theta_2 \\
\sin\theta_2 & \cos\theta_2 * \cos(0) & -\cos\theta_2 * \sin(0) & 1075 * \sin\theta_2 \\
0 & \sin(0) & \cos(0) & 0 \\
0 & 0 & 0 & 1
\end{bmatrix} *
\begin{bmatrix}
\cos\theta_3 & -\sin\theta_3 * \cos(\pi/2) & \sin\theta_3 * \sin(\pi/2) & 225 * \cos\theta_3 \\
\sin\theta_3 & \cos\theta_3 * \cos(\pi/2) & -\cos\theta_3 * \sin(\pi/2) & 225 * \sin\theta_3 \\
0 & \sin(\pi/2) & \cos(\pi/2) & 0 \\
0 & 0 & 0 & 1
\end{bmatrix} *
\begin{bmatrix}
\cos\theta_4 & -\sin\theta_4 * \cos(-\pi/2) & \sin\theta_4 * \sin(-\pi/2) & 0 * \cos\theta_4 \\
\sin\theta_4 & \cos\theta_4 * \cos(-\pi/2) & -\cos\theta_4 * \sin(-\pi/2) & 0 * \sin\theta_4 \\
0 & \sin(-\pi/2) & \cos(-\pi/2) & 1280 \\
0 & 0 & 0 & 1
\end{bmatrix} *
\begin{bmatrix}
\cos\theta_5 & -\sin\theta_5 * \cos(\pi/2) & \sin\theta_5 * \sin(\pi/2) & 0 * \cos\theta_5 \\
\sin\theta_5 & \cos\theta_5 * \cos(\pi/2) & -\cos\theta_5 * \sin(\pi/2) & 0 * \sin\theta_5 \\
0 & \sin(\pi/2) & \cos(\pi/2) & 0 \\
0 & 0 & 0 & 1
\end{bmatrix} *
\begin{bmatrix}
\cos\theta_6 & -\sin\theta_6 * \cos(0) & \sin\theta_6 * \sin(0) & 0 * \cos\theta_6 \\
\sin\theta_6 & \cos\theta_6 * \cos(0) & -\cos\theta_6 * \sin(0) & 0 * \sin\theta_6 \\
0 & \sin(0) & \cos(0) & 235 \\
0 & 0 & 0 & 1
\end{bmatrix}
\tag{G.2}
$$

With some simplification this results in the following equation:

$$
{}^{0}T_{6} =
\begin{bmatrix}
\cos\theta_1 & 0 & \sin\theta_1 & 312 * \cos\theta_1 \\
\sin\theta_1 & 0 & -\cos\theta_1 & 312 * \sin\theta_1 \\
0 & 1 & 0 & 0 \\
0 & 0 & 0 & 1
\end{bmatrix} *
\begin{bmatrix}
\cos\theta_2 & -\sin\theta_2 * & 0 & 1075 * \cos\theta_2 \\
\sin\theta_2 & \cos\theta_2 & 0 & 1075 * \sin\theta_2 \\
0 & 0 & 1 & 0 \\
0 & 0 & 0 & 1
\end{bmatrix} *
\begin{bmatrix}
\cos\theta_3 & 0 & \sin\theta_3 & 225 * \cos\theta_3 \\
\sin\theta_3 & 0 & -\cos\theta_3 & 225 * \sin\theta_3 \\
0 & 1 & 0 & 0 \\
0 & 0 & 0 & 1
\end{bmatrix} *
\begin{bmatrix}
\cos\theta_4 & 0 & -\sin\theta_4 & 0 \\
\sin\theta_4 & 0 & \cos\theta_4 & 0 \\
0 & -1 & 0 & 1280 \\
0 & 0 & 0 & 1
\end{bmatrix} *
\begin{bmatrix}
\cos\theta_5 & 0 & \sin\theta_5 & 0 \\
\sin\theta_5 & 0 & -\cos\theta_5 & 0 \\
0 & 1 & 0 & 0 \\
0 & 0 & 0 & 1
\end{bmatrix} *
\begin{bmatrix}
\cos\theta_6 & -\sin\theta_6 & 0 & 0 \\
\sin\theta_6 & \cos\theta_6 & 0 & 0 \\
0 & 0 & 1 & 235 \\
0 & 0 & 0 & 1
\end{bmatrix}
\tag{G.3}
$$

H. Intermediate Steps in Forming Inverse Kinematic Equations

In a first step 4.11 is reformed into H.1.

$$ {}^{2}T_{3}\,{}^{3}T_{4}\,{}^{4}T_{5} = {}^{0}T_{6}({}^{0}T_{1})^{-1}({}^{1}T_{2})^{-1}({}^{5}T_{6})^{-1} \tag{H.1} $$

This gives 2 matrices, formed by the terms found in appendix G.1. These matrices are the terms found in H.1. With the help of MATLAB and the symbolic toolbox, the inverses and the overall product of each side can be calculated.

$$ {}^{2}T_{3}\,{}^{3}T_{4}\,{}^{4}T_{5} = $$

	1	2	3	4
1	$c_3c_4c_5 - s_3s_5$	$-c_3c_4s_5 - s_3c_5$	c_3s_4	$c_3a_3 - s_3d_4 + a_2$
2	$s_3c_4c_5 + c_3s_5$	$-s_3c_4s_5 + c_3c_5$	s_3s_4	$s_3a_3 + c_3d_4$
3	$-s_4c_5$	s_4s_5	c_4	0
4	0	0	0	1

(H.2)

$$ {}^{0}T_{6}({}^{0}T_{1})^{-1}({}^{1}T_{2})^{-1}({}^{5}T_{6})^{-1} = $$

$(c_2c_1n_x+c_2s_1n_y-s_2n_z)c_6-(c_2c_1o_x+c_2s_1o_y-s_2o_z)s_6$	$c_2c_1a_x+c_2s_1a_y-s_2a_z$	$-(c_2c_1n_x+c_2s_1n_y-s_2n_z)s_6-(c_2c_1o_x+c_2s_1o_y-s_2o_z)c_6$	$-(c_2c_1a_x+c_2s_1a_y-s_2a_z)d_6+c_2c_1p_x+c_2s_1p_y-s_2p_z-c_2a_1$
$-(s_2c_1n_x-s_2s_1n_y-c_2n_z)c_6-(-s_2c_1o_x-s_2s_1o_y-c_2o_z)s_6$	$-s_2c_1a_x-s_2s_1a_y-c_2a_z$	$-(-s_2c_1n_x-s_2s_1n_y-c_2n_z)s_6-(-s_2c_1o_x-s_2s_1o_y-c_2o_z)c_6$	$-(-s_2c_1a_x-s_2s_1a_y-c_2a_z)d_6-s_2c_1p_x-s_2s_1p_y-c_2p_z+s_2a_1$
$(-s_1n_x+c_1n_y)c_6-(-s_1o_x+c_1o_y)s_6$	$-s_1a_1+c_1a_2$	$-(-s_1n_x+c_1n_y)s_6-(-s_1o_x+c_1o_y)c_6$	$-(-s_1a_x+c_1a_y)d_6-s_1p_x+c_1p_y$
0	0	0	1

(H.3)

In equations H.2 H.3 $sin(\theta_i)$ and $\cos(\theta_i)$ have been replaced by s_i and c_i to allow for appropriate presentation of the matrix. This abbreviation will be used for following steps as well.

As shown by H.1, the elements of matrices H.2 H.3 are corresponding equal. This allows to find analytic expressions for all θ_i.

θ_1: By taking the elements (3,4) of H.2 H.3 as corresponding equal, equation H.4 is formed.

$$ -(-s_1a_x + c_1a_y)d_6 - s_1p_x + c_1p_y = 0 \tag{H.4} $$

Reforming equation H.4 gives a solution for θ_1 as seen in equations H.5.

$$ \theta_{1_1} = \operatorname{atan2}(p_y - a_yd_6, p_x - a_xd_6) \qquad \theta_{1_2} = \operatorname{atan2}(-p_y + a_yd_6, -p_x + a_xd_6) \tag{H.5} $$

θ_2: Taking elements (1,4) and (2,4) of H.2 H.3 as corresponding equal, H.6 and H.7 are formed.

$$a_3c_3 - d_4s_3 + a_2 = -(c_2c_1a_x + c_2s_1a_y - s_2a_z)d_6 + c_2c_1p_x + c_2s_1p_y - s_2p_z - c_2a_1 \tag{H.6}$$

$$a_3s_3 + d_4c_3 = -(-s_2c_1a_x - s_2s_1a_y - c_2a_z)d_6s_2c_1p_xs_2s_1p_y - c_2p_z + s_2a_1 \tag{H.7}$$

For simplification of the terms in equations eqs. (H.6) and (H.7), middle variables are chosen:

$$u = c_1(p_xd_6a_x) + s_1(p_y - a_yd_6) - a_1 \ ...]v = p_z - a_zd_6 \tag{H.8}$$

With these middle variables equations H.6 and H.7 can be rewritten as seen in H.9 and H.10.

$$a_3c_3 - d_4s_3 = c_2u - s_2v - a_2 \tag{H.9}$$

$$a_3s_3 + d_4c_3 = -s_u - c_2v \tag{H.10}$$

To eliminate s_3 and c_3 H.9 and H.10 are squared and then added to form H.11.

$$a_3^2 + d_4^2 = u^2 + v^2 - 2a_2c_2u + 2a_2s_2v + a_2^2 \tag{H.11}$$

$$\frac{a_3^2 + d_4^2 - u^2 - v^2 - a_2^2}{-2a_2} = c_2u - s_2v \tag{H.12}$$

This equation can be further simplified by taking another middle variable:

$$m = \frac{a_3^2 + d_4^2 - u^1 - v^2 - a_2^2}{-2a_2}$$

Reforming equation H.12 and inserting the middle variable reveals a solution for θ_2 as seen in H.13.

$$\theta_{2_{1/2}} = \operatorname{atan2}(-v, u) \pm \operatorname{atan2}(\sqrt{v^2 + u^2 - m^2}, m) \tag{H.13}$$

θ_3: To receive θ_3 it helps to take H.6 and H.7. Equation H.6 needs to be reformed into H.14.

$$a_cc_3d_4s_3 + a_2 = c_2u - s_2v \tag{H.14}$$

Squaring of H.14 and H.7 and adding them eliminates s_2 and c_2 as seen in H.15.

$$a_2^2 + a_3^2 + d_4^2 + 2a_2(a_3c_3 - d_4s_3) = u^2 + v^2 \tag{H.15}$$

$$a_3c_3 - d_4s_3 = \frac{a_3^2 + d_4^2 - u^2 - v^2 + a_2^2}{2a_2} \tag{H.16}$$

Again, this equation can be further simplified by taking another middle variable:

$$-\frac{a_3^2 + d_4^2 - u^2 - v^2 + a_2^2}{2a_2}$$

Reforming equation H.16 and inserting the middle variable reveals a solution for θ_3 as seen in eq. (H.17).

$$\theta_{3_{1/2}} = \operatorname{atan2}(-d_4, a_3) \pm A\tan(2(\sqrt{d_4^2 + a_3^2 - h^2}, h)) \tag{H.17}$$

θ_5: Taking elements (2,2) and (1,2) of H.2 and H.3 as corresponding equal, H.18 and H.19 are formed.

$$-s_3c_4s_5 + c_3c_5 = -c_1s_2a_x - s_1s_2a_y - c_2a_z \tag{H.18}$$

$$-c_3c_4s_5 - s_3c_4 = c_1c_2a_x + s_1c_2a_y - s_2a_z \tag{H.19}$$

To combine equations H.18 and H.19, H.18 is multiplied with c_3 and H.19 is multiplied by s_2. The resulting equations are then subtracted from each other giving H.20.

$$c_5 = (-c_1s_2a_x - s_1s_2a_y - c_2a_z)c_3 - (c_1c_2a_x + s_1c_2a_y - s_2a_z)s_3 \tag{H.20}$$

Reforming equation H.20 gives a solution for θ_5 as seen in equation H.21.

$$\theta_{5_{1/2}} = \pm\operatorname{acos}[(-c_1s_2a_x - s_1s_2a_y - c_2a_z)c_3 - (c_1c_2a_x + s_1c_2a_y - s_2a_z)s_3] \tag{H.21}$$

θ_4: With θ_5 and θ_{1-3} known, equation H.19 can be reformed into a solution for θ_4 as seen in equation H.22.

$$\theta_{4_{1/2}} = \pm \arccos[\frac{c_1c_2a_x + s_1c_2a_y - s_2a_z + s_3c_5}{-c_3s_5}] \tag{H.22}$$

θ_6: By taking the elements (3,3) of eqs. (H.2) and (H.3) as corresponding equal, equation H.23 is formed.

$$s_6(s_1n_x - c_1n_y) + c_6(s_1o_x - c_1o_y) = c_4 \tag{H.23}$$

With θ_4 and θ_1 known, equation H.23 can be reformed into a solution for θ_6 as seen in equation H.24.

$$\theta_{6_{1/2}} = \text{atan2}(s_1n_x - c_1n_y, s_1o_x - c_1o_y) \pm \text{atan2}(\sqrt{(s_1n_x - c_1n_y)^2 + (s_1o_x - c_1o_y)^2 - c_4^2}, c_4) \tag{H.24}$$

With 4.12,4.13,4.14,4.16,4.15 and 4.17 the values for $(\theta_1, \theta_2, \theta_3, \theta_4, \theta_5, \theta_6)$ can be calculated for all reachable end effector positions.

For details on the atan2 function, see the glossary.

I. Matlab Code

The commented code to build and test a model of the 210F in matlab can be found below.

```
%% init
close all
clear all
clc

import +ETS3.*

%% Robot building
fprintf('Build the robot')

%Define a few positions:
qz=[0 0 0 0 0 0]; %ZERO angle
qr=[0 0 pi/2 0 0 0]; %READY, straight and vertical
qs=[0 pi/2 pi/2 0 0 0]; %STRECH, straight and horizontal
qn=[0 pi/4 0 0 pi/4 0]; %NOMINAL, the arm is in a dextrous working pose
%Matrix of predefined standard positions
POS=[qz; qr; qs; qn];
POS_name=["ZERO";"READY";"STRETCH";"NOMINAL"];

%define DH parameters with the links [METRIC]
%Link      Thet d    a     Alp   r/p offs
L(1)=Link([0     0      0.312 pi/2  0   0   ],'standard');
L(2)=Link([0     0      1.075 0     0   pi/2],'standard');
L(3)=Link([0     0      0.225 pi/2  0   0   ],'standard');
L(4)=Link([0     1.280  0      pi/2  0   0   ],'standard');
L(5)=Link([0     0      0     pi/2  0   0   ],'standard');
L(6)=Link([0     0.235  0      0     0   0   ],'standard');

%Set joint limits
Rot_freedom=[360; 136; 362; 720; 250; 720]; %rotational freedom in degrees
Limits=[ Rot_freedom/2 Rot_freedom/2]; % calculate upper and lower limits in
    degrees
Lim=Limits*pi/180; %convert to radians
%assign values to joints
L(1).qlim = Lim(1,:);
%Manually adjusted  took maximum rotational freedom and adjusted for
%hoizontal alginment
L(2).qlim = [ pi/2 ( pi/2+Rot_freedom(2)*pi/180)]; %Lim(2,:);
%Manually adjusted  same freedom in both directions without changing
%offset
L(3).qlim = [Lim(3,1)+pi/2 Lim(3,2)+pi/2];
L(4).qlim = Lim(4,:);
L(5).qlim = Lim(5,:);
L(6).qlim = Lim(6,:);
```

```
%create robot object
R= SerialLink(L, 'name', '210F', 'comment', 'six DOF', 'manufacturer', 'FANUC');
    %Add base transform if necessary with 'base' and tool transformation with '
    tool'

%Put it on a base [METRIC]
R.base = SE3(0, 0, 0.670);
%Turn Robot upside down
%R.base =  SE3(0,0,670) * SE3.Rx(pi);

%Add a tool [METRIC]
toolLength=0.658; %See CAD drawing from Laurence for measurements
R.tool =SE3(0, 0, toolLength); %(0 0  {SE3.rand OR toolLength OR 0})



%Plot the robot
figure(1);
R.plot(qz)
%create teach pendant
R.teach
%output some infos
%r.structure %doesn't work

numoflinks=size(R.links,2);
fprintf('Number of links:\n%d',numoflinks)

fprintf('Is robot of configuration RRRRRR ? (1=Y, 0=N)')
R.isconfig('RRRRRR') %returns 1, if joint config string is right
fprintf('Are all joints of revolute type ? (1=Y, 0=N)')
R.isrevolute %returns 1, for all joints that are revolute joints

R.fast=1; %set to 1 for fast RNE

%%Showall
fprintf('Overview of the robot parameters:\n')
R.display

%Break
fprintf('press any key to continue...\n')
pause();
%% Forward kinematics
fprintf('forward Kinematics')

for i=1:size(POS,1)
fprintf('Robot in position %s :\n', POS_name(i,:));
T(i)=R.fkine(POS(i,:));
disp(T(i));
figure(1);
R.plot(POS(i,:));
pause(1);
end



%Break
fprintf('press any key to continue...\n')
pause();
%% Inverse Kinematics

fprintf('inverse kinematics')
```

```
%Set robot type
R.ikineType = 'nooffset';

%Configurations:
cl='l'; %LEFT handed
cr='r'; %Right handed
cu='u'; %elbow UP
cd='d'; %elbow DOWN
cf='f'; %wrist FLIPPED
cn='n'; %wrist NOT flipped
%List of all configs:
% configurations=[[cl,cr],[cu,cd],[cf,cn]];
conf=[
cl cu cf
cr cu cf
cl cd cf
cr cd cf
cl cu cn
cr cu cn
cl cd cn
cr cd cn
];

fprintf('Inverse kinematics for all defined standard positions in all
    configurations: \n')
for i = 1 : length(conf)
%Sort by configuration iteration
% TI(1,:,i)=R.ikine6s(T(1),conf(i,:))'
% TI(2,:,i)=R.ikine6s(T(2),conf(i,:))'
% TI(3,:,i)=R.ikine6s(T(3),conf(i,:))'
% TI(4,:,i)=R.ikine6s(T(4),conf(i,:))'

%sort by pose
TI(i,:,1)=R.ikine6s(T(1),conf(i,:))'
TI(i,:,2)=R.ikine6s(T(2),conf(i,:))'
TI(i,:,3)=R.ikine6s(T(3),conf(i,:))'
TI(i,:,4)=R.ikine6s(T(4),conf(i,:))'
end

%should deliver same endeffector position, but doesn't!!! OMG
for i= 1:length(TI(:,:,1))
figure(1);
R.plot(TI(i,:,1))
pause(1)
end
fprintf('This gives strange results, as all should point to the same endeffector
    position... \n \n')


%Same test again for single position, no special matrices, direct feeding
fprintf('Now a small test for a single position with ikine6s: \n')
for i = 1 : length(conf)
TI1(i,:)=R.ikine6s(R.fkine(qr),conf(i,:))'
TF1(i,:)=R.fkine(TI1(i,:))
end
%should be the same as
fprintf('should be the same as: \n')
R.fkine(qr)
%Why no match???

% One of these should be 1!!
```

```
166 fprintf('comparing the previous two results shold deliver at least one match:\n'
    )
167 for i= 1:size(TI,1)
168 TI(i,:,1) == T(1)
169 end
170
171 fprintf('Numerical solution:\n')
172 %Numerical solution works, but is not precise
173 R.fkine(R.ikine(R.fkine(qr)))
174 fprintf('should equal:\n')
175 %equeals
176 R.fkine(qr)
177
178 fprintf('inverse kinematics with ikcon')
179 %inverse kinematics using optimisation with joint limits   This one works
180 %perfect
181 figure(1);
182 R.plot(qz)
183 pause(1);
184 R.plot(R.ikcon(R.fkine(qz)))
185
186 fprintf('testing unreachable position:\n')
187 % Testing unreachable Position
188 [Q,ERR,EXITFLAG,OUTPUT]=R.ikcon(SE3(30000,0,0))
189 %doesn't throw proper error message, but at least gives error output
190 fprintf('does not throw proper error message, but at least gives error output\n'
    )
191 %List of inverse kinematic functions
192 % ikine6s       inverse kinematics for 6 axis spherical wrist revolute robot
193 % ikine         inverse kinematics using iterative numerical method
194 % ikunc         inverse kinematics using optimisation
195 % ikcon         inverse kinematics using optimisation with joint limits
196 % ikine_sym     analytic inverse kinematics obtained symbolically
197
198 %What *** is wrong with ikine6s? all other solutions work
199 % R.plot(qn)
200 % R.plot(R.ikcon(R.fkine(qn)))
201 % R.plot(R.ikunc(R.fkine(qn)))
202 % R.plot(R.ikine(R.fkine(qn)))
203 % R.plot(R.ikine6s(R.fkine(qn)))
204 % R.plot(R.ikine6s(R.fkine(qn),'luf'))
205 % R.plot(R.ikine6s(R.fkine(qn),'ruf'))
206 % R.plot(R.ikine6s(R.fkine(qn),'ldf'))
207 % R.plot(R.ikine6s(R.fkine(qn),'rdf'))
208 % R.plot(R.ikine6s(R.fkine(qn),'lun'))
209 % R.plot(R.ikine6s(R.fkine(qn),'run'))
210 % R.plot(R.ikine6s(R.fkine(qn),'ldn'))
211 % R.plot(R.ikine6s(R.fkine(qn),'rdn'))
212
213 %Break
214 fprintf('press any key to continue...\n')
215 pause();
216 %% Trajectories
217 fprintf('trajectories')
218
219 % Joint space motion
220
221 %Define two positions in xy plane [METRIC]
222 T1 = SE3(1.5,  1, 0) * SE3.Rx(pi);
223 T2 = SE3(1.5,  1, 0) * SE3.Rx(pi/2);
224 %joint coordinate vectors:
```

```
q1 = R.ikcon(T1);
q2 = R.ikcon(T2);
%Motion should occur over a time period of 2 sec. in 50ms time steps
t = [0:0.05:2]';
% For mtraj and jtraj the ?nal argument can be a time vector, as here, or an integer
% specifying the number of time steps.
% q = mtraj(@tpoly, q1, q2, t);
% q = mtraj(@lspb, q1, q2, t);
%For convenience use jtraj
% q = jtraj(q1, q2, t);
%joint velocity and acceleration vectors:
[q,qd,qdd] = jtraj(q1, q2, t);
%Alternatively, a jatraj method is provided by SerialLink class, but only
%provides position output
% q=R.jtraj(T1,T2,t)
%animate the output:
figure(1);
R.plot(q)
%plot the angles versus time
figure();
qplot(t, q);
%plot the velocities versus time
figure();
qplot(t, qd);
%plot the accelerations versus time
figure();
qplot(t, qdd);
%determine movement of endeffector in cartesian space
clear T
T=R.fkine(q);
p=T.transl;
about(p)
%plot the path in 2D (x,y with z assumed as neglible) and 3D
figure();
plot(p(:,1), p(:,2))
plot3(p(:,1), p(:,2), p(:,3))
%  orientation of the end effector, in XYZ roll pitch yaw angle form
plot(t, T.torpy('xyz'))

%pause();
% Cartesian motion
%where straight line motion in caresian space is reqired. only accepts number of time steps
Ts = ctraj(T1, T2, length(t));
%joint space trjaectory
qc = R.ikine(Ts)
%animate the path
figure(1);
R.plot(qc)
%plot the path
figure();
plot(t, Ts.transl);
%Break
fprintf('press any key to continue...\n')
pause();
%% Timeseries
% q=[       0         0         0         0         0         0
%           0    0.0365    0.0365         0         0         0
%           0    0.2273    0.2273         0         0         0
%           0    0.5779    0.5779         0         0         0
```

```
%           0     0.9929    0.9929          0          0          0
%           0     1.3435    1.3435          0          0          0
%           0     1.5343    1.5343          0          0          0
%           0     1.5708    1.5708          0          0          0 ]
%
% T=R.fkine(q)
%
% about T
%

% %% Jacobian
% fprintf('Jacobian translational part (upper matrix):\n')
% R.jacobe(qz,'trans')
% fprintf('Jacobian rotational part (lower matrix):\n')
% R.jacobe(qz,'rot')



%% Dynamic Modelling

% Volumentric Estimation of Linkweight

%Alternative: STL file volume estimation. Too big to be implemented now.
%See here for more:
%https://de.mathworks.com/matlabcentral/fileexchange/26982 volume of a surface
    triangulation



% Assumed parameter values:
%syms MROB L_H1 L_H2 L_H3 L_H4 L_H5 L_H6 L_R1 L_R2 L_R3 L_R4 L_R5 L_R6 L_D1 L_D2
     L_D3 L_D4 L_D5 L_D6 L_W1 L_W2 L_W3 L_W4 L_W5 L_W6 %integer positive
Mtot=   1170 %[Kg] [METRIC] %Total mass of Robot according to datasheet
L_h= [0.280     0.600       0.310     1.280      0.240     0.50     ]%LinkHeight
    (z)
L_r= [0.600/2   0.600/2     0.310/2   0.180/2    0.235/2   0.150/2  ]%LinkRadius
L_d= [0.600     0.670 0.280 0.310     0.180      0.240     0.150    ]%LinkDepth (
    y)
L_w= [0.600     0.600       1.075     0.180      0.230     0.150    ]%LinkWidth (
    x)
L_sh=[1         1           1         0          1         0        ]%Shape (0=
    cyl 1=cube 1<zeroMass 0>error)

%Volume of links based on cylinder model
for i=1:numoflinks
h=L_h(i)    %height
r=L_r(i)    %radius
Vcyl(i)=r^2*h*pi
end

%Volume of links based on cuboid model
for i=1:numoflinks
h=L_h(i)    %height
d=L_d(i)    %depth
w=L_w(i)    %width
Vcube(i)=h * d * w
end

%Total volume based on shape vector:
Vtot=0
```

```
for i=1:numoflinks
if L_sh(i)==0
Vtot=Vtot+ Vcyl(i)
elseif L_sh(i)==1
Vtot=Vtot+ Vcube(i)
else
Vtot=Vtot
end
end

%Create link mass based on shape vector
for i=1:numoflinks
if L_sh(i)==0
Mlink(i)=Mtot*Vcyl(i)/Vtot
elseif L_sh(i)==1
Mlink(i)=Mtot*Vcube(i)/Vtot
else
Mlink(i)=0
end
end

% %Mass of links based on cylinder model
% for i=1:numoflinks
%     Mcyl(i)=Mtot*Vcyl(i)/sum(Vcyl)
% end
%
% %Mass of links based on cuboid model
% for i=1:numoflinks
%     Mcube(i)=Mtot*Vcube(i)/sum(Vcube)
% end

%create innertia tensors based on shape vector
%Innertia Tensor
for i=1:numoflinks
if L_sh(i)==0
m=Mlink(i)    %mass
h=L_h(i)      %height
r=L_r(i)      %depth

Icyl=[
1/12*m*(3*r^2+h^2)    0                      0
0                     1/12*m*(3*r^2+h^2)     0
0                     0                      1/2*m*r^2
]

InTens(:,:,i)= Icyl;
elseif L_sh(i)==1

m=Mlink(i)   %mass
h=L_h(i)     %height
d=L_d(i)     %depth
w=L_w(i)     %width

Icube=[
1/12*m*(h^2+d^2)      0                      0
0                     1/12*m*(w^2+d^2)       0
0                     0                      1/12*m*(w^2+h^2)
]

InTens(:,:,i)= Icube;

```

```
400 else
401 InTens(:,:,i)=zeros(3)
402 end
403 end
404
405
406 %%calculate Inertia marix based on cylinder model
407 % for i=1:numoflinks
408 %     m=Mcyl(i)    %mass
409 %     h=L_h(i)     %height
410 %     r=L_r(i)     %depth
411 %     Icyl(i,:,:)=[
412 %         1/12*m*(3*r^2+h^2)    0                    0
413 %         0                     1/12*m*(3*r^2+h^2)   0
414 %         0                     0                    1/2*m*r^2
415 %         ]
416 % end
417 %
418 %%Calculate Inertia matrix based on cuboid model
419 % for i=1:numoflinks
420 %     m=Mcube(i)  %mass
421 %     h=L_h(i)     %height
422 %     d=L_d(i)     %depth
423 %     w=L_w(i)     %width
424 %     Icube(i,:,:)=[
425 %         1/12*m*(h^2+d^2)     0                    0
426 %         0                    1/12*m*(w^2+d^2)     0
427 %         0                    0                    1/12*m*(w^2+h^2)
428 %         ]
429 % end
430
431 %Estimation of Center of gravity
432 %R_simple=[L_w'/2 L_d'/2 L_h'/2]; %WRONG! only offset along axis
433 R_simple=[zeros(size(L_w'/2)) zeros(size(L_d'/2)) L_h'/2];
434 %Correct for DH Frame/ physical link deviation
435 R_corrected=[ %Corrective Matrix for link base displacement
436 0    0    0
437 0    0    0
438 0    0    0
439 0    0    0
440 0    0    0
441 0    0    0
442 ] + R_simple;
443
444 %Steal fricion model form other Robot.
445 %Here: puma560
446 mdl_puma560
447 for i=1:numoflinks %has to be grabbed 1 by 1
448 Lfr_stolen(i,:)=p560.links(:,i).Tc; %Linkfriction %fr_stolen because very vague
        friction
449 MB_stolen(i,:)=p560.links(:,i).B;     %Motor viscous friction
450 gears_stolen(i,:)=p560.links(:,i).G; %Gear ratio
451 MI_stolen(i,:) =p560.links(:,i).Jm;  %Motor innertia
452 end
453
454
455 %Assign parameters to Links
456 for i=1:numoflinks
457 R.links(i).m=Mlink(i);
458 R.links(i).I=InTens(:,:,i);
459 R.links(i).r=R_corrected(i,:);
```

```
460 R.links(i).Tc=Lfr_stolen(i,:);
461 R.links(i).B = MB_stolen(i,:);
462 R.links(i).G = gears_stolen(i,:);
463 R.links(i).Jm = MI_stolen(i,:);
464 end
465
466 %Display dynamics:
467 R.dyn
468
469 %Set weight of the endeffector with payload command (Endeffector is assumed
470 %massless, so combined weight of endeffector and payload is defined here
471 %at their commmon center of mass
472 toolmass=25 % Value given by Laurence
473 R.payload(toolmass, [0 0 0.5*toolLength ]); %point mass of 25 kg is assumed at
      half the lenght of tool 0.5*toolLength
474
475
476 %Forces for holding the robot in normal position:
477 Q = R.rne(POS(4,:), POS(1,:), POS(1,:)) %Nonzero due to gravity
478 %Forces necessary to hold robot arm at different steps between two positions (
      See moving gravity vector)
479 q = jtraj(POS(1,:), POS(4,:), 10)
480 Q = R.rne(q, 0*q, 0*q)
481 %Zero gravity
482 R.rne(POS(4,:), POS(1,:), POS(1,:), 'gravity', [0 0 0]) %should give zero forces
483 %Demonstrate influence of innertia on other links
484 R.rne(POS(4,:), [1 0 0 0 0 0], POS(1,:), 'gravity', [0 0 0])
485 %Demonstrate Gravity loads for different poses:
486 for i=1:size(POS,1)
487 R.gravload(POS(i,:))
488 end
489
490 %calculate the forces needed for different positions
491 [Q2,Q3] = meshgrid( pi:0.1:pi,  pi:0.1:pi);
492 for i=1:numcols(Q2),
493 for j=1:numcols(Q3);
494 g = R.gravload([0 Q2(i,j) Q3(i,j) 0 0 0]);
495 g2(i,j) = g(2);
496 g3(i,j) = g(3);
497 %Simulate them all!
498 %R.plot([0 Q2(i,j) Q3(i,j) 0 0 0])
499 end
500 end
501 figure()
502 surfl(Q2, Q3, g2);
503 figure()
504 surfl(Q2, Q3, g3);
505
506 %
507 figure(1)
508 %[t q qd] = R.nofriction().fdyn(10, [], POS(4,:));
509 [t q qd] = R.nofriction().fdyn(200, [], POS(1,:));
510 figure(1)
511 R.plot(q)
512
513
514
515 %% Jacobian
516
517 % Section of a of a hyperellipsoid in Cartesian acceleration space.
518 % The major axis of this ellipsoid is the direction in which
```

```
519 % the manipulator has maximum acceleration at this con?guration.
520
521 % pose = qs;
522 % J = R.jacob0(pose);
523 % M = R.inertia(pose);
524 % Mx = (J * inv(M) * inv(M)' * J');
525 % Mx = Mx(1:3, 1:3);
526 % figure()
527 % plot_ellipse( Mx )
528
529 %% clean up the graphs
530 %Break
531 fprintf('press any key to continue and close all figures? strg c aborts and
      leaves all figures opened...\n')
532 pause();
533 close figure 2
534 close figure 3
535 close figure 4
536 close figure 5
537 close figure 6
538 close figure 7
539 close figure 8
540 close figure 9
```

J. Digital Twinning

Additionally to the modelling and control of the robot arm, digital twinning became a research topic within the project. In cooperation with Qing, an overall control strategy based was designed in visual Components. Visual components can provide twinning services like animation and control of existent machines with a virtual model. A review of existing literature on digital twinning will be done as well.

In the setup of the experimental production line for FRP at the SPC Digital Twin (DT) has been chosen as one of the key technologies for “Smart manufacturing”. In the cooperation with Qing, there have been a few discrepancies on the understanding of a DT.

To approach this topic, the term digital twin has to be defined. This Concept was introduced for the first time by Grieves at one of his presentations about Product Lifecycle Management in 2003 at University of Michigan as a virtual representation of what has been produced [23]. As stated by Qinglin Qi [37], a DT is a high fidelity virtual model for physical objects in a digital way to simulate their behaviour. When looking on the Wikipedia page on digital twinning [17], a wide variety of definitions can be found. This shows, that the concept of the DT is not yet clearly defined. All of these definitions have in common though, that DTs are ”digital replications of living as well as nonliving entities that enable data to be seamlessly transmitted between the physical and virtual worlds” [18].

To approach the problem form the other side, it might help to look at, what is not a digital twin. A virtual representation of a physical object without any exchange of data is a digital model [59]. If data is fed from the physical system into the model, a digital shadow is created [29]. Only a “bi-directional relation between a physical artefact and the set of its virtual models” [43] fulfils Schleich’s vision of a DT.

As an example, a computer aided design (CAD) model is a representation of a physical entity and it is typically used to describe the shape, dimensions and materials of a construction. This model can be a 2D or a 3D model. Only if the latest sensor data associated with a matching physical device is fed into this CAD model, it can be considered a DS. [59] By simulating different scenarios in a model and representing the current state of the system the DS turns into a digital twin, if decisions are made automatically fed back through an actuator into the physical entity [43]. An implementation of this data transmission can be found in ”Sensor Data Transmission from a Physical Twin to a Digital Twin” [2].

As the DT is mostly used in the context of production lines, it makes sense to look for similar examples in other industries, that have undergone comparable transformations with the beginning of the digital age. Such a comparable system can be found in the transport industry. Production lines have several points in common with train networks. One of the latest internationally standardized systems in rail infrastructure is European Train Control System (ETCS) which will probably become the standard signalling and control component in all European countries. A great overview on the ETCS-standard is given by Thales on their website [20].

K. TP Program

TP language Fanuc robots are usually programmed in a language called TP. For more information, the website of Onerobotics [49] gives an extensive collection of tutorials, manuals and guides. The book "Robot whispering" [50] available on their website is a fast way to get started with programming Fanuc Robots.

Pick and place TP program for Fanuc 210F Below, the program developed for the pick and place operation together with Qing can be found.

```
/PROG  FANUC2000_NoNUToffset_ToolframNumber
/ATTR
OWNER = MNEDITOR;
COMMENT = "RSL generated Program";
PROG_SIZE = 64000;
CREATE = DATE 19 09 12  TIME 16:02:18;
MODIFIED = DATE 19 09 12  TIME 16:02:18;
FILE_NAME = FANUC 2000;
VERSION = 0;
LINE_COUNT = 41;
MEMORY_SIZE = 64000;
PROTECT = READ_WRITE;
TCD: STACK_SIZE = 0,
TASK_PRIORITY = 50,
TIME_SLICE = 0,
BUSY_LAMP_OFF = 0,
ABORT_REQUEST = 0,
PAUSE_REQUEST = 0;
DEFAULT_GROUP = 1,*,*,*,*;
CONTROL_CODE = 00000000 00000000;
/MN
1:  LBL[1] ;
2:J P[1: P1]  80% FINE    ;
3:  WAIT DI[100: ]= OFF ;
4:  DO[106: ]= ON ;
5:J P[2: P14]  80% FINE    ;
6:  DO[1: ]= ON ;
7:  DO[102: ]= ON ;
8:  DO[104: ]= ON ;
9:  WAIT   0.50(sec) ;
10:J P[3: P3]  80% FINE    ;
11:J P[4: P12]  80% FINE    ;
12:J P[5: P2]  80% FINE    ;
13:L P[6: P13]  1500mm/sec FINE    ;
14:L P[7: P10]  1500mm/sec FINE    ;
15:  DO[1: ]= OFF ;
16:  DO[102: ]= OFF ;
17:  DO[104: ]= OFF ;
18:  DO[103: ]= ON ;
19:  DO[105: ]= ON ;
20:  WAIT   0.50(sec) ;
```

```
21:L P[8: P4]   1500mm/sec FINE     ;
22:J P[9: P15]   80% FINE     ;
23:  DO[101: ]= ON ;
24:  WAIT   50.00(sec) ;
25:L P[10: P18]   1500mm/sec FINE     ;
26:  DO[2: ]= ON ;
27:  DO[107: ]= OFF ;
28:  DO[106: ]= ON ;
29:  WAIT    0.50(sec) ;
30:L P[11: P7]   1500mm/sec FINE     ;
31:L P[12: P5]   1500mm/sec FINE     ;
32:J P[13: P8]   80% FINE     ;
33:J P[14: P11]   80% FINE     ;
34:  DO[2: ]= OFF ;
35:  DO[106: ]= OFF ;
36:  DO[107: ]= ON ;
37:  WAIT    0.50(sec) ;
38:J P[15: P9]   80% FINE     ;
39:J P[16: P16]   80% FINE     ;
40:L P[17: P17]   1500mm/sec FINE     ;
41: JMP LBL[1] ;
/POS
P[1]{
GP1:
UF : 1, UT : 1,              CONFIG : 'N U T, 0, 0, 0',
X =    900.00  mm,      Y =  1600.00  mm,      Z =     26.00  mm,
W =    180.00 deg,      P =     0.00 deg,      R =    180.00 deg
};
P[2]{
GP1:
UF : 1, UT : 1,              CONFIG : 'N U T, 0, 0, 0',
X =    900.00  mm,      Y =  1600.00  mm,      Z =     50.00  mm,
W =    180.00 deg,      P =     0.00 deg,      R =    180.00 deg
};
P[3]{
GP1:
UF : 1, UT : 1,              CONFIG : 'N U T, 0, 0, 0',
X =    900.00  mm,      Y =  1600.00  mm,      Z =     26.00  mm,
W =    180.00 deg,      P =     0.00 deg,      R =    180.00 deg
};
P[4]{
GP1:
UF : 1, UT : 1,              CONFIG : 'N U T, 0, 0, 0',
X =    900.00  mm,      Y =  1000.00  mm,      Z =     26.00  mm,
W =    180.00 deg,      P =     0.00 deg,      R =    180.00 deg
};
P[5]{
GP1:
UF : 1, UT : 1,              CONFIG : 'N U T, 0, 0, 0',
X =   1850.00  mm,      Y =     0.00  mm,      Z =    140.00  mm,
W =    180.00 deg,      P =     0.00 deg,      R =     90.00 deg
};
P[6]{
GP1:
UF : 1, UT : 1,              CONFIG : 'N U T, 0, 0, 0',
X =   2250.00  mm,      Y =     0.00  mm,      Z =    123.00  mm,
W =    180.00 deg,      P =     0.00 deg,      R =     90.00 deg
};
P[7]{
GP1:
UF : 1, UT : 1,              CONFIG : 'N U T, 0, 0, 0',
```

```
X =  2250.00  mm,     Y =     0.00  mm,     Z =    84.00  mm,
W =   180.00 deg,     P =     0.00 deg,     R =    90.00 deg
};
P[8]{
GP1:
UF : 1, UT : 1,              CONFIG : 'N U T, 0, 0, 0',
X =  2000.00  mm,     Y =     0.00  mm,     Z =   130.00  mm,
W =   180.00 deg,     P =     0.00 deg,     R =    90.00 deg
};
P[9]{
GP1:
UF : 1, UT : 1,              CONFIG : 'N U T, 0, 0, 0',
X =  1650.00  mm,     Y =     0.00  mm,     Z =   130.00  mm,
W =   180.00 deg,     P =     0.00 deg,     R =    90.00 deg
};
P[10]{
GP1:
UF : 1, UT : 1,              CONFIG : 'N U T, 0, 0, 0',
X =  2250.00  mm,     Y =     0.00  mm,     Z =   130.00  mm,
W =   180.00 deg,     P =     0.00 deg,     R =    90.00 deg
};
P[11]{
GP1:
UF : 1, UT : 1,              CONFIG : 'N U T, 0, 0, 0',
X =  2250.00  mm,     Y =     0.00  mm,     Z =   155.00  mm,
W =   180.00 deg,     P =     0.00 deg,     R =    90.00 deg
};
P[12]{
GP1:
UF : 1, UT : 1,              CONFIG : 'N U T, 0, 0, 0',
X =  1250.00  mm,     Y =     0.00  mm,     Z =   180.00  mm,
W =   180.00 deg,     P =     0.00 deg,     R =    90.00 deg
};
P[13]{
GP1:
UF : 1, UT : 1,              CONFIG : 'N U T, 0, 0, 0',
X =   815.00  mm,     Y =  1695.00  mm,     Z =   170.00  mm,
W =   180.00 deg,     P =     0.00 deg,     R =    90.00 deg
};
P[14]{
GP1:
UF : 1, UT : 1,              CONFIG : 'N U T, 0, 0, 0',
X =   815.00  mm,     Y =  1695.00  mm,     Z =   296.00  mm,
W =   180.00 deg,     P =     0.00 deg,     R =    90.00 deg
};
P[15]{
GP1:
UF : 1, UT : 1,              CONFIG : 'N U T, 0, 0, 0',
X =   815.00  mm,     Y =  1695.00  mm,     Z =   170.00  mm,
W =   180.00 deg,     P =     0.00 deg,     R =    90.00 deg
};
P[16]{
GP1:
UF : 1, UT : 1,              CONFIG : 'N U T, 0, 0, 0',
X =   885.00  mm,     Y =   500.00  mm,     Z =    26.00  mm,
W =   180.00 deg,     P =     0.00 deg,     R =   180.00 deg
};
P[17]{
GP1:
UF : 1, UT : 1,              CONFIG : 'N U T, 0, 0, 0',
X =   900.00  mm,     Y =  1600.00  mm,     Z =    26.00  mm,
```

```
164             W =    180.00 deg,      P =      0.00 deg,      R =    180.00 deg
165             };
166             /END
```

L. Robot Quick start guide

A big part of this was derived from [54].

Parts of the robot A quick overview over the visible parts.

FANUC R-2000iC/210F 6 axis robotic arm The robot has 6 movable joints with a possible payload attached to the end. Its features include a wide reach (2655 mm), sturdy but flexible arm design, a spring loaded counterbalance, relatively high payload capacity (210Kg) and fast moving axes. The joints also have hashes to indicate the zero positions which help with the calibration of the robot.

R30iA Robot Controller The Robot is controlled using an original equipment manufacturer controller called the FANUC R30iA controller. Its features include faster sustained speed and superior position accuracies. It also has the I/O ports that are used to connect grippers and other payloads. (It also houses the camera circuits which are required to access the data from the SONY camera provided with the robot. - Delta only) The controller is also provided with a data card slot in which the special SD cards manufactured can be inserted and used as external memory. On the outside of the controller (side of iPendant at Delta) a USB port can be found. With these, programs, firmware files and other files can easily be transferred. Additionally, there is extended connectivity via its Ethernet port e.g. for FTP available.

iPendant The teaching pendant is the primary user interface to the robot. It is used to move the joints of the robot manually, to program specific trajectories, to control the gripper, and various other actions. It also is an interface that can be used for input and output of the robot controller parameters. The user can access the system variables and position variables. (It is provided with a USB port that can be used to connect to a USB drive for external storage. - Delta only) It can also be used to setup an FTP server and client in order to communicate with the PC.

Gripper The robot as handed over does not have an active gripper system. It has been tested though with a pneumatic gripper controlled via the DO ports and pneumatic valves. A 2-way pneumatic valve, some piping and a pressure regulator are still available. (Delta Equipment varies here a lot).

iPendant Navigation Manual: The teaching pendant is the primary user interface to the robot. This section deals with the important buttons on the TP.

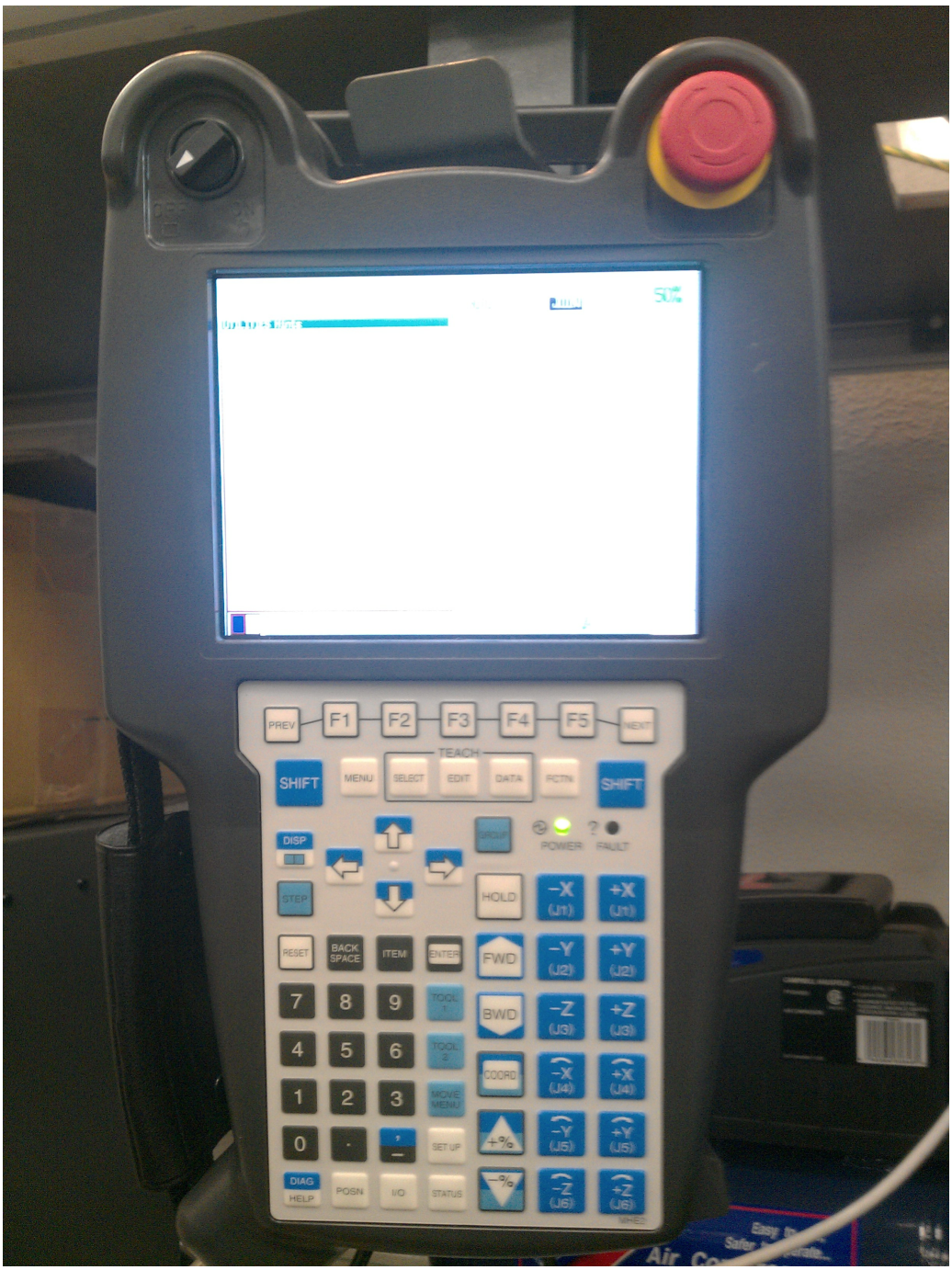

Figure L.1: iPendant

Emergency stop Makes the robot stop immediately by applying brakes. Use it only when necessary as the brakes wear down. There is also an E-stop on the controller. Press down on the button to activate it. Twist it to the right to release.

Hold If a slow and gradual halt is required, press HOLD on the iPendant.

TP on/off Below the E-stop button. It should be ON to access any function in the Teach Pendant (Set-ups, calibration, programs etc) and OFF when running in AUTO mode.

Deadman switch The 2 yellow bars behind the iPendant. 3 modes are available: Fully released, halfway pressed, fully pressed. Only the middle mode activates control.

Axis rotation There are 12 buttons on the right side to rotate each axis in either direction. They can only be used, after movement authority is given. Movement autority can be achieved by putting the controller in

manual mode with the key switch, setting the TP switch to on, and pressing Shift+Reset+ deadman. Shift+ deadman need to be pressed continuously to move the robot.

FWD and BWD Program execution can be guided step by step with these buttons.

+/- % With these two keys, the maximum movement speed can be varied.