



# **Simulation Model for a Six Axis Articulated Arm Industrial Ro- bot**

For Educational Use in Control System Design

Olivier Preuss

BACHELOR'S THESIS  
APRIL 2019

Double Degree Program in Mechanical Engineering

## ABSTRACT

Tampereen Ammattikorkeakoulu  
Tampere University of Applied Sciences  
Double Degree Program in Mechanical Engineering  
First Supervisor: Senior Lecturer Sami Hämäläinen, M.Sc. (Eng.)  
Second Supervisor: Principal Lecturer Markus Aho, D.Sc. (Tech.)  
&  
Hochschule Hannover  
Hannover University of Applied Sciences and Arts  
Double Degree Program in Mechanical Engineering  
First Supervisor: Professor Franz Christoph Kallage, Dr.-Ing.  
Second Supervisor: Professor Jens Hofschulte, Dr.-Ing.

Olivier Preuss  
Simulation Model for a Six Axis Articulated Arm Industrial Robot  
For Educational Use in Control System Design

Bachelor's Thesis 138 Pages, Appendices 66 Pages  
April 2019

---

Virtual system mappings and simulations are fundamental and contemporary engineering methods, especially within the widely spread field of complex, multi-domain mechatronic robotic systems. Therefore, companies as well as educational institutions typically try to apply the most recent simulation methods and software in order to achieve future-oriented and successful teaching, studies and researches.

The client of the thesis work at hand, Tampereen Ammattikorkeakoulu (TAMK) owns an ABB industrial robot manipulator that operates, among others, as a MIG/ MAG welding robot. Related to that, this thesis work aimed at the development and implementation of a MATLAB® Simulink® simulation model of an ABB IRB 2600-12/1.85 six axis articulated arm industrial robot for the purpose of educational use in control system design. The main objectives were: Design and implementation of a modular, maintainable and extendable simulation model based on a MATLAB® Simulink® Simscape™ Multibody™ model, derived from the robot's CAD model. The simulation model shall be used as a ready-to-use environment for control system structures design and include sufficient parameterization as well as user interface(s), motion planning and an operating manual.

In the context of the accomplishment, firstly the entirety of requirements was identified. In order to clearly outline the work extent and to meet the requirements satisfyingly, necessary definitions and regulations were formulated; this also covered general simplifications and restrictions. Task related, corresponding, common and state-of-the-art theory was studied and gathered from appropriate sources and adapted if required. The conceptual design was related to the preliminary determination of general matters e.g. the project structure and simulation flow but also of particular tasks e.g. the design of joint actuation models and graphical user interfaces. Finally, the conceptual design was implemented under the continuous consideration of the projects requirements, previously determined definitions and regulations and the corresponding theory.

The result of this thesis work is considered as a comprehensive and fully functional simulation program/ model that meets the client's requirements, covers optional accomplished tasks and can be used for the educational purposes it was initially meant for.

Nevertheless, the simulation model at its state at the finalization of the thesis work at hand, suffers weaknesses, incompleteness and limited capabilities due to incomplete parameterization, not conducted model validation and necessarily applied simplifications. Therefore, future continuations of this thesis work, e.g. in the context of further thesis works or semester projects, need to be applied to obtain a fully comprehensive and accurate simulation model.

---

Key words: robot, simulation, modeling, ABB IRB 2600, control systems, MATLAB® Simulink®

# CONTENTS

1	INTRODUCTION .....	1
2	TASK DEFINITION .....	3
3	DEFINITIONS AND REGULATIONS .....	5
3.1	Units.....	5
3.2	Gravitational Acceleration .....	5
3.3	Manipulator Axes .....	5
3.4	Coordinate Systems (Frames) .....	6
3.4.1	Base Coordinate System.....	7
3.4.2	World Coordinate System.....	7
3.4.3	Reference Coordinate System.....	8
3.4.4	Tool/ End Effector Coordinate System .....	8
3.5	Home Position.....	8
3.6	Tool/ End Effector Orientation .....	8
3.7	General Simplifications and Restrictions of the Simulation Model .....	9
3.7.1	Air Resistances.....	9
3.7.2	Rigidities .....	9
3.7.3	Frictions .....	10
3.7.4	Bearings .....	10
3.7.5	Backlashes and Uncertainties .....	10
3.7.6	Time Delays (Dead-Times).....	11
3.7.7	External Loads.....	11
3.7.8	Electric Motors.....	11
3.7.9	Transmission Gears .....	12
3.7.10	Other Electric Components and Computers.....	12
3.7.11	Thermal Effects .....	12
3.7.12	Environment.....	12
4	THEORY .....	13
4.1	Industrial Robot Manipulators .....	13
4.1.1	Articulated Arm Manipulators .....	14
4.1.2	Direct Kinematics.....	16
4.1.3	Inverse Kinematics .....	20
4.1.4	Dynamics.....	23
4.2	Motion Planning .....	26
4.2.1	Linear Trajectory Planning.....	28
4.2.2	Joint Trajectory Planning .....	33
4.3	Control Systems.....	37
4.4	MATLAB Simulink .....	41
4.5	Programming.....	42

**CONTENTS (CONTINUATION)**

5	CONCEPTUAL DESIGN.....	44
5.1	General Simulation Program Structure .....	45
5.2	Simulink/ Simulink Simscape .....	46
5.2.1	Simscape Multibody Model.....	46
5.2.2	Simulink.....	51
5.2.3	Parameter Provision .....	53
5.3	MATLAB.....	53
5.3.1	Simulation Model Variables .....	54
5.3.2	Graphical User Interfaces.....	56
5.3.3	Programs & Program Flow Charts.....	58
6	ACCOMPLISHMENT .....	64
6.1	CAD Model.....	64
6.2	Simulink Simulation Model .....	67
6.2.1	Simulink Simscape Multibody Robot Model.....	67
6.2.2	Signal Bus .....	73
6.2.3	Control System Structures.....	75
6.2.4	Measurements.....	77
6.3	MATLAB Program(s).....	81
6.3.1	Program & Program Structure(s).....	82
6.3.2	Simulation Model Variables .....	85
6.3.3	Motion Planning.....	91
6.3.4	Graphical User Interfaces.....	94
6.4	Simulation Model Parameters .....	98
6.4.1	Parameter Acquisition .....	99
6.4.2	Parameter Spreadsheet .....	106
6.4.3	Simulation Model Parameterization .....	109
6.5	Data Set File Structure.....	109
6.6	Operating Manual.....	110
6.7	Optional Tasks .....	111
6.7.1	Simplified Joint Actuation Motor Model(s) .....	111
6.7.2	Virtual Identification Measurements.....	113
7	TESTING AND DEBUGGING.....	117
8	OPERATION OF THE SIMULATION MODEL .....	119
9	CONCLUSION .....	120
10	OUTLOOK .....	123
	REFERENCES .....	125
	DECLARATION OF AUTHORSHIP .....	128
	APPENDICES.....	129

## ABBREVIATIONS AND TERMS

3D	Three-Dimensional
ABB	Asea Brown Boveri
AC	Alternating Current
ASM	Asynchronous Machine (Motor)
CAD	Computer Aided Design
CoM	Center of Mass
CP	Continuous Path
DC	Direct Current
DH	Denavit-Hartenberg
DOF	Degree of Freedom
FF	Feedforward
GUI	Graphical User Interface
GUIDE	Graphical User Interface Development Environment
HsH	Hannover University of Applied Sciences and Arts
IRB	Industrial Robot
MAG	Metal Active Gas Welding
MATLAB <sup>®1</sup>	Matrix Laboratory
MIG	Metal Inert Gas Welding
MIMO	Multiple-Input Multiple-Output
MISO	Multiple-Input Single-Output
Mol	Moment of Inertia
PFC	Program Flow Chart
Pol	Product of Inertia
PTP	Point-to-Point
SI	Système International (d'unités)
SIMO	Single-Input Multiple-Output
SISO	Single-Input Single-Output
STL/ .stl	Stereolithography
TAMK	Tampere University of Applied Sciences
TCP	Tool Center Point
XML/ .xml	Extensible Markup Language

---

<sup>1</sup> MATLAB<sup>®</sup> is a registered trademark of The MathWorks, Inc.

## LIST OF FIGURES

FIGURE 1.1: ABB IRB 2600 industrial robot (ABB Asea Brown Boveri Ltd. 2019a).....	2
FIGURE 3.1: ABB IRB 2600 robot axes and rotational directions definitions.....	6
FIGURE 3.2: ABB IRB 2600 robot coordinate systems definitions .....	7
FIGURE 4.1: ABB IRB 2600 individual robot elements assignment.....	14
FIGURE 4.2: Exemplary depiction of a kinematic structure's coordinate frames (DH-formalism).....	15
FIGURE 4.3: Exemplary description of the relative tool position and orientation (Siciliano, Sciavicco, Villani & Oriolo 2009, 59, modified). .....	18
FIGURE 4.4: Exemplary depiction of homogenous frame transformations (Siciliano, Sciavicco, Villani & Oriolo 2009, 61).....	19
FIGURE 4.5: Example of a multiple solution problem of inverse kinematics (Siciliano, Sciavicco, Villani & Oriolo 2009, 93, modified) .....	21
FIGURE 4.6: Vector based linear workspace path formulation (Weber 2017, 86, modified).....	28
FIGURE 4.7: Trapezoidal velocity profile (left) and the corresponding acceleration profile (right) (Weber 2017, 75, modified) .....	30
FIGURE 4.8: Sinusoidal acceleration profile and corresponding velocity and position graphs (Weber 2017, 79, modified) .....	31
FIGURE 4.9: Flow chart: adaption of the applicable path velocity (Weber 2017, 77, modified).....	32
FIGURE 4.10: Exemplary depiction of a fully synchronized axis motion (velocity) .....	34
FIGURE 4.11: Schematic depiction of internal and external robot control (Weber 2017, 25, modified) .....	38
FIGURE 4.12: Decentralized SISO control system structure (Bajd, Mihelj, Lenarčič, Stanovnik & Munih 2010, 78, modified).....	39
FIGURE 4.13: Schematic depiction of a decentralized cascaded SIMO control system structure (Grote, Bender & Göhlich 2018, T112, modified).....	40
FIGURE 4.14: Schematic depiction of a decentralized cascaded MIMO control system structure with centralized feedforward control (Grote, Bender & Göhlich 2018, T112, modified).....	41
FIGURE 5.1: Automatically generated Simscape Multibody model block diagram.....	44

## LIST OF FIGURES (CONTINUATION)

FIGURE 5.2: General simulation program structure diagram .....	46
FIGURE 5.3: Freehand sketch of the concept of the simulation model's Simscape block diagram.....	47
FIGURE 5.4: Freehand sketch of the concept of a subsystem of the Simscape block diagram.....	47
FIGURE 5.5: Freehand sketch of the concept of a joint actuation subsystem ..	48
FIGURE 5.6: Screen capture of a Simulink Simscape Asynchronous Machine block .....	49
FIGURE 5.7: Screen capture of a Simulink Simscape Cycloidal Drive block....	49
FIGURE 5.8: Screen capture of the MATLAB Simulink Simscape "Asynchronous Machine Scalar Control" example block diagram.....	50
FIGURE 5.9: Block diagram of a common basic closed-loop control system structure.....	51
FIGURE 5.10: Screen capture of an exemplary block parameterization .....	53
FIGURE 5.11: General Simulation Program Flow Chart .....	54
FIGURE 5.12: Freehand sketch of the concept of the Main (G)UI window .....	57
FIGURE 5.13: Freehand sketch of the concept of the Motion (G)UI window ....	58
FIGURE 5.14: Flow chart diagram of the linear movement planning .....	59
FIGURE 5.15: Program flow chart of the inverse_kinematics() function .....	61
FIGURE 6.1: Procedure of the generation of the Simscape Multibody model import files.....	65
FIGURE 6.2: Screen capture of the SolidWorks assembly of the robot manipulator .....	66
FIGURE 6.3: Screen capture of the final simulation model's Simulink block diagram.....	67
FIGURE 6.4: Screen capture of the final simulation model's Simulink Simscape robot model.....	68
FIGURE 6.5: Screen capture of the Simscape simulation model's Robot Base Subsystem .....	69
FIGURE 6.6: Screen capture of the Simscape simulation model's Joint 1 Drive Subsystem (1/2).....	71
FIGURE 6.7: Screen capture of the Simscape simulation model's ASM1 Driver Subsystem .....	72

**LIST OF FIGURES (CONTINUATION)**

FIGURE 6.8: Screen capture of the Simscape simulation model's Robot End Effector Subsystem.....	72
FIGURE 6.9: Screen capture of the Simscape simulation model's Joint 1 Drive Subsystem (2/2).....	73
FIGURE 6.10: Screen capture of the signal bus structure of the Control System subsystem InBus.....	74
FIGURE 6.11: Screen capture of the Control System subsystem of the Simulink block diagram.....	75
FIGURE 6.12: Screen capture of the Controller Joint 1 (Axis 1) (sub) subsystem .....	77
FIGURE 6.13: Screen capture of the Measurements subsystem of the Simulink block diagram.....	78
FIGURE 6.14: Screen capture of the Joint Angles (sub) subsystem of the Measurements subsystem .....	79
FIGURE 6.15: Screen capture of a Joint Angle Axis 1 Scope block signal plot	79
FIGURE 6.16: Screen capture of the Joint Velocities (sub) subsystem of the Measurements subsystem .....	80
FIGURE 6.17: Screen capture of the robot's Simscape Multibody model simulation animation (Mechanics Explorer) .....	81
FIGURE 6.18: Flow chart of the MATLAB General Program Flow.....	82
FIGURE 6.19: MATLAB program(s) structure and function/ file dependencies.	83
FIGURE 6.20: Screen capture of the first level of the structure of the robotPara variable .....	86
FIGURE 6.21: Screen capture of the first level of the structure of the simVar variable .....	87
FIGURE 6.22: Screen capture of the MATLAB Command Window and Workspace after the successful MATLAB program execution .....	87
FIGURE 6.23: Depiction of the linear movement's end effector position and pose.....	92
FIGURE 6.24: Screen capture of the simVar.linPathPlan and simVar.linPathPlan.pRes variables (example).....	93
FIGURE 6.25: Screen capture of the simVar.qSetValues and simVar.qSetValues.q1SV variables (example).....	94
FIGURE 6.26: Screen capture of the MATLAB main GUI main_ui .....	96



**LIST OF FIGURES (CONTINUATION)**

FIGURE 6.27: Screen capture of the MATLAB joint movement GUI joint_move_ui.....	97
FIGURE 6.28: Screen capture of the MATLAB linear movement GUI lin_path_ui .....	97
FIGURE 6.29: Screen capture of an invalid input MATLAB error message box	98
FIGURE 6.30: Cycloidal reduction gear of the RV-N series of the Nabtesco Corporation (Nabtesco Corporation 2019a) .....	100
FIGURE 6.31: Manufacturer’s rating table of the RV-N series cycloidal reduction gear (Nabtesco Corporation 2015, 8, modified) .....	101
FIGURE 6.32: Manufacturer’s rating table of the RV-N series cycloidal reduction gear (continuation) (Nabtesco Corporation 2015, 9, modified) .....	101
FIGURE 6.33: Manufacturer’s efficiency table of the RV42-N cycloidal reduction gear (Nabtesco Corporation 2015, 36).....	101
FIGURE 6.34: ABB IRB 2600 Axis 4, 5 & 6 AC motor (ABB Asea Brown Boveri Ltd. 2019d).....	102
FIGURE 6.35: TBL-I IV series compact size AC servomotor basic specifications (TAMAGAWA SEIKI Co., Ltd. 2019, 2, modified).....	102
FIGURE 6.36: TSM3204 400W AC200V torque characteristic diagram (TAMAGAWA SEIKI Co., Ltd. 2019, 8) .....	103
FIGURE 6.37: Screen capture of link 2 of the robot manipulator’s CAD model .....	105
FIGURE 6.38: Screen capture of the (8) Joint Parameters worksheet of the parameters spreadsheet .....	107
FIGURE 6.39: Screen capture of the ASM1 block parameterization.....	109
FIGURE 6.40: General file structure of the data set of the simulation model ..	110
FIGURE 6.41: Simulink screen capture of the Robot Link 3 Subsystem with a DC motor model.....	112
FIGURE 6.42: Simulink screen capture of the DC Motor 4 Driver subsystem	113
FIGURE 6.43: Exemplary plot of a trapezoidal joint velocity profile of the robot’s first axis.....	115

**LIST OF TABLES**

TABLE 3.1: Assignments and limitations of the manipulator's axes (ABB Asea Brown Boveri Ltd. 2019b, 11).....	6
TABLE 4.1: Assignment and typecast of the ABB IRB 2600 robot elements ....	14
TABLE 5.1: Listing and description of the conceptualized MATLAB function(s) (files).....	62
TABLE 6.1: List of SolidWorks parts and assemblies of the manipulator.....	65
TABLE 6.2: List of Simscape Multibody simulation model input files of the manipulator .....	66
TABLE 6.3: Listing of the available signal bus signal types .....	74
TABLE 6.4: Listing and description of additionally implemented MATLAB function(s) (files) .....	84
TABLE 6.5: Descriptions of the five simulation variables .....	88
TABLE 6.6: Detailed description of the robotPara variable .....	89
TABLE 6.7: Detailed description of the simVar variable.....	90
TABLE 6.8: ABB IRB 2600 gearbox spare part information (axis 3) .....	100
TABLE 6.9: ABB IRB 2600 motor spare part information (axis 4, 5 & 6).....	100
TABLE 6.10: Robot manipulator's link mass and volume information.....	106
TABLE 6.11: Structure and contents of the parameters spreadsheet.....	108

## 1 INTRODUCTION

From a general perspective, the origin of this thesis work bases on the demand of providing contemporary teaching methods to pupils and students. In this context, simulation is an important share of the teaching content, especially within the widely spread engineering domain.

Due to the continuous and rapid development of powerful simulation software during the last decades, nowadays complex systems can be virtually mapped and simulated with moderate effort. In parallel to that, the capabilities and availabilities of computer systems in general, but also common PCs, increased massively while the costs decreased. Furthermore, software developers often provide discounted or free (academic) software versions for general educational purposes to the institutions or to the students/ pupils directly.

In sum, this allows educational institutions to include the most recent simulation methods and software in teaching, studies and researches with moderate administrative and financial efforts.

From a more particular perspective, the topic of this thesis work was initiated by lecturers of the engineering department of the Tampereen Ammattikorkeakoulu<sup>1</sup> (TAMK). The TAMK owns an ABB<sup>2</sup> IRB 2600-12/1.85 industrial robot manipulator that operates, among others, as a MIG/ MAG welding robot equipped with a Fronius<sup>3</sup> Robacta Drive CMT welding solution (which also includes the torch end effector). Unified with other ABB components and additional applications and equipment, the entirety represents a multifunctional robot cell located at the TAMK production engineering laboratory (room F0-19). The industrial manipulator was and is still used for teaching, laboratory-, project- and thesis works but also in the context of the accomplishment of external commercial customer orders.

The purpose of the thesis work is the development of a MATLAB<sup>®</sup> Simulink<sup>®4</sup> simulation model of an ABB IRB 2600 six axis articulated arm industrial robot for educational use in control system design.

---

<sup>1</sup> English: Tampere University of Applied Sciences

<sup>2</sup> ABB Asea Brown Boveri Ltd.

<sup>3</sup> Fronius International GmbH

<sup>4</sup> MATLAB<sup>®</sup> Simulink<sup>®</sup> is a registered trademark of The MathWorks, Inc.

The simulation model shall be used as a ready-to-use environment for control system structures designed and implemented by pupils/ students. Thus, the simulation model shall include all components to execute, monitor and record kinematic and dynamic simulations of the robot, except the controller structures themselves. Additionally, the simulation model shall be easily varied, e.g. change of motor types, change of computer aided design (CAD) data and/ or extended, e.g. with pneumatics, hydraulics or any other applicable elements within the MATLAB Simulink (this also covers Simulink<sup>®</sup> Simscape<sup>™</sup> and Simscape<sup>™</sup> Multibody<sup>™</sup>)<sup>5</sup> environment. Furthermore, MATLAB Simulink simulation results shall be comparable to results gained from hand calculations, other simulation types or real measurements.



FIGURE 1.1: ABB IRB 2600 industrial robot (ABB Asea Brown Boveri Ltd. 2019a)

In the context of recent thesis works at the TAMK, the industrial manipulator was thematised in: Älykäs Huuva (Hyypä 2015), Design of an Intelligent Protection Shield (Rodewald 2016), Designing and implementing a Robot Gripper using additive manufacturing (Gerland 2017), Creation of an Augmented Reality App for an Introduction to Industrial Machine Mechanics (Compton 2018).

---

<sup>5</sup> Simulink<sup>®</sup> Simscape<sup>™</sup> and Simscape<sup>™</sup> Multibody<sup>™</sup> are trademarks or registered trademarks of The MathWorks, Inc.

## 2 TASK DEFINITION

In addition to the description of the purpose of this thesis, written in the second last paragraph of the section 1, a conclusion of the thesis objectives is listed subsequently. The objectives were defined by the customer (TAMK) and do contain required as well as optional elements.

Important note: The administrative part of the accomplishment of the thesis work also included a conclusion of a contract between the receiving institution (client; TAMK) and the author. This thesis contract also contains a complete listing of the thesis objectives and can be found from the appendices as Appendix 1. Thesis Contract. The corresponding project plan is available from Appendix 2. Project Plan.

The thesis work bases on the industrial robot manipulator of the type: ABB IRB 2600-12/1.85. Due to the topic of the thesis (simulation), processes and their outcomes are mainly related to software such as simulation software (MATLAB/ Simulink/ ABB RobotStudio) and CAD software. In this context, more details can be obtained from the thesis contract (Appendix 1. Thesis Contract) or the list of requirements (Appendix 3. List of Requirements).

The simulation model to be created shall base on the MATLAB programming language which also covers Simulink and Simscape block diagrams. Predefined MATLAB/ Simulink contents like toolboxes, classes, functions, blocks, etc. shall be preferred and used whenever available to accomplish a task.

As a minimum requirement, the interaction between the user and the simulation model shall be realised via the MATLAB command window. The creation of a (graphical) user interface(s) is optional, but if implemented, the design shall be kept simple and intuitive.

The simulation model shall include a graphical representation of the robot based on a CAD model with a sufficient precision implemented via Simulink Simscape Multibody. The simulation model shall represent the real robotic system with all its properties such as geometry and dimensions, physics, etc. as sufficient as an appropriate effort of the acquisition of the properties allows.

Identification measurements taken from the real robotic system are not part of the thesis. Required data shall be acquired from product documents, software sources, third parties, etc.

In case of not obtainable data, simplifications and assumptions are allowed but to be clearly revealed and founded in a sufficient way.

The simulation model shall use the BASE coordinate system (frame) as its main coordinate system. The BASE coordinate system shall be in coincidence with the WORLD coordinate system and represent a reference coordinate system which acts as reference for target definitions and end effector orientations. All coordinate systems are right-handed Cartesian coordinate systems.

The home configuration, axis designations, initial angular positions, angular limitations and directions of rotation (signs) of the simulation model shall be in accordance with the defaults of the real robotic system defined by the manufacturer.

The simulation model shall include two types of motion planning. Firstly, a linear path planning from coordinate "A" to coordinate "B", specified by the user in the reference coordinate system. Secondly, the values of the angles of every individual rotational (revolute) joint of the robot model shall be allowed as user input for the motion planning of a joint movement.

The structure of the simulation model shall be created in a modular way. Maintaining, editing, updating and extending the model shall be possible with moderate effort. The program flow and the operation of the model shall be designed clearly structured. Simulation results gained from MATLAB Simulink shall be comparable to measurements taken from the real robot system and/ or the ABB RobotStudio software, which is optional. Furthermore, a short concise instruction document for the operation and service of the simulation model shall be created.

The simulation model shall generally base on the SI base units and derived units, exceptions are allowed if meaningful and sufficiently justified.

### 3 DEFINITIONS AND REGULATIONS

In order to ensure a consistent content within this thesis work, the following definitions and regulations are valid for the complete thesis work. This also covers all the documents and data created in the context of the accomplishment of this work such as documentations, program codes, the simulation model, CAD data, etc.

#### 3.1 Units

Only SI-units and derived SI-units are used. Deviating from this, the units “percentage [%]” and “degree [°]” are used in the context of user inputs (user interfaces and robot parameters spreadsheet) and simulation result measurements (Simulink Simscape environment) in order to increase the comprehensibility of the provided/ measured values.

#### 3.2 Gravitational Acceleration

The value of the gravitational acceleration is in accordance with the MATLAB Simulink default settings and defined as:

$$g = 9.80665 \frac{\text{m}}{\text{s}^2} \quad (3.1)$$

The direction is defined as the negative direction of the z-axis ( $Z_0$ ) of the base frame (see FIGURE 3.2).

#### 3.3 Manipulator Axes

The definitions of the manipulator’s axes and the corresponding rotational directions of the revolute joints are in accordance with the manufacturer’s definitions and are shown below (FIGURE 3.1).

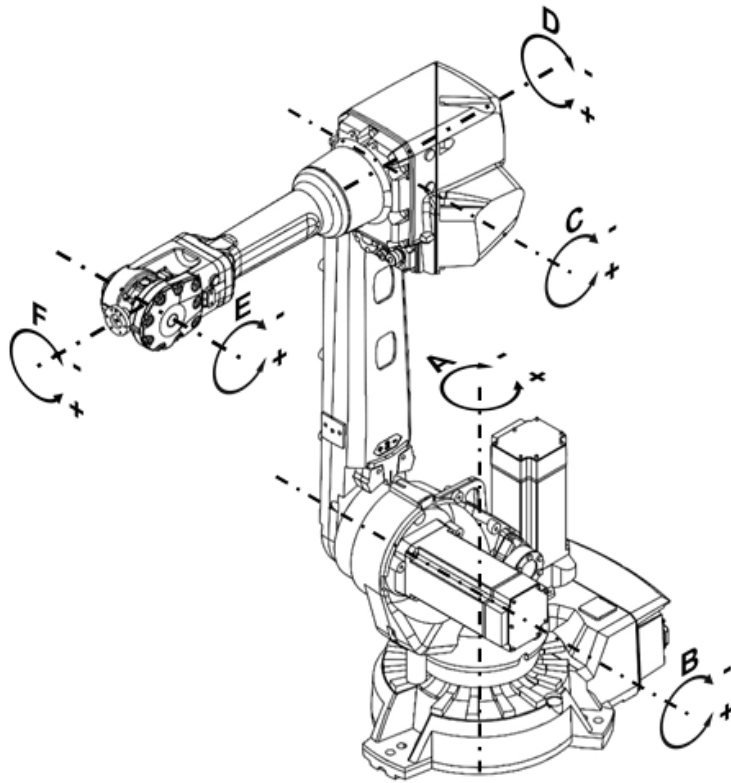


FIGURE 3.1: ABB IRB 2600 robot axes and rotational directions definitions

TABLE 3.1: Assignments and limitations of the manipulator's axes (ABB Asea Brown Boveri Ltd. 2019b, 11)

n:	Axis:	Name:	Symbol:	Upper Limit [°]:	Lower Limit [°]:
1	Axis 1	A	$q_1$	+180	-180
2	Axis 2	B	$q_2$	+155	-95
3	Axis 3	C	$q_3$	+75	-180
4	Axis 4	D	$q_4$	+400*	-400*
5	Axis 5	E	$q_5$	+120	-120
6	Axis 6	F	$q_6$	+400**	-400**
*: (+ 251 rev. to - 251 rev. Max.) **: (+ 274 rev. to - 274 rev. Max.)					

### 3.4 Coordinate Systems (Frames)

Coordinate systems (frames) are always right-handed Cartesian coordinate systems.



### 3.4.1 Base Coordinate System

The base coordinate system (frame) (index 0, see FIGURE 3.2) definition is in accordance with the common definition: The x-y-plane of the base frame is in coincidence with the set-up area of the base of the robot. The z-axis of the base frame is in coincidence with the robot's first revolute joint axis (Axis 1, A) and points away from the x-y-plane.

The described definition of the base frame is in accordance with the manufacturer's definition (ABB Asea Brown Boveri Ltd. 2019c, 24-28).

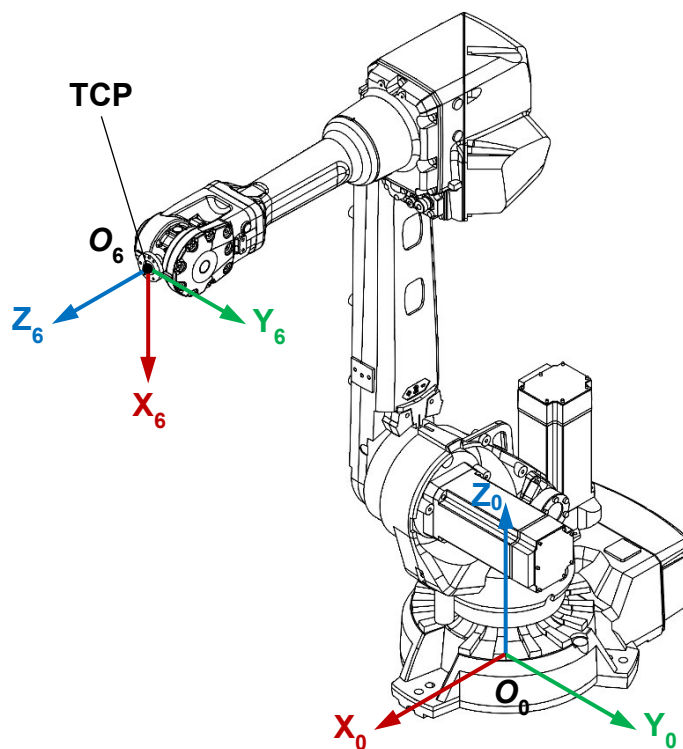


FIGURE 3.2: ABB IRB 2600 robot coordinate systems definitions

### 3.4.2 World Coordinate System

The world coordinate system (frame) is in coincidence with the base coordinate system.

### 3.4.3 Reference Coordinate System

The reference coordinate system (frame) is in coincidence with the base coordinate system.

### 3.4.4 Tool/ End Effector Coordinate System

The tool/ end effector coordinate system (frame) (index 6, see FIGURE 3.2) definition is in accordance with the common definition: The x-y-plane of the tool/ end effector frame is in coincidence with the tool mounting surface of the last (seventh) link of the manipulator. The z-axis of the tool/ end effector frame is in coincidence with the manipulator's last revolute joint axis (Axis 6, F) and points away from the x-y-plane. The origin of the tool/ end effector coordinate system  $O_6$  (see FIGURE 3.2) is called tool center point (TCP).

The described definition of the tool/ end effector frame is in accordance with the manufacturer's definition (ABB Asea Brown Boveri Ltd. 2019c, 24-28).

## 3.5 Home Position

The home position of the manipulator is shown FIGURE 3.1. In the home position, the manipulator's pose is defined by the angular values of the axes:

$$q_1 \dots q_6 = 0^\circ \quad (3.2)$$

This axes configuration also acts as the reference for any angular joint movements.

## 3.6 Tool/ End Effector Orientation

In the context of simulation model user inputs, the tool/ end effector orientation is described with the common definition of the ZYX-Euler angles  $[\alpha \beta \gamma]$ .

The base frame of the manipulator (see sub section 3.4.1) is the reference frame for the description of the tool/ end effector orientation. This topic is discussed more detailed in section 4.1.3.

### **3.7 General Simplifications and Restrictions of the Simulation Model**

Real robotic systems are highly dynamic and complex structures. The robotic system's static and dynamic behaviour is influenced by means of effects originated in their type, application, conditions, environment etc. Despite the fact that some of the effects cause non-negligible impacts on the system's behaviour, simplifications are necessarily made and restrictions applied in order to keep the thesis work within a manageable extent. Furthermore, the created simulation is meant for educational purposes with emphasis on control system design at undergraduate level. Therefore, the complexity of the model needs to cover main characteristics of the real system but also needs to be kept at a moderate level to ensure the traceability of its behaviour.

#### **3.7.1 Air Resistances**

Influences caused by forces evoked by the movement of the real robotic system in its ambient atmosphere (air) are neglected.

Justification: Air resistance influences were not mentioned by any source listed in the references in the context of industrial robots, therefore, they were considered as negligible. This is only valid for the robotic system itself. In case of the simulation of loads with large dimensions, e.g. sheet metals, combined with high velocity movements, non-negligible deviations can occur which are not considered by the simulation model.

#### **3.7.2 Rigidities**

All types of bodies of the simulation model such as links, joints, shafts, transmission gears, belts, etc. are considered as ideal rigid bodies.

Justification: The determination of the non-rigid properties of the real robotic system components can only be obtained from non-public manufacturer's data and/ or sophisticated measurements not covered by the scope of the thesis work. If obtained, elasticities can be taken into consideration by adding the corresponding blocks within the Simulink environment.

### **3.7.3 Frictions**

The simulation of frictions is limited to the number of Simulink Simscape blocks of main elements of the real robotic system modelled in the simulation model and their individual level of detail (e.g. viscous rotor damping). This covers only constant and linear frictional effects such as breakaway frictions, Coulomb frictions and (linear) viscous (damping) frictions. Non-linear frictional effects are not implemented but can be added to the Simulink block diagram(s) if required and if sufficient parameters are available.

### **3.7.4 Bearings**

Only joint bearings are simulated separately within the simulation model. The assumption is made that the effects related to the bearing of each individual component (e.g. motor) are sufficiently covered by the applicable parameters of the corresponding individual Simulink block.

### **3.7.5 Backlashes and Uncertainties**

All types of backlashes, e.g. originated from bearings and transmissions are neglected. All types of uncertainties of the real robotic system, especially geometrical uncertainties are neglected.

Justification: At the case at hand, both backlashes and uncertainties can only be obtained from non-public manufacturer's data and/ or sophisticated measurements not covered by the scope of the thesis work.

### **3.7.6 Time Delays (Dead-Times)**

Time delays occur on the real robotic system, originated from the specific behaviour of each physical component and their interactions within the complete system. This also covers time delays caused by the differences of continuous signals considered in the theory and non-continuous (= discrete) signals typically processed in real systems, especially in the context of control systems (Weber 2017, 177). Furthermore, the simulation bases on computation and is executed on PCs with non-real-time operating systems, thus discrete signals are used and time delays also depend on the recent workload of the PC.

Because sophisticated measurements are necessary to identify the time delays of the real robotic system, time delays are not considered in the simulation.

If obtained, time delays can be taken into consideration during control system design in the Simulink environment using appropriate Simulink blocks.

### **3.7.7 External Loads**

The simulation model represents the real robotic system only equipped with a welding torch end effector. The welding system is not connected to the real robotic manipulator, except the end effector and its supply wiring. In this context, the end effector supply wiring is not included in the simulation model because the modelling of its specific behaviour is considered as too complex.

If required, external loads, e.g. represented by rigid bodies, can be added to the CAD model within the CAD software or the simulation model within the Simulink Simscape environment.

### **3.7.8 Electric Motors**

The level of detail of the electric motor models (joint actuation) of the simulation model is limited to the level of details of the corresponding Simulink Simscape blocks (diagrams). This covers typical electrical (e.g. voltage, power, impedances, etc.) and mechanical (e.g. viscous rotor damping) parameters only.

### **3.7.9 Transmission Gears**

The level of detail of the transmission gears/ gearboxes is limited to the level of detail of the corresponding Simulink Simscape blocks (diagrams) e.g. non-uniform transmission behaviour. Transmission gears parameterization covers gear ratios, (rotational) inertias, input to output efficiencies and output to input efficiencies only. The values of (rotational) inertias of the transmission gears are always related to the input shaft (motor side).

### **3.7.10 Other Electric Components and Computers**

The level of detail of the electric components of the simulation model, e.g. motor driver circuits, is limited to the level of detail of the corresponding Simulink Simscape blocks (diagrams) e.g. parasitic capacities.

Specific features and properties of the manufacturer's computational units and power systems are not identified or implemented in any way due to the lack of sufficient acquirable sources.

### **3.7.11 Thermal Effects**

Thermal effects are generally not considered in the context of this thesis work (e.g. transmission lubricant viscosity) but can be considered by additional Simulink blocks or appropriate parameterization of existent blocks with specific prepared but deactivated functionalities (e.g. transmission gears/ gearbox blocks).

### **3.7.12 Environment**

Influences on the real robotic system caused by environmental effects such as any kinds of external forces, vibrations, energy supply fluctuations, electromagnetic disturbances, atmospheric changes, etc. cannot be considered due to a lack of acquirable information and/ or their unpredictable character.

## 4 THEORY

The subsequently described theory only covers the particular topics which were necessarily needed to accomplish the tasks related to this thesis work. If not defined divergently, the general overall usage of formulas, symbols, naming, indexing and units relates to the content described in this theory section.

### 4.1 Industrial Robot Manipulators

At the present state and in the context of modern technologies and societies, the term “robot” is often used loosely to describe a particular machine or application from a huge variety of subareas (e.g. industrial robots, service robots, etc.). Several national and international systematic definitions, classifications and standardizations exist in order to define and categorize each robot precisely. A short but concise definition of industrial robots:

A manipulating industrial robot is an automatically controlled, re-programmable, multipurpose manipulator programmable in three or more axes, which may be either fixed in place or mobile for use in industrial automation applications (Kelly, Santibáñez & Loría 2005, 4).

Due to the predetermination of a particular robot type (ABB IRB 2600) to be used in this thesis work, the theory is narrowed to stationary (fixed in place) industrial robot manipulators with (six axes) serial kinematics.

Real industrial robot manipulators as well as their theoretical abstractions do usually consist of several elements but can be broken down into two types of kinematic main elements, links (rigid bodies) and joints, as shown subsequently (FIGURE 4.1).

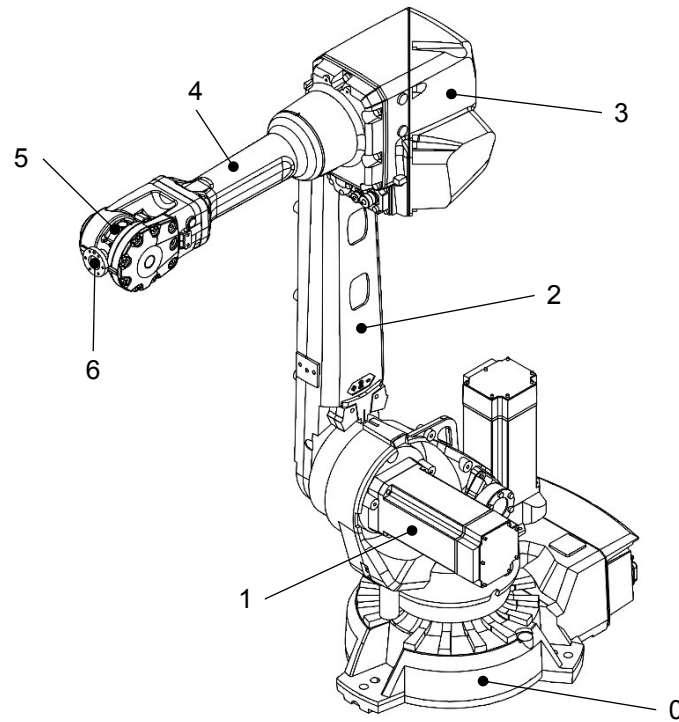


FIGURE 4.1: ABB IRB 2600 individual robot elements assignment

TABLE 4.1: Assignment and typecast of the ABB IRB 2600 robot elements

No.:	(Trivial-)Name:	Element Type:	Connection Link $i+1$ to Link $i$ via:
0	Base	(Rigid Body) Link	Revolute Joint ( $f = 1$ )
1	Shoulder		
2	Lower Arm		
3	Upper Arm		
4	Wrist		
5	Wrist		
6	Wrist (End Effector)		

#### 4.1.1 Articulated Arm Manipulators

From the perspective of mechanics, manipulators are commonly distinguished by their kinematic structure. The structures are generally divided into the fields of serial and parallel kinematics, whereby in the case at hand the serial kinematics were considered only.



Serial kinematic structures are characterized by an open kinematic chain, a chain of links connected by (kinematic) joints, typically revolute or prismatic joints. Links are considered as ideal rigid bodies with surfaces that are geometrically perfect in both position and shape. Each link has its own fixed frame.

A kinematic joint is a connection between two bodies that allows relative motion with a particular number of degrees-of-freedom  $f$  (DOF) and without any clearances. In the case at hand, the robotic manipulator only contains of a number of six single revolute (R) joints, whereby a revolute joint itself is a lower-pair-joint (surface contact) with one DOF. (Siciliano & Khatib 2008, 18-19)

$$f = 1 \quad (4.1)$$

Hence, each revolute joint allows only one direction of motion and is represented by one motion variable:

$$q_i \quad (4.2)$$

In general, the  $z$ -axis of the  $i-1$ -coordinate frame is in coincidence with the  $i$ -revolute axis of the joint, see FIGURE 4.2 below.

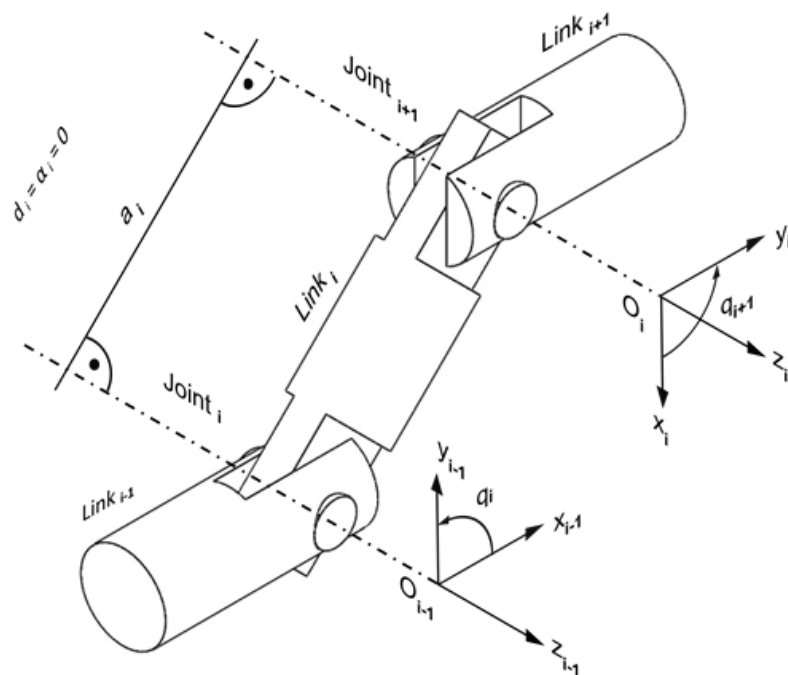


FIGURE 4.2: Exemplary depiction of a kinematic structure's coordinate frames (DH-formalism)

The motion variable  $q_i$  typically describes the angle between the fixed frames of the two links connected to the joint. The motion variables of the robot, six in this case, are collected conveniently in the column vector  $\mathbf{q}$ :

$$\mathbf{q} = \begin{bmatrix} q_1 \\ q_2 \\ \vdots \\ q_6 \end{bmatrix} \quad (4.3)$$

The robotic manipulator at hand contains six single revolute joints, whereby every single revolute joint increases the DOF of the robot by one. Thus, the six axis robotic manipulator has six DOF:

$$n = F = f = 6 \quad (4.4)$$

In theory, this allows the end effector (also tool), or more precisely, the TCP to reach every point within the workspace (neglecting the angular joint limitations of real robotic systems).

#### 4.1.2 Direct Kinematics

In general, the theory of kinematics describes the motion of a kinematic structure without the consideration of forces and/ or torques causing that motion (Siciliano & Khatib 2008, 9). In the general context of robotics and in the particular context of this document, the theory of kinematics was split up into direct (also forward) kinematics (section 4.1.2) and the inverse kinematics (section 4.1.3).

Note: The kinematic structure of the robotic manipulator (`RigidBodyTree`) was automatically generated during the procedure of the Simulink Simscape Multi-body CAD model import. Direct kinematics are automatically solved in the Simulink/ Simulink Simscape environment during the computation of the solution of the simulation model. Inverse kinematics, required for motion planning purposes (see section 4.2), are solved using the predefined MATLAB inverse kinematics solver (`GeneralizedInverseKinematics; gik()`).

Therefore, the subsequently described theories of direct and inverse kinematics were narrowed to the basic required extent.

Continuing from the descriptions of the previous section 4.1.1, the robotic manipulator's kinematic structure is described by an open chain, a series of rigid bodies (links) and joints, whereby the first (fixed) link (index 0) is typically named "base" and the last link (index 6 in this case) is typically named "end effector" or "tool".

The purpose of the theory of direct kinematics is the description of the end effector's/ tool's position and orientation, relative to a reference (fixed base frame, index 0), as a function of the joints motion variables (united in the  $6 \times 1$ -dimensional vector  $\mathbf{q}$ ), hence, in joint-space (Siciliano, Sciavicco, Villani & Oriolo 2009, 58).

In the case at hand, the six DOF of the robotic system are divided into the position of the tool frame ( $O_6$ ) (three DOF) and the orientation of the tool frame (another three DOF) with respect to the reference frame ( $O_0$ ). This is typically described by a  $4 \times 4$ -matrix:

$$\mathbf{T}(\mathbf{q})_6^0 = \begin{bmatrix} \mathbf{x}_6^0(\mathbf{q}) & \mathbf{y}_6^0(\mathbf{q}) & \mathbf{z}_6^0(\mathbf{q}) & \mathbf{p}_6^0(\mathbf{q}) \\ 0 & 0 & 0 & 1 \end{bmatrix} \quad (4.5)$$

Whereby  $\mathbf{p}_6^0$  is a  $3 \times 1$ -dimensional vector that points from the origin of the fixed frame ( $O_0$ ) to the origin of the tool frame ( $O_6$ ) (= Cartesian coordinates of the TCP) and  $\mathbf{x}_6^0$ ,  $\mathbf{y}_6^0$ ,  $\mathbf{z}_6^0$  are each  $3 \times 1$ -dimensional unit vectors that describe the orientation of the tool frame, both position and orientation with respect to the fixed reference frame (see FIGURE 4.3 below).

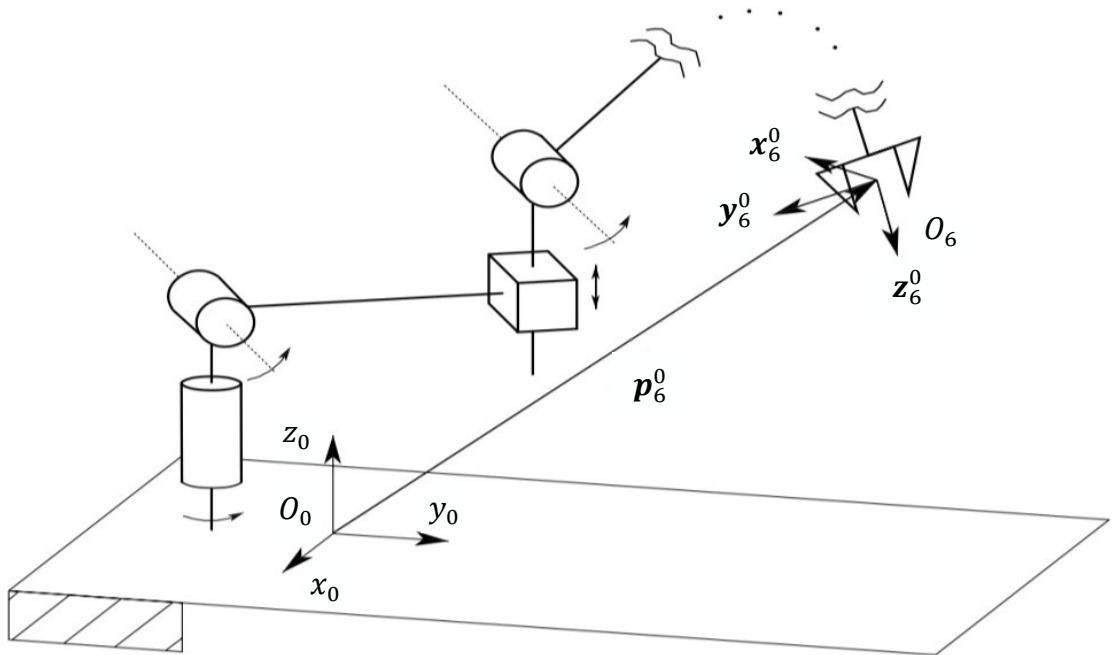


FIGURE 4.3: Exemplary description of the relative tool position and orientation (Siciliano, Sciacicco, Villani & Oriolo 2009, 59, modified).

To obtain the description of the tool position and orientation depending on the properties of the kinematic structure, so called homogeneous transformations, represented by homogenous transformation matrices:

$$\mathbf{A}_i^{i-1}(q_i) \quad (4.6)$$

Need to be applied. In the case at hand, a sequence of six homogenous transformations is required to obtain the transformation from the base frame to the tool frame:

$$\mathbf{T}(\mathbf{q})_6^0 = \mathbf{A}_1^0(q_1) \mathbf{A}_2^1(q_2) \mathbf{A}_3^2(q_3) \mathbf{A}_4^3(q_4) \mathbf{A}_5^4(q_5) \mathbf{A}_6^5(q_6) \quad (4.7)$$

This procedure is exemplarily depicted in the subsequent FIGURE 4.4.

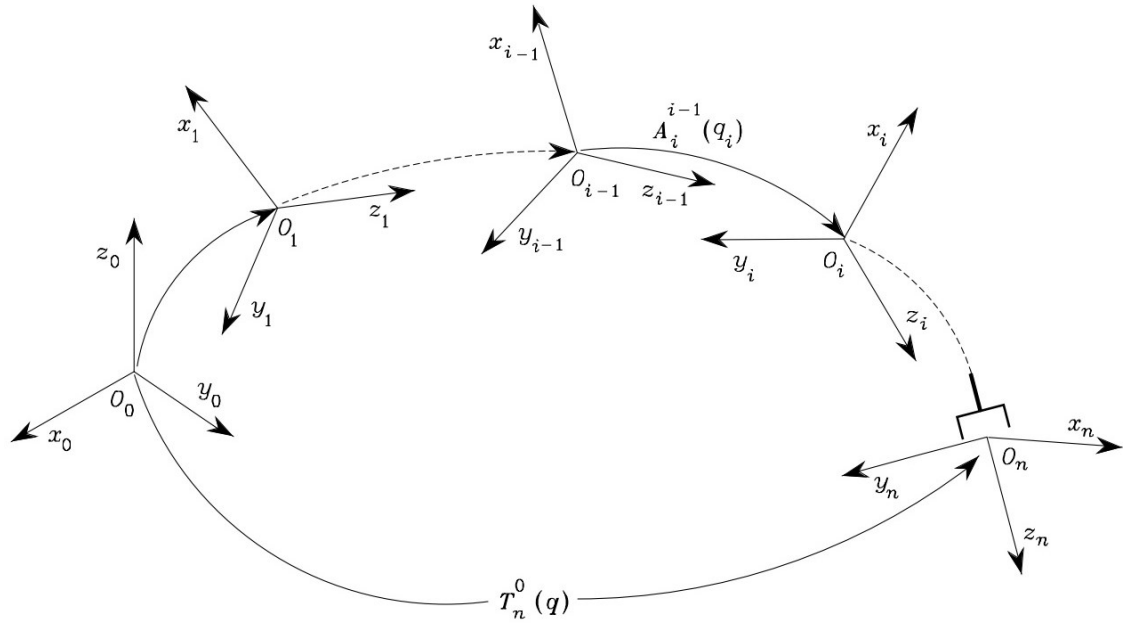


FIGURE 4.4: Exemplary depiction of homogenous frame transformations (Siciliano, Sciavicco, Villani & Oriolo 2009, 61).

The homogenous transformation matrices in turn are each based on a sequence of four fundamental matrix operations:

$$A_i^{i-1} = Rot(\vec{z}_{i-1}, \theta_i) \cdot Trans(\vec{z}_{i-1}, d_i) \cdot Trans(\vec{x}_{i-1}, a_i) \cdot Rot(\vec{x}_i, \alpha_i) \quad (4.8)$$

The described method as well as the parameters  $\theta_i$ ,  $d_i$ ,  $a_i$  and  $\alpha_i$  (the so called Denavit-Hartenberg (DH) parameters), are based on and determined during the application of the quasi-standard (but non-unique!) Denavit-Hartenberg formalism for the general determination of the frames of a kinematic structure:

- Axis  $z_i$  along the  $i + 1$ -joint axis.
- $O_i$  is located at the intersection of the  $z_i$  axis with the common normal to the  $z_{i-1}$  axis and the  $z_i$  axis.
- Axis  $x_i$  along the common normal to the axes  $z_{i-1}$  and  $z_i$  with positive direction from  $i$ -joint to  $i + 1$ -joint.
- Axis  $y_i$  is chosen in a way to obtain a right-handed frame.

(Siciliano, Sciavicco, Villani & Oriolo 2009, 62). (See also FIGURE 4.2)

Typically, the DH parameters are united in  $n \times 1$ -dimensional parameters vectors each:

$$\boldsymbol{\theta}, \boldsymbol{d}, \boldsymbol{a}, \boldsymbol{\alpha} \quad (4.9)$$

In the case at hand, the elements of the  $6 \times 1$ -dimensional parameter vectors  $\boldsymbol{d}$ ,  $\boldsymbol{a}$  and  $\boldsymbol{\alpha}$  are fixed values based on/ derived from the mechanical structure (geometry) of the robotic system. The elements of the  $6 \times 1$ -dimensional parameter vector  $\boldsymbol{\theta}$  are variable parameters (values) and defined by the elements (values) of the joint position/ motion variable  $\boldsymbol{q}$ .

### 4.1.3 Inverse Kinematics

The inverse kinematics theory aims at the determination of the required values of the joint position variable in order to describe a given position and orientation of the end effector (relative to the reference frame), hence, inverse kinematics is the inversion of the direct kinematics (Siciliano & Khatib 2008, 27). Furthermore, inverse kinematics are an important and fundamental part of robotic manipulators theory, especially in the context of motion planning, e.g. for the calculation of a linear reference trajectory for welding applications.

In contrary to the direct kinematics, where orientation and pose of the end effector are always same for the same set of joint position variable values (= unique), solving the inverse kinematics is a much more complex problem due to:

- The corresponding equations are typically non-linear; closed-form solutions do not necessarily exist.
- Multiple solutions may exist.
- An infinite number of solutions may exist.
- Possibly no admissible solutions are available due to the specific manipulator's kinematic structure.

(Siciliano, Sciavicco, Villani & Oriolo 2009, 90-91).

Modern commonly used applications like e.g. MATLAB use numerical methods for the solving the inverse kinematics problem. As closed-form solutions are typically related to the analytical solving method, numerical solving does not suffer this problem. Furthermore, and in contrary to the analytical method, numerical methods are independent from the specific robot manipulators type. In turn, numerical solving can be less performant in some cases and typically does not allow the computation of all possible solutions (theoretically 16 possible and admissible (without limitations) solutions in the case of a six axis (revolute) joint manipulator). (Siciliano & Khatib 2008, 28)

The problem of the existence of multiple solutions of the inverse kinematics needs to be considered especially in the robotic simulation context and is exemplarily shown in FIGURE 4.5 below. From the mathematical perspective, both poses (solid and dashed lines) represent a valid solution for same pre-determined pose and orientation of the end effector.

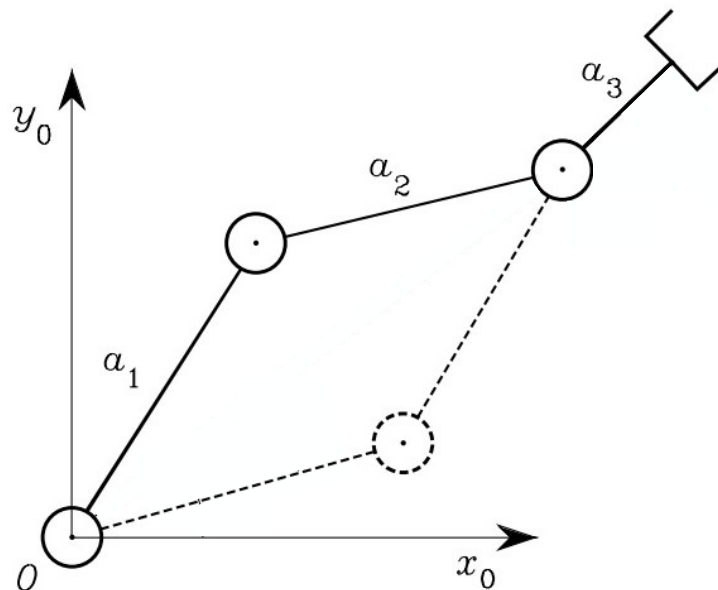


FIGURE 4.5: Example of a multiple solution problem of inverse kinematics (Siciliano, Sciavicco, Villani & Oriolo 2009, 93, modified)

The problem of an infinite number of solutions of the inverse kinematics is related the superior topic of the so called singularities, which also play an important role in robotics theory. In this context, an example related to the description of the end effector's position orientation is discussed subsequently.

Therefore, continuing from equation (4.5):

While the three elements of the vector  $\mathbf{p}_6^0$  are sufficient to determine the tool position within the reference frame, the orientation of the tool needs to be described with nine values, due to each of the unit vectors  $\mathbf{x}_6^0$ ,  $\mathbf{y}_6^0$ ,  $\mathbf{z}_6^0$  is  $3 \times 1$ -dimensional. The orientation of the tool frame can also be described with only three variables  $\alpha, \beta, \gamma$ , the so called Euler angles, and with the help of three rotational matrices (Weber 2017, 39). Amongst a number of other possible variants of the Euler angles formulation, in this case the order Z-Y-X was used to meet the thesis requirements and to be in line with the standard MATLAB formulation. To obtain the unit vectors from the Euler angles, three rotational transformations need to be applied to the tool frame in the corresponding and predetermined order:

$$\mathbf{R}_0^6 = \mathbf{R}_z(\alpha) \mathbf{R}_{y'}(\beta) \mathbf{R}_{x''}(\gamma) \quad (4.10)$$

Whereby:

$$\mathbf{R}_{\dots}(\dots) = \begin{bmatrix} \cos(\dots) & -\sin(\dots) & 0 \\ \sin(\dots) & \cos(\dots) & 0 \\ 0 & 0 & 1 \end{bmatrix} \quad (4.11)$$

Similar to the all other vector/ matrix robotics kinematic theory formulations, the Euler angles formulation suffers from non-uniqueness and can cause serious problems like the so called “Gimbal Lock”, e.g. in the case of a spherical robot manipulator’s wrist (as applicable for the industrial robot type of this thesis work).

For appropriate and more efficient computation, modern (numerical) robotic calculation algorithms, e.g. like implemented in MATLAB, are based on the usage of the so called and, most important, unique unit quaternion:

$$\epsilon = \epsilon_0 + \epsilon_1 i + \epsilon_2 j + \epsilon_3 k \quad (4.12)$$

Whereby  $\epsilon_0, \epsilon_1, \epsilon_2$  and  $\epsilon_3$  are scalars and  $i, j$  and  $k$  are operators satisfying:

$$i^2 = j^2 = k^2 = ijk = -1 \quad (4.13)$$



Euler angles can be converted into the unit quaternion and vice versa, e.g. using standard conversion tables obtainable from respective literature. (Siciliano & Khatib 2008, 13)

As the Euler angle formulation is more comprehensible and therefore used for user input purposes, the Euler angles are converted to the unit quaternion using the MATLAB `eul2quat()` function before passed in to the MATLAB inverse kinematics solver.

The problem of not admissible solutions caused by the specific manipulator's kinematic structure can be solved by the provision of a sufficient number of degrees of freedom and/ or an appropriate set of workspace limitations.

#### **4.1.4 Dynamics**

As the theory of kinematics describes the robot manipulator's motion without the consideration of any forces or torques, the theory of dynamics covers the scope of kinematics as well as forces and torques. Equal to the theory of kinematics, the theory of dynamics can also be split up into direct (forward) and inverse dynamics.

Direct dynamics theory typically describes the robot manipulator's joint motion (accelerations), from which forces and torques can be calculated, for any given joint actuation forces/ torques. Inverse dynamics in turn allow determining the joint actuation forces required for any specific robotic manipulator's motion (specified by a trajectory). In the context of practical applications, direct dynamics are usually used for simulation purposes. Inverse dynamics in contrast are typically implemented for the appropriate calculations related to feedforward (FF) control (as part of control system structures). (Siciliano & Khatib 2008, 36)

Due to the fact that control system design was not part of this thesis work, the theory of inverse dynamics is not discussed any further.

Additionally and according to the purpose of the thesis work at hand, the robot manipulator's dynamics of the simulation model created were not to be solved by the author by own program codes.

Instead, the simulation model's dynamics are solved numerically within the "background" of the Simulink Simscape Multibody environment. Therefore, the (direct) dynamics theory is only slightly touched in this section in order to present the fundamental coherences between the general theory and the simulation model.

Most commonly, dynamics of robot manipulators are described by either the Newton–Euler formulation or the Lagrange formulation (Siciliano & Khatib 2008, 44). Using the joint-space, neglecting external forces applied to the robot manipulator (free robotic motion) but considering gravitational and frictional effects, both the Newton–Euler formulation and the Lagrange formulation will lead to (Kelly, Santibáñez & Loría 2005, 77):

$$\mathbf{M}(\mathbf{q})\ddot{\mathbf{q}} + \mathbf{C}(\mathbf{q}, \dot{\mathbf{q}})\dot{\mathbf{q}} + \mathbf{g}(\mathbf{q}) + \mathbf{f}(\dot{\mathbf{q}}) = \boldsymbol{\tau} \quad (4.14)$$

Whereby:

$$\mathbf{q}, \dot{\mathbf{q}}, \ddot{\mathbf{q}} \quad (4.15)$$

Are the  $n$ -dimensional vectors of the (revolute) joint positions, velocities and accelerations.

The  $n \times n$ -dimensional matrix:

$$\mathbf{M}(\mathbf{q}) \quad (4.16)$$

Is the (joint-space related) inertia matrix. In the context of the simulation model, the inertia matrix is represented by the rigid bodies (links) of the Simulink Simscape Multibody simulation model and their individual properties (masses, Centers of Mass (CoM), Moments of Inertia (MoI) and Products of Inertia (PoI)), derived from the imported CAD model assembly of the industrial robot.

The  $n \times n$ -dimensional matrix:

$$C(\mathbf{q}, \dot{\mathbf{q}}) \quad (4.17)$$

Is the so called centrifugal and Coriolis matrix. In combination with the vector of the joint velocities, the centrifugal and Coriolis matrix represents the centrifugal and Coriolis torques of the dynamic model evoked by the robotic motion. In the context of the simulation model, the centrifugal and Coriolis phenomena are fully covered by the Simscape Multibody simulation environment.

The  $n$ -dimensional vector:

$$\mathbf{g}(\mathbf{q}) \quad (4.18)$$

Is the vector of gravitational torques. This vector represents torques within the dynamic model evoked by the effect of gravitational accelerations applied to every element (rigid bodies; links) of the simulation model.

In the context of the simulation model, gravitational effects are simulated within the Simulink Simscape Multibody environment. Also refer to section 3.2.

The  $n$ -dimensional vector:

$$\mathbf{f}(\dot{\mathbf{q}}) \quad (4.19)$$

Is the friction torque vector. The appearance of the friction vector depends on the applied frictional model. In the case at hand, a common static friction model considering viscous ( $\mathbf{F}_{m1}$ ) and Coulomb frictions ( $\mathbf{F}_{m2}$ ) was applied and therefore the friction vector  $\mathbf{f}$  is substituted by:

$$\mathbf{f}(\dot{\mathbf{q}}) = \mathbf{F}_{m1} \dot{\mathbf{q}} + \mathbf{F}_{m2} \text{sign}(\dot{\mathbf{q}}) \quad (4.20)$$

Whereby  $F_{m1}$  and  $F_{m2}$  are  $n \times n$ -dimensional diagonal matrices, containing the individual constant friction coefficients/ torques (Kelly, Santibáñez & Loría 2005, 76).

In the context of the simulation model, the viscous friction coefficient matrix  $F_{m1}$  is represented by the entirety of the viscous damping/ friction parameters and the Coulomb friction torque matrix  $F_{m2}$  is represented by the entirety of the Coulomb friction torque parameters applied to the Simulink Simscape simulation model.

The  $n$ -dimensional vector:

$$\tau \tag{4.21}$$

Contains the individual joint torques. In the context of the simulation model, the torques of the joint actuation motor models, transformed by the transmission gear models, are applied to the joint torque vector of the dynamic model. Based on that the simulation model's motion is solved within the Simulink Simscape Multibody environment.

## 4.2 Motion Planning

In contrast to the theories of kinematics (section 4.1.2 and section 4.1.3) and the theory of dynamics (section 4.1.4), motion planning is not automatically calculated by MATLAB Simulink at any point, therefore, the theory of motion planning is discussed more comprehensively and more detailed.

The purpose of robotic manipulators is the execution of predefined tasks within their workspace, based on controlled motions of the combination of the elements of the complete robotic system. In this context, typically two main problems need to be considered and solved: The avoidance of collisions of the manipulator's elements with the environment and the correct positioning and orientation of the manipulator's tool.

Because collision avoidance was not a part of the thesis work, the subsequent theory only focuses on motion planning with emphasis to the manipulator's tool. Depending on the task to be accomplished, a proper motion type needs to be chosen from the range of existing types of robotic motions such as, point-to-point (PTP) movements, linear movements, circular movements etc. According to the task definition, only linear and joint movements were taken into consideration. After the determination of a motion type, the motion needs to be planned in order to obtain set values, also references, for the closed-loop control system structures of the manipulator.

The task of motion planning is part of the covering topic "robot navigation" which in turn can be divided into three sub tasks: path planning, trajectory planning and control design (Kelly, Santibáñez & Loría 2005, 13).

Path planning covers the determination of a curve between the initial and the final position and orientation of the manipulator's end effector, avoiding collisions with obstacles (Kelly, Santibáñez & Loría 2005, 14).

Control design is thematised in section 4.3.

Trajectory planning is about generating a time dependent trajectory from the curve obtained during the process of path planning, typically defined in workspace coordinates. (The so obtained trajectory is also called reference trajectory). (Kelly, Santibáñez & Loría 2005, 14)

Based on the demands from the task definition, in this case joint and linear (referring to the welding application) movement types were considered only. In this context, motion planning is also narrowed to movements from a starting (A) to target (B) position (and orientation of the manipulator's tool).

As the joint movement type does not require following a defined path of the manipulator's tool within the workspace and already operates in joint (space) coordinates, no path planning is required. Linear motions by contrast require a defined movement of the tool along a defined (linear) path. In general, this defined paths, also called continuous paths (CP), are formulated in workspace coordinates and require a sufficient path planning in order to enable the robot following the demanded path.

### 4.2.1 Linear Trajectory Planning

If not defined divergently, the theory of this section 4.2.1 and the subsequent section 4.2.2 base exclusively on the theory described in Weber (2017, 71-105).

A linear path (defined in the robotic manipulator's workspace), to be followed by the robot's TCP (e.g. for the creation of a weld seam), can be described by a vector  $\mathbf{p}_p$ . This vector is defined as:

$$\mathbf{p}_p = \mathbf{p}_{tgt} - \mathbf{p}_{stt} \quad (4.22)$$

Whereby the  $\mathbf{p}_{tgt}$  vector is a vector aiming at the target point "B" of the linear workspace path and the  $\mathbf{p}_{stt}$  vector is a vector aiming at the start point "A" of the linear workspace path. Both are position vectors originated in the origin of the reference frame ( $O_0$ ). See FIGURE 4.6 below.

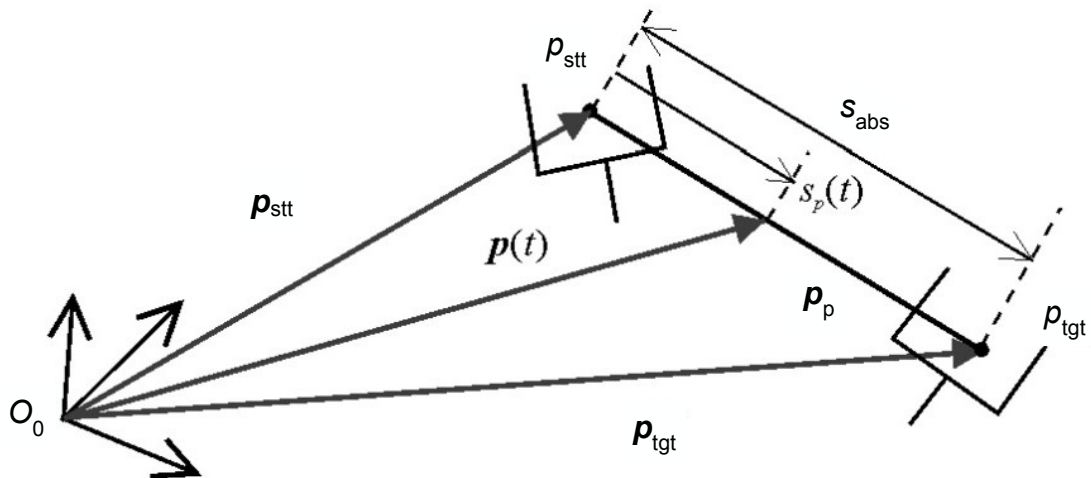


FIGURE 4.6: Vector based linear workspace path formulation (Weber 2017, 86, modified)

The time dependent movement of the TCP along the vector  $\mathbf{p}_p$  can be described with the help of the newly introduced, scalar and time dependent path parameter  $s_p(t)$ .

Furthermore, time counting starts with the start of the motion at  $t_{start} = t_0 = 0$  s, hence:

$$s_p(t_{\text{start}}) = s_p(t_0) = s_p(0 \text{ s}) = 0 \text{ m} \quad (4.23)$$

Accordingly, time counting ends with the end of the motion at  $t_{\text{end}}$ , hence:

$$s_p(t_{\text{end}}) = s_{\text{abs}} \quad (4.24)$$

Whereby  $s_{\text{abs}}$  is the absolute length of the vector  $\mathbf{p}_p$  and calculated from:

$$s_{\text{abs}} = |\mathbf{p}_p| = |\mathbf{p}_{\text{tgt}} - \mathbf{p}_{\text{stt}}| = \sqrt{(p_{\text{tgt},z} - p_{\text{stt},z})^2 + (p_{\text{tgt},y} - p_{\text{stt},y})^2 + (p_{\text{tgt},x} - p_{\text{stt},x})^2} \quad (4.25)$$

The time dependent workspace trajectory vector  $\mathbf{p}(t)$ , originated in the origin of the reference frame ( $O_0$ ) and pointing at the desired contemporary TCP workspace position (= origin of the tool frame  $O_6$ ) can be now described as:

$$\mathbf{p}(t) = \mathbf{p}_{\text{stt}} + \left[ s_p(t) \cdot \frac{(\mathbf{p}_{\text{tgt}} - \mathbf{p}_{\text{stt}})}{s_{\text{abs}}} \right] \quad (4.26)$$

As the variables  $\mathbf{p}_{\text{tgt}}$  and  $\mathbf{p}_{\text{stt}}$  are initially directly defined (e.g. by a user input) and the variable  $s_{\text{abs}}$  only needs to be calculated once, the path parameter  $s_p(t)$  needs to be described more detailed.

Therefore, the necessary assumption of a motion starting from a resting state and ending at a resting state is made, thus:

$$\dot{s}_p(t_{\text{start}}) = \dot{s}_p(0 \text{ s}) = v_p(0 \text{ s}) = 0 \frac{\text{m}}{\text{s}} \quad (4.27)$$

And:

$$\dot{s}_p(t_{\text{end}}) = v_p(t_{\text{end}}) = 0 \frac{\text{m}}{\text{s}} \quad (4.28)$$

For any further determination of the path parameter  $s_p(t)$ , its specific course needs to be determined. This is typically accomplished by the determination of the path velocity  $\dot{s}_p(t) = v_p(t)$  by a so called velocity profile or by the determination of the path acceleration  $\ddot{s}_p(t) = a_p(t)$  by a so called acceleration profile.

As a simple trapezoidal velocity profile only consists of linear and continuous functions, the derivation of the velocity (profile)  $\dot{s}_p(t)$ , the acceleration (profile)  $\ddot{s}_p(t)$  reveals that the motion is not (sufficiently) jerk-free (jerk  $j = \ddot{\ddot{s}}_p(t)$ ), see FIGURE 4.7.

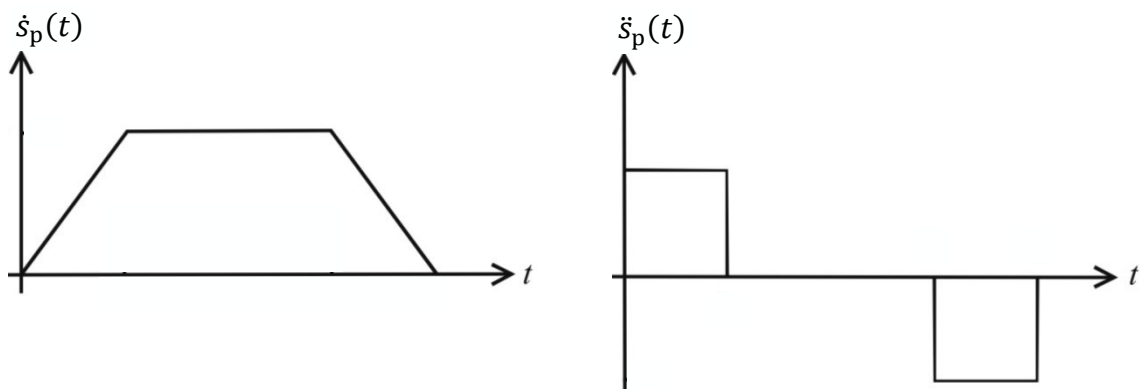


FIGURE 4.7: Trapezoidal velocity profile (left) and the corresponding acceleration profile (right) (Weber 2017, 75, modified)

This in turn could cause harmful vibrations and stresses of the robotic system. In order to obtain a smooth motion, a sinusoidal acceleration profile was chosen. The motion is basically described by (general formulation):

$$\ddot{s}(t) = \hat{a} \cdot \sin^2\left(\frac{\pi}{t_s} \cdot t\right) \quad (4.29)$$

(Whereby  $\hat{a}$  describes the peak value of the acceleration and  $t_s$  describes the length of the considered time span) and, along with the corresponding velocity profile and the corresponding position graph, depicted in FIGURE 4.8 below.



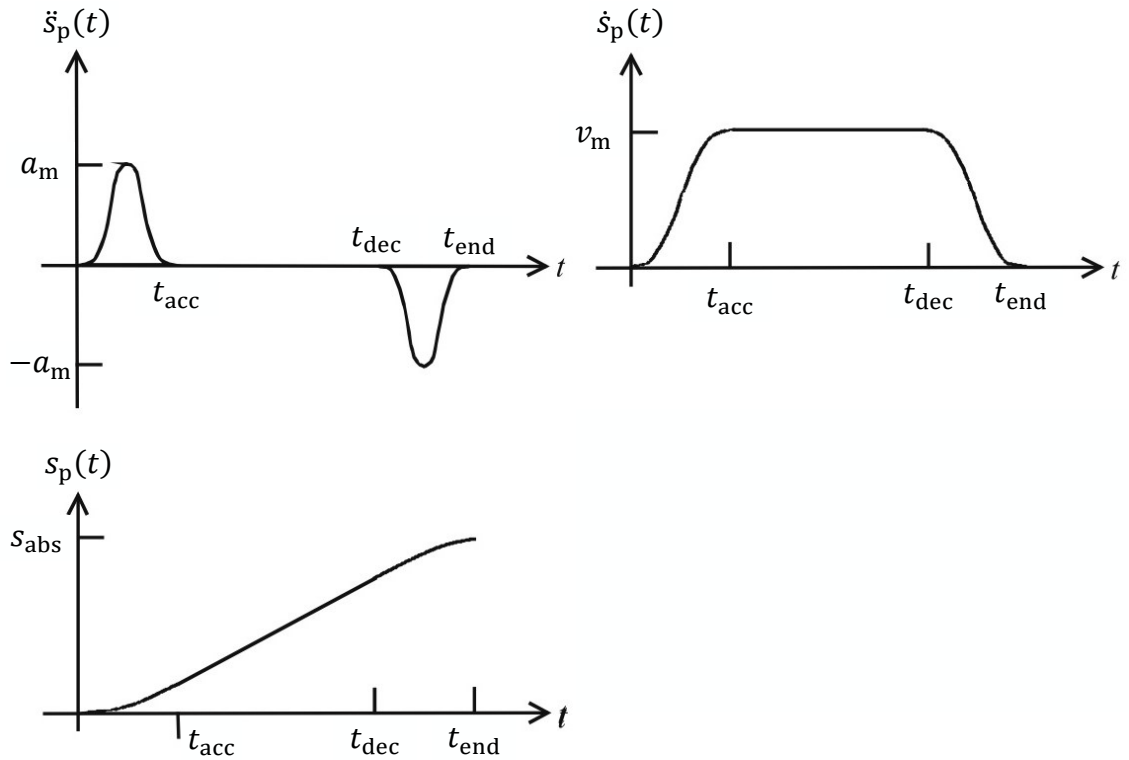


FIGURE 4.8: Sinusoidal acceleration profile and corresponding velocity and position graphs (Weber 2017, 79, modified)

For a further mathematical description, firstly the new time variables  $t_{acc}$ ,  $t_{dec}$  and  $t_{end}$  are introduced accordingly to the definition shown in FIGURE 4.8 above.

As the aim of the determination of the sinusoidal acceleration profile is to obtain an appropriate description of the path parameter  $s_p(t)$ , equation (4.29) needs to be integrated two times for each of the three distinctive motion phases:

The motion phase of acceleration ( $0 \leq t < t_{acc}$ ) is described by:

$$s_p(t) = a_m \cdot \left\{ \left( \frac{1}{4} \cdot t^2 \right) + \left[ \frac{t_{acc}^2}{8\pi^2} \cdot \left( \cos \left( \frac{2\pi}{t_{acc}} \cdot t \right) - 1 \right) \right] \right\} \quad (4.30)$$

The motion phase of continuous velocity ( $t_{acc} < t \leq t_{dec}$ ) is described by:

$$s_p(t) = v_m \cdot \left[ t - \left( \frac{1}{2} \cdot t_{acc} \right) \right] \quad (4.31)$$

The motion phase of deceleration ( $t_{\text{dec}} < t \leq t_{\text{end}}$ ) is described by:

$$s_p(t) = \frac{a_m}{2} \cdot \left\{ [t_{\text{end}} \cdot (t + t_{\text{acc}})] - \left[ \frac{(t^2 + t_{\text{end}}^2 + 2 \cdot t_{\text{acc}}^2)}{2} \right] + \left[ \frac{t_{\text{acc}}^2}{4\pi^2} \cdot \left( 1 - \cos\left(\frac{2\pi}{t_{\text{acc}}}\cdot(t - t_{\text{dec}})\right) \right) \right] \right\} \quad (4.32)$$

Whereby the value  $a_m$  describes the maximum applied (desired) path acceleration value and  $v_m$  describes the maximum applied (desired) path velocity value. Both are necessarily predetermined by user inputs or obtained from any other source in advance.

In the case of short trajectory and/ or high acceleration values, the maximum desired path velocity value  $v_m$  can possibly not be reached and needs to be adapted to the maximum reachable path velocity. For obvious practical matters, the value of the maximum applied path acceleration  $a_m$  is kept constant. Starting from this, the maximum reachable path velocity can be calculated from:

$$v_{m,\text{max}} = \sqrt{\frac{a_m \cdot s_{\text{abs}}}{2}} \quad (4.33)$$

Then be compared to the maximum desired path velocity value  $v_m$  and adapted if necessary. The procedure is depicted in the flow chart shown in FIGURE 4.9: below.

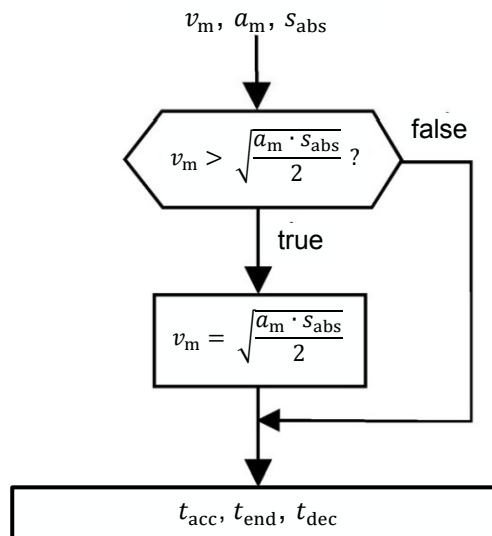


FIGURE 4.9: Flow chart: adaption of the applicable path velocity (Weber 2017, 77, modified)

Finally, the time variables  $t_{\text{acc}}$ ,  $t_{\text{dec}}$  and  $t_{\text{end}}$  are calculated from (same length of the acceleration and deceleration phases (= symmetric velocity profile)):

$$t_{\text{acc}} = \frac{2 \cdot v_m}{a_m} \quad (4.34)$$

$$t_{\text{end}} = \frac{s_{\text{abs}}}{v_m} + t_{\text{acc}} \quad (4.35)$$

$$t_{\text{dec}} = t_{\text{end}} - t_{\text{acc}} \quad (4.36)$$

## 4.2.2 Joint Trajectory Planning

In contrast to the linear trajectory planning which requires subsequent solving of the inverse kinematics in order to obtain the joint-space trajectory from the workspace trajectory, joint trajectory planning is directly accomplished in the joint-space.

The general procedure of joint trajectory planning is similar to the procedure of linear trajectory planning and also uses a sinusoidal acceleration profile, but the path parameter is now:

$$\mathbf{q}_p(t) = [q_{1,p}(t) \quad q_{2,p}(t) \quad q_{3,p}(t) \quad q_{4,p}(t) \quad q_{5,p}(t) \quad q_{6,p}(t)]^T \quad (4.37)$$

Instead of  $s_p(t)$ . Hence, the joint trajectory planning needs to be applied for each of the joint position variables (= robot manipulator's axes) individually, six times in this particular case.

$$\mathbf{q}(t) = [q_1(t) \quad q_2(t) \quad q_3(t) \quad q_4(t) \quad q_5(t) \quad q_6(t)]^T \quad (4.38)$$

In theory, the individual joint trajectory planning procedures could be executed independently from each other's, which is called asynchronous motion and typically causes non-obvious trajectory courses and a higher overall mechanical stress level of the robotic manipulator.



And:

$$v_m = v_{i,m} \left[ \frac{\text{rad}}{\text{s}} \right] \quad (4.42)$$

$$a_m = a_{i,m} \left[ \frac{\text{rad}}{\text{s}^2} \right] \quad (4.43)$$

$$t_{\text{acc}} = t_{i,\text{acc}} \quad (4.44)$$

$$t_{\text{dec}} = t_{i,\text{dec}} \quad (4.45)$$

$$t_{\text{end}} = t_{i,\text{end}} \quad (4.46)$$

Hence:

$$v_{i,m,\text{max}} = \sqrt{\frac{a_{i,m} \cdot q_{i,\text{abs}}}{2}} \quad (4.47)$$

$$t_{i,\text{acc}} = \frac{2 \cdot v_{i,m}}{a_{i,m}} \quad (4.48)$$

$$t_{i,\text{end}} = \frac{q_{i,\text{abs}}}{v_{i,m}} + t_{i,\text{acc}} \quad (4.49)$$

$$t_{i,\text{dec}} = t_{i,\text{end}} - t_{i,\text{acc}} \quad (4.50)$$

Then the maximum traveling time is calculated from:

$$t_{\text{end,max}} = \max(t_{i,\text{end}}) \quad (4.51)$$

This also identifies the leading axis. If once determined, the maximum traveling time is now valid for all axes, thus:

$$t_{i,\text{end}} = t_{\text{end,max}} \quad (4.52)$$

Accordingly, the values of  $t_{\text{acc}}$  and  $t_{\text{dec}}$  of the leading axis are applied for all other axes:

$$t_{i,\text{acc}} = t_{\text{acc,max}} \quad (4.53)$$

$$t_{i,\text{dec}} = t_{\text{end,max}} \quad (4.54)$$

Subsequently, the individual maximum desired path velocity and acceleration values of the non-leading axes need to be adapted to the new time variable values with:

$$v_{i,m} = \frac{q_{i,\text{abs}}}{t_{i,\text{dec}}} \quad (4.55)$$

And:

$$a_{i,m} = \frac{2 \cdot v_{i,m}}{t_{i,\text{acc}}} \quad (4.56)$$

Finally, the individual joint-space trajectories can be calculated from:

$$q_i(t) = q_{i,\text{stt}} + \left[ q_{i,p}(t) \cdot \frac{(q_{i,\text{tgt}} - q_{i,\text{stt}})}{q_{i,\text{abs}}} \right] \quad (4.57)$$

Using:

The motion phase of acceleration ( $0 \leq t < t_{i,\text{acc}}$ ) is described by:

$$q_{i,p}(t) = a_{i,m} \cdot \left\{ \left( \frac{1}{4} \cdot t^2 \right) + \left[ \frac{t_{i,\text{acc}}^2}{8\pi^2} \cdot \left( \cos \left( \frac{2\pi}{t_{i,\text{acc}}} \cdot t \right) - 1 \right) \right] \right\} \quad (4.58)$$

The motion phase of continuous velocity ( $t_{i,acc} < t \leq t_{i,dec}$ ) is described by:

$$q_{i,p}(t) = v_{i,m} \cdot \left[ t - \left( \frac{1}{2} \cdot t_{i,acc} \right) \right] \quad (4.59)$$

The motion phase of deceleration ( $t_{i,dec} < t \leq t_{i,end}$ ) is described by:

$$q_{i,p}(t) = \frac{a_{i,m}}{2} \cdot \left\{ [t_{i,end} \cdot (t + t_{i,acc})] - \left[ \frac{(t^2 + t_{i,end}^2 + 2 \cdot t_{i,acc}^2)}{2} \right] + \left[ \frac{t_{i,acc}^2}{4\pi^2} \cdot \left( 1 - \cos \left( \frac{2\pi}{t_{i,acc}} \cdot (t - t_{i,dec}) \right) \right) \right] \right\} \quad (4.60)$$

### 4.3 Control Systems

As the topic of robotic manipulator control is highly sophisticated and comprehensive, only coarse and narrowed outlines of the topic are discussed in this section. Furthermore, the scope of the thesis work did not cover any control system design, but common control system theory was considered in order to create an environment that meets the requirements for state of the art control systems design.

In a first distinction, the field of robot control can be divided into the area of force control and the areas of position/ motion control, whereby hybrid forms also exist. The force control is typically meant for the purpose of the control of forces and torques applied from the robot to its environment or vice versa (Weber 2017, 23).

In the case at hand, only free robotic motion was considered and therefore, the topic of force control was not discussed any further.

Continuing from that, position control is related to the task of controlling the robot manipulator to reach a specific set-point (Kelly, Santibáñez & Loría 2005, 135). Motion control in contrast is related to the task of controlling desired manipulator's motions, or more precise, following a desired trajectory (Kelly, Santibáñez & Loría 2005, 224).

Both position and motion control are independent from external forces and torques.

Furthermore, a distinction can be made between internal (joint) robot control and external robot control as shown in the subsequent FIGURE 4.11.

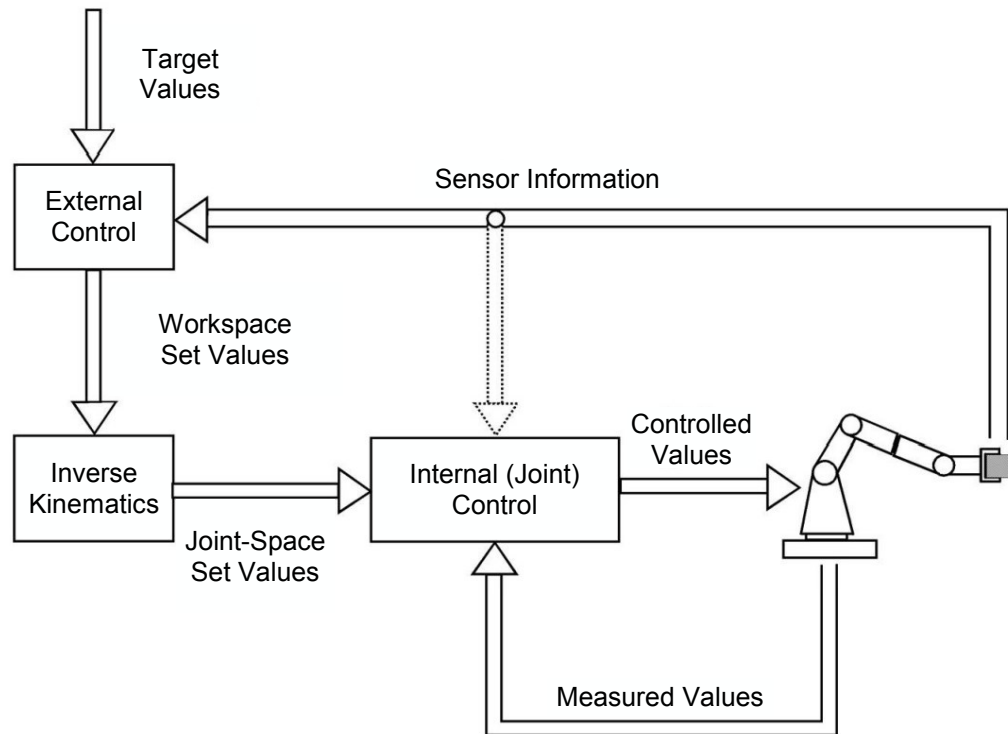


FIGURE 4.11: Schematic depiction of internal and external robot control (Weber 2017, 25, modified)

From FIGURE 4.11 above can be obtained that external robot control requires:

- Additional external measuring devices
- Inverse kinematics solving
- Subsequent internal control system structures

Because external measuring devices were not covered by the thesis scope, the theory of external robot control was neglected but without excluding possible future implementations of external control to the simulation model, because internal robot control is subordinated to external robot control.



As joint actuations (e.g. by servomotors) as well as joint variable measurements (e.g. by the servomotor's resolver) are typically joint-space related, internal robot control can be considered as the fundamental control system type of a common industrial robot.

Following this, it must be considered that changes of one specific joint variable in most of the cases also causes impacts on all other joint variables of the robotic manipulator's system, therefore multi-variable control is required for appropriate control of the typically non-linear coupled robotic structure. (Weber 2017, 25)

In this context, another distinction is made between the so called centralized and decentralized control. Decentralized control bases on the assumption of a robotic manipulator consisting of a number of  $n$  independent systems to be controlled ( $n$  joint variables), whereby coupling effects are treated as disturbances (Siciliano, Sciavicco, Villani & Oriolo 2009, 309).

Centralized control in contrast includes the decentralized control structures but also considers the inter-system connections and dependencies (influences based on the typically non-linear couplings within the simulation model) (Siciliano, Sciavicco, Villani & Oriolo 2009, 327).

Based on that, the superordinate structure, centralized control, was again not further considered but without excluding possible future implementations of a centralized control to the simulation model.

A typical and common scheme of a closed-loop single-input single-output (SISO) decentralized control system structure, including up-streamed inverse kinematics, is shown below (FIGURE 4.12).

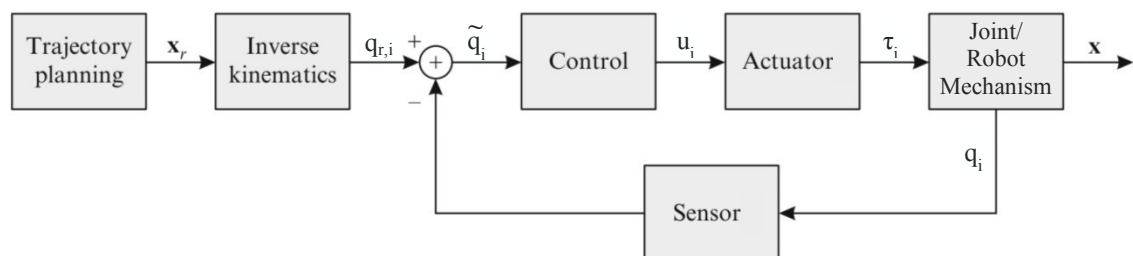


FIGURE 4.12: Decentralized SISO control system structure (Bajd, Mihelj, Lenarčič, Stanovnik & Munih 2010, 78, modified)

Theoretically, decentralized control system structures can be of the types:

- Single-input single-output (single variable control without feedforward control (e.g. individual joint position  $q_i$ ))
- Multiple-input single-output (MISO) (single variable control with feedforward control (e.g. individual joint position  $q_i$ ))
- Single-input multiple-output (SIMO) (cascaded control system without feedforward control)
- Multiple-input multiple-output (MIMO) (cascaded control system with feedforward control)

Whereby the cascaded (SIMO) control system structures, as shown in FIGURE 4.13, are proven and common in the context of all kinds of (electric motor driven) positioning tasks.

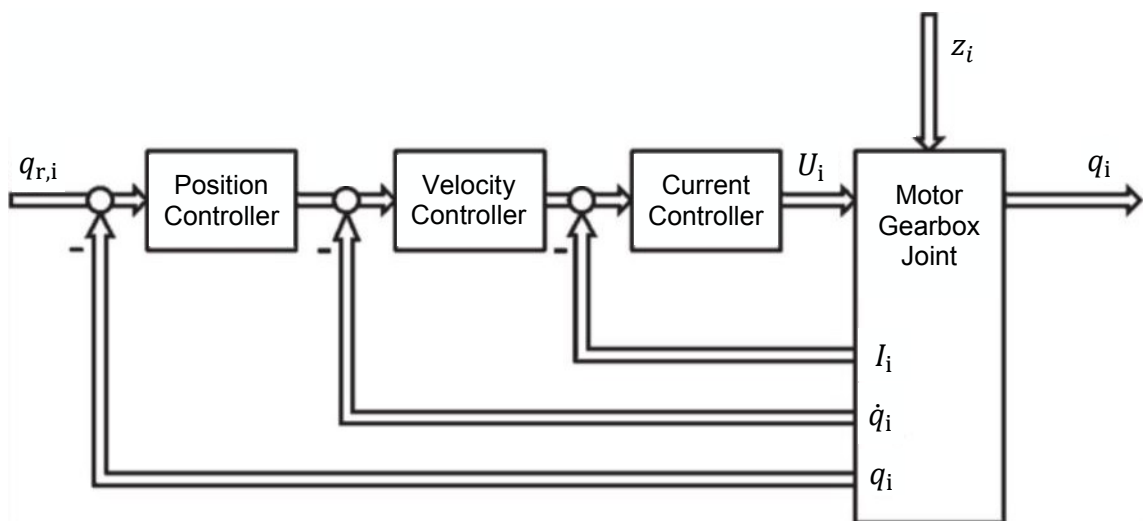


FIGURE 4.13: Schematic depiction of a decentralized cascaded SIMO control system structure (Grote, Bender & Göhlich 2018, T112, modified)

This cascaded decentralized structure can be extended to a more performant MIMO structure with moderate efforts by the implementation of a superior centralized feedforward control system (also called Computed Torque Feedforward Control), as shown in the subsequent FIGURE 4.14.

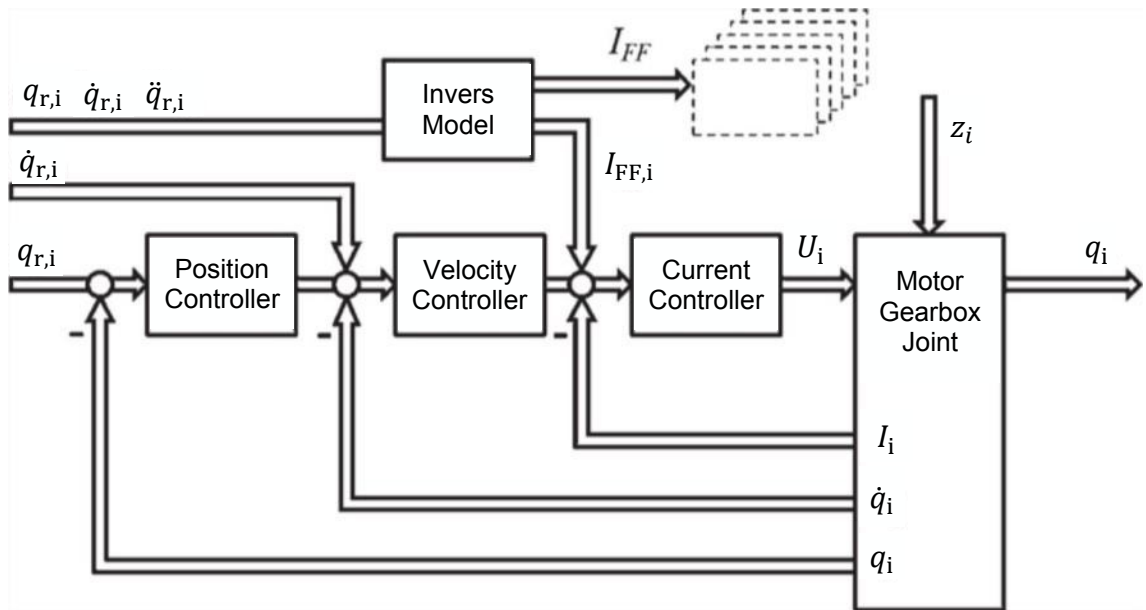


FIGURE 4.14: Schematic depiction of a decentralized cascaded MIMO control system structure with centralized feedforward control (Grote, Bender & Göhlich 2018, T112, modified)

#### 4.4 MATLAB Simulink

The software MATLAB is a numerical computing environment based on vector- and matrix operations, suitable for the operating systems (OS) Microsoft® Windows®<sup>6</sup>, Apple® macOS®<sup>7</sup> and Linux®<sup>8</sup>. MATLAB provides numerical calculations and visualisations using its own high-level programming language. Nowadays, MATLAB is widespread in the research, development and industry and mainly used in the context of mathematical and engineering sciences. (Pietruszka 2014, 1)

MATLAB also contains the graphical development environment Simulink. Simulink provides modelling and simulation of dynamic systems (linear and non-linear) mainly based on block diagrams. In- and outputs of the simulation can be provided and evaluated directly in the Simulink environment but also indirectly from the MATLAB Workspace or M-files. This also allows further processing of the Simulink simulation results within the MATLAB environment. (Pietruszka 2014, 167)

<sup>6</sup> Microsoft® Windows® is a registered trademark of Microsoft Corporation

<sup>7</sup> Apple® and macOS® are registered trademarks of Apple Inc.

<sup>8</sup> Linux® is a registered trademark of The Linux Foundation®

Simulink itself provides several integrated tools such as Stateflow<sup>®9</sup> or Simscape. Simscape is a tool for modelling and simulating multi-domain physical systems typically occurring within the area of mechatronic systems. Simscape is also mainly based on block diagrams, covering electrical, mechanical, and hydraulic components. (Pietruszka 2014, 353)

Focusing on mechanical issues, Simscape Multibody provides a multibody simulation environment for three-dimensional (3D) mechanical systems which also covers the import of CAD model assemblies. As it is a part of the Simulink environment, in general, Simulink functionalities can be applied in the Simscape environment and models can be parameterised using MATLAB variables and expressions. (The MathWorks Inc. 2019a)

#### 4.5 Programming

A wide variety of programming languages, types, methods and supporting tools exist, typically related to the specific problem to be solved. The problems solved in the context of the thesis work at hand were mainly related to the technical domain and the working environments were predetermined. Hence, programming was accomplished using the MATLAB programming language (text-based) within the MATLAB environment and block diagrams (graphical) within the Simulink/ Simulink Simscape environment exclusively.

Program planning was managed following the typical basic programming procedure:

1. Identification of the demanded/ required program outputs
2. Identification of the available/ required program inputs
3. Determination of the required processing program code

In this context, the outlines, the main program flow, the main characteristics and functionalities were sufficiently described and documented visually using appropriate tools for the creation of program flow charts like PapDesigner<sup>10</sup> Version 2.2.0.8.04.

---

<sup>9</sup> Stateflow<sup>®</sup> is a registered trademark of The MathWorks, Inc.

<sup>10</sup> Copyright© friedrich-folkmann.de 2017

Additionally, the basic code creation rules were applied:

- Sufficient code comments
- Consistent naming of data, variables, functions, etc.
- Modular code structures
- A descriptive header for each individual code, exemplarily shown below:

```
#####
%
% Project           :
% File Name        :
% Author           :
% Date Created     :
% Purpose          :
% Revision History :
%
% Date             Author             Revision             Changes
%
#####
```

## 5 CONCEPTUAL DESIGN

Related to the general purpose and requirements of the thesis work, the simulation model is mainly centred around a Simulink Simscape Multibody representation (block diagram) of the real robotic system, based on and derived from its CAD model.

Firstly, the level of completeness (compared to the real robotic system) of an automatically generated Simulink Simscape Multibody simulation model was examined with the help of a prepared CAD model of the ABB IRB 2600-12/1.85 industrial robot (provided by the TAMK) and the MATLAB `smimport()` function. (The procedure of the preparation of the CAD model is precisely described in section 6.1). FIGURE 5.1 shows the Simulink Simscape Multibody block diagram automatically derived from the CAD model assembly of the industrial robot.

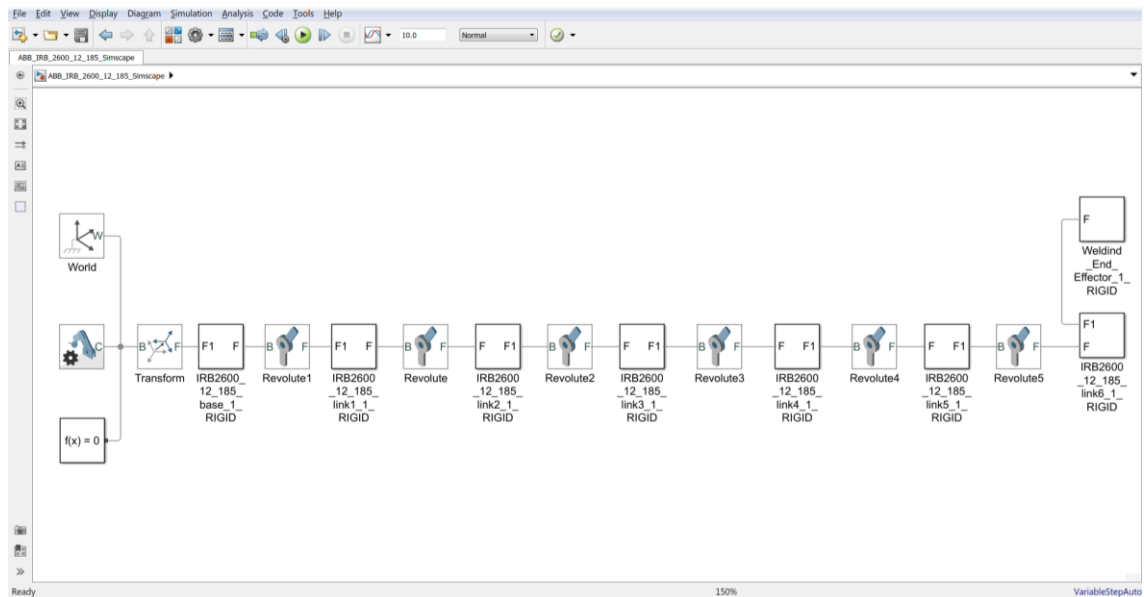


FIGURE 5.1: Automatically generated Simscape Multibody model block diagram

The block diagram fully represents the manipulator's kinematic structure, consisting of the two basic elements, (revolute) joints and rigid bodies (links) blocks. Rigid Transform blocks are used to describe the geometrical relationships between the individual links and joints.

It can be clearly seen that the joint actuation systems are completely missing.

Thus, no control, signal or measurement system structures exist. Furthermore, no parameterization was applied, except the kinematic parameters (DH-parameters, joint types) and mass/ inertia/ graphical appearance (CoM, Mol, Pol, etc.) parameters derived from the CAD model during the automatic generation.

Therefore, the conceptual design of the Simulink/ Simulink Simscape model was mainly related to the identified and summarized tasks to be accomplished listed below:

- Design of the robot manipulator's joint actuation (bearings, transmissions, motors, motor drivers (inverters))
- Design of appropriate simulation model signal processing (acquisition, routing, provision)
- Design of appropriate preparations for control system structures
- Design of measurement systems

Furthermore, the task of identifying and outlining appropriate solutions for gathering, determining, processing and providing all the required information to the simulation model was pending.

The decision was made to accomplish these tasks within the MATLAB environment, whereby the main tasks can be expressed as:

- Parameterization of the simulation model (acquisition, preparation and provision of the parameters)
- Set values (reference values, reference trajectories) calculation and provision

## **5.1 General Simulation Program Structure**

The draft of the general simulation program structure, depicted as simplified function diagram, is shown in the subsequent FIGURE 5.2.

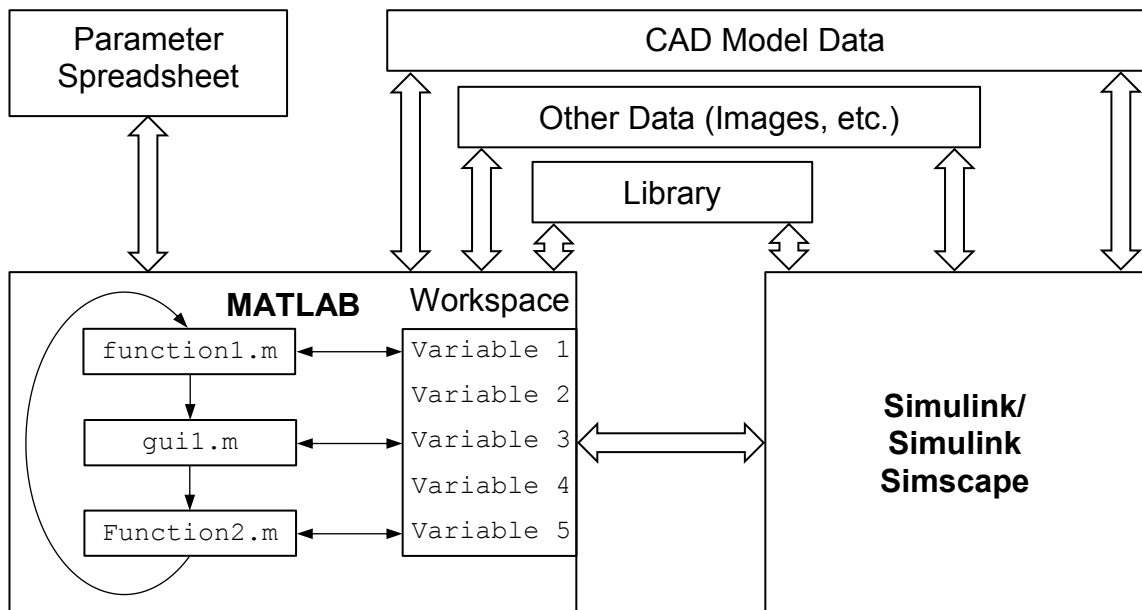


FIGURE 5.2: General simulation program structure diagram

The image shown above should be self-explanatory and was used as guidance for clearly structured progresses of the processed work.

As the required CAD data of the robotic manipulator only needed to be built and assembled once and are already provided to the simulation program, the process of the CAD model handling was not specified any further in the diagram but is discussed sufficiently in section 6.1.

## 5.2 Simulink/ Simulink Simscape

### 5.2.1 Simscape Multibody Model

Due to the similarities of the automatically generated Simulink Simscape Multibody block diagram structure (FIGURE 5.1) to the general kinematic structure (open, serial kinematic chain) of the real robotic system, the decision was made to retain the already existent general block diagram structure unchanged.

FIGURE 5.3 shows a sketch of the concept of the simulation model's Simscape block diagram.



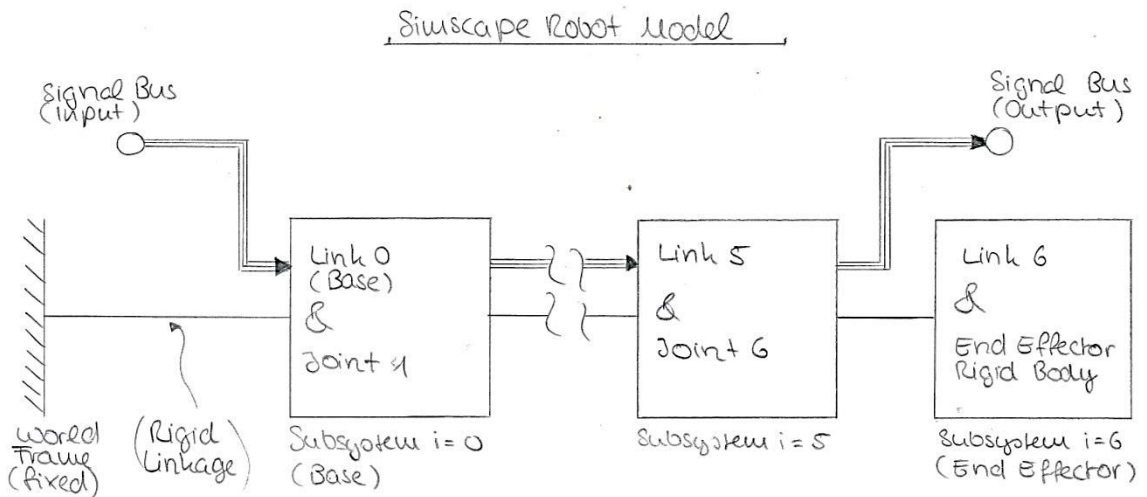


FIGURE 5.3: Freehand sketch of the concept of the simulation model's Simscape block diagram

In accordance with the (kinematic) theory (section 4.1), the number of degrees of freedom is equal to the number of joints (with  $f = 1$ ), thus the robotic structure consists of  $n = F = f = 6$  (revolute) joints and seven rigid bodies (considering link 6 and the end effector as one body due to the rigid connection). Therefore, a number of seven subsystems shall represent the kinematic structure of the manipulator. The subsystems shall be in series, rigidly linked and connected to a signal bus. Each subsystem shall mainly contain one rigid body (link  $i$ ), the ( $i+1$  revolute) joint and the corresponding joint actuation (except the end effector subsystem). The general subsystem structure shall be same for all other subsystems, except the end effector subsystem which only shall contain the end effector rigidly connected to link 6. Exemplarily, a sketch of the concept of a subsystem is shown in FIGURE 5.4 below.

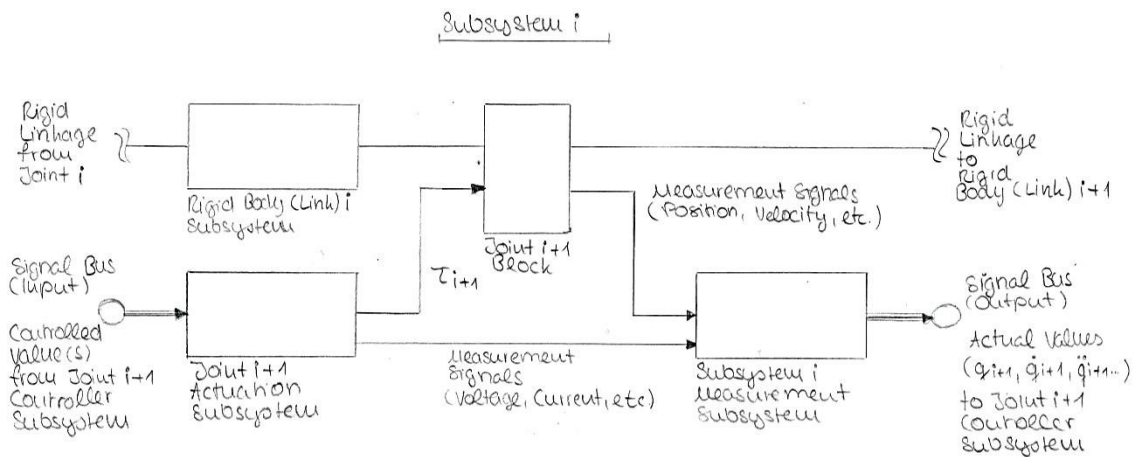


FIGURE 5.4: Freehand sketch of the concept of a subsystem of the Simscape block diagram

The main task of the conceptual design was related to the determination of general concepts and structures - the specific contents of the further (sub) subsystem were unknown at this point. Therefore, the “black-box-method” was used for the depiction of required elements/ blocks of the subsystems block diagram.

The rigid body (sub) subsystem and (revolute) joint block already existed (see FIGURE 5.1) and no further specifications were needed.

The measurement (sub) subsystem was not drafted any further as its structure and contents were highly dependent on the specific obtainable signals of the finally implemented individual elements/ blocks of the block diagram. Nevertheless, the discussion of the control system theory (section 4.3) revealed that each Subsystem  $i$  Measurement Subsystem at minimum needs to capture at minimum the corresponding:

- Joint position, velocity and acceleration variables  $q_i, \dot{q}_i, \ddot{q}_i$
- Joint torque variable  $\tau_i$

A sufficient description of the finally implemented measurement subsystem can be found from the later section 6.2.4.

A sketch of the concept of a joint actuation (sub) subsystem is shown in FIGURE 5.5 below.

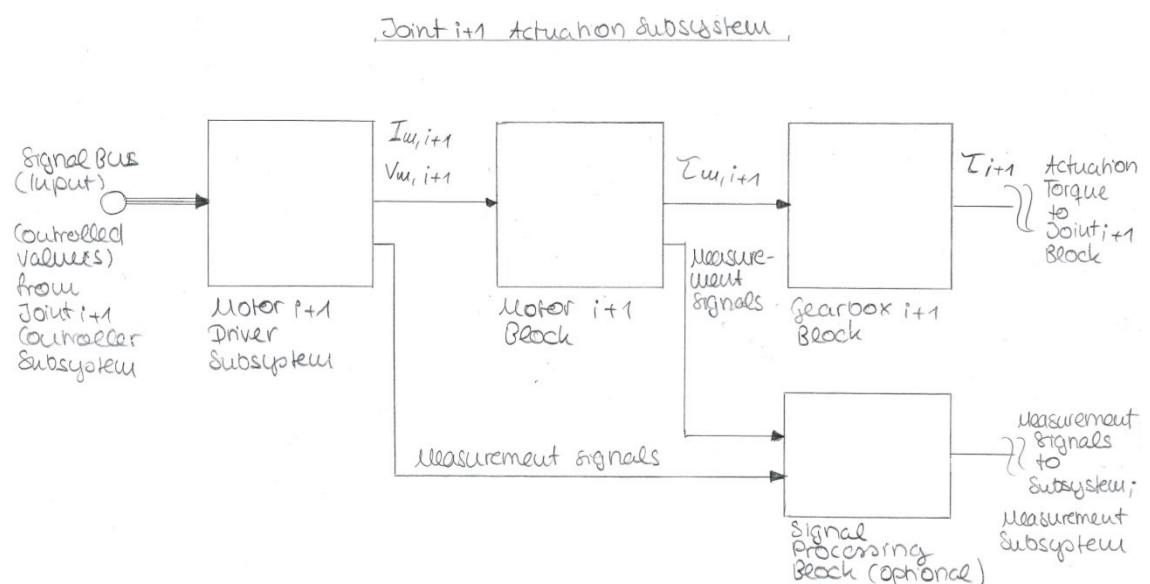


FIGURE 5.5: Freehand sketch of the concept of a joint actuation subsystem

The motor block of the Joint  $i+1$  Actuation Subsystem (FIGURE 5.5) shall be chosen from the range of predefined motor model blocks of the Simulink Simscape Library.

The motor type of the real robotic system was identified as alternating current (AC) asynchronous motor (see section 6.4.1). Therefore, the decision was made that an Asynchronous Machine Squirrel Cage (ASM) model block from the Simulink Simscape Electrical Library shall be implemented as it meets the characteristics of the real motor best (from the range of available library elements).

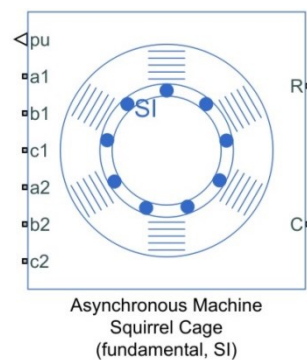


FIGURE 5.6: Screen capture of a Simulink Simscape Asynchronous Machine block

Similar to the motor block, the gearbox block of the Joint  $i+1$  Actuation Subsystem shall be chosen from the range of predefined gearbox model blocks of the Simulink Simscape Library.

The gearbox type of the real robotic system was identified as cycloidal reduction gear (see section 6.4.1). Therefore, the decision was made that a Cycloidal Drive model block from the Simulink Simscape Driveline Library shall be implemented as it is the only available and reasonable applicable library element.

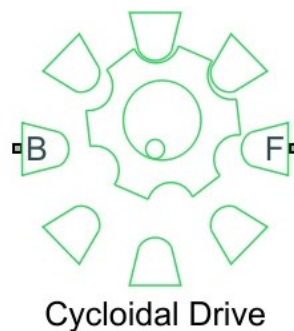


FIGURE 5.7: Screen capture of a Simulink Simscape Cycloidal Drive block

According to the requirements, predefined MATLAB/ MATLAB Simulink (Simscape) contents should always be preferred if reasonable and applicable. In the case of the motor driver subsystem, the decision was made that the required function principles and blocks of the MATLAB Simulink Simscape “Asynchronous Machine Scalar Control” (pe\_asm\_scalar\_control) example block diagram shall be utilized as it is applicable to the chosen motor model block and meets the requirements best.

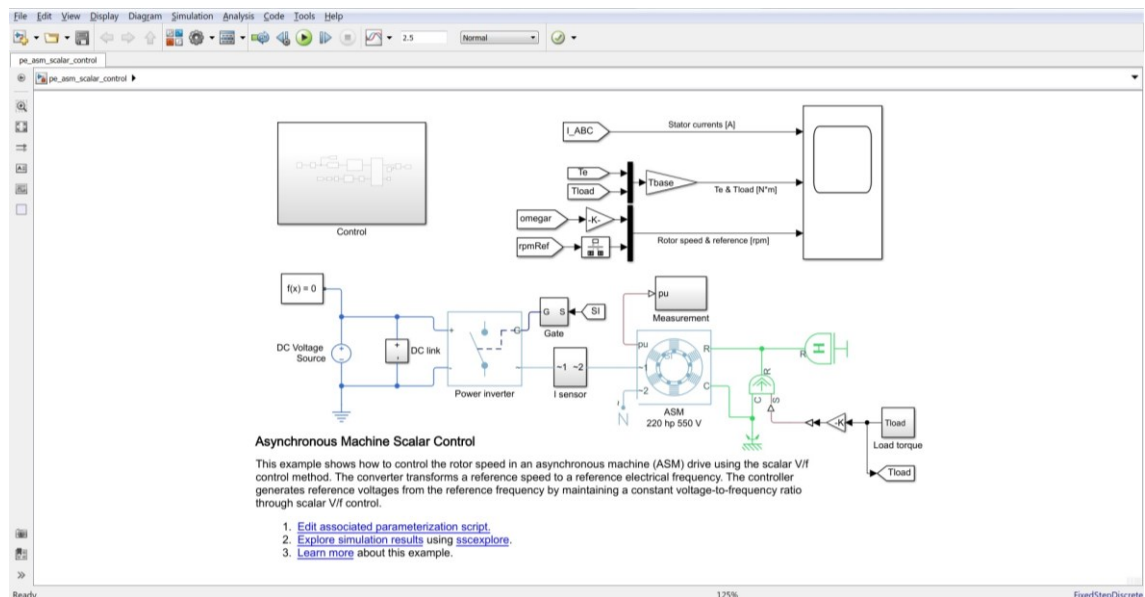


FIGURE 5.8: Screen capture of the MATLAB Simulink Simscape “Asynchronous Machine Scalar Control” example block diagram

From FIGURE 5.5 can be obtained that the motor driver subsystem input needs to be aligned to the controlled value from the joint controller subsystem. In this context, it needs to be mentioned again that control system structure design was not part of the thesis scope and therefore, later and from external parties applied control system structures and the corresponding controlled values were unknown at this point. Therefore, and in order to increase the comprehensibility of the simulation model, the decision was made that the motor driver shall be created in a way to expect scalar values (similar to the mentioned example) within the range from -1 to +1 as the controlled value. Thus: “+1” = “100% power in the positive direction” and “-1” = “100% power in the negative direction”. In the case of more sophisticated control system structure design to be applied by an external party (SIMO, MIMO), the motor driver subsystem can be adapted accordingly if required.

The signal processing block of the Joint  $i+1$  Actuation Subsystem shall only be implemented if required, e.g. for signal bundling, depending on the final structure and contents of the joint actuation subsystem unknown at this point.

Additional elements/ blocks shall also be implemented if the finally applied predefined motor and/ or gearbox blocks not include all required simulation model parameters (e.g. viscous rotor damping).

### 5.2.2 Simulink

The simulation model is meant for the purpose of control structure system design in an educational context. Therefore, the decision was made that the appearance of the Simulink block diagram shall be similar to the appearance of a common basic closed-loop control system block diagram, as shown in FIGURE 5.9 below.

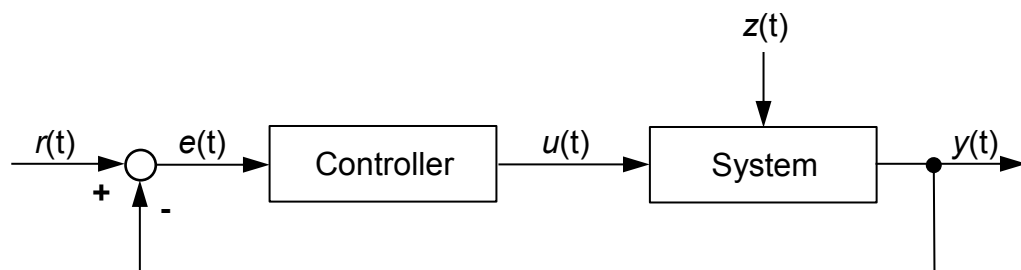


FIGURE 5.9: Block diagram of a common basic closed-loop control system structure

As Simulink Simscape (Multibody) is a sub environment of Simulink, the Simscape Multibody model of the robotic system (as described in the previous section 5.2.1) shall be implemented as a subsystem of the overlaying Simulink block diagram. In the context of FIGURE 5.9, the Multibody block diagram would be then embedded in the “System” block.

From FIGURE 5.9 can also be obtained that a controller subsystem and signaling were needed to complete the demanded Simulink block diagram structure.

Therefore, summarizing the theory formerly discussed in section 4.3 and the additionally studied theory from the literature Weber (2017), Kelly, Santibáñez & Loría (2005) and Bajd, Mihelj, Lenarčič, Stanovnik & Munič (2010):

A number of  $n$  decentralized MIMO structures is necessarily the fundamental control system structure to be provided within the simulation model in order to enable the design and implementation of any other control system structure. Hence, the control system structure to be designed and implemented in the context of the thesis work shall cover at minimum:

- A superior control system structure carrying the individual decentralized control system structures in order to provide an environment for centralized control if required
- Six independent (decentralized) control system structures, one for each of the simulation model's (revolute) joints (joint variables ( $q_1$ - $q_6$ ))
- Multiple inputs covering joint-space position set values as well as measured actual values of the joint actuation systems (e.g. joint position, velocity, acceleration and torque)
- Multiple outputs covering the joint space controlled values as well as the controlled values of the joint actuation systems (e.g. motor driver set value)

For signalling between the individual Simulink and Simulink/ Simscape subsystems and any other (sub) subsystems, all signalling options available from the Simulink Signal Routing Library were investigated and rated. Based on that, the decision was made that a signal bus shall be used, as it provides a minimum amount of signal lines (= clear overview) but maximum comprehensibility, adaptability and extensibility (compared to e.g. direct wiring or "From" & "Goto" blocks).

Because the signal bus only allows transferring Simulink domain signals, converting shall be applied for interfacing from/ to other signal domains when adding or branching off bus signals (e.g. "PS-Simulink Converter" or "Simulink-PS Converter" blocks). Furthermore, the decision was made to only transfer SI unit and derived SI unit signal values within the bus (except signals originally without unit) in order to prevent errors, as the bus signals are values without any unit. Therefore, unit conversion shall also be applied when adding or branching off bus signals if required.

### 5.2.3 Parameter Provision

In order to enable an efficient and convenient provision of the parameters of the individual elements/ blocks of the Simulink and Simulink Simscape block diagrams, parameterization shall be implemented indirectly via the simulation model variables obtainable from the MATLAB Workspace (see later section 5.3.1).

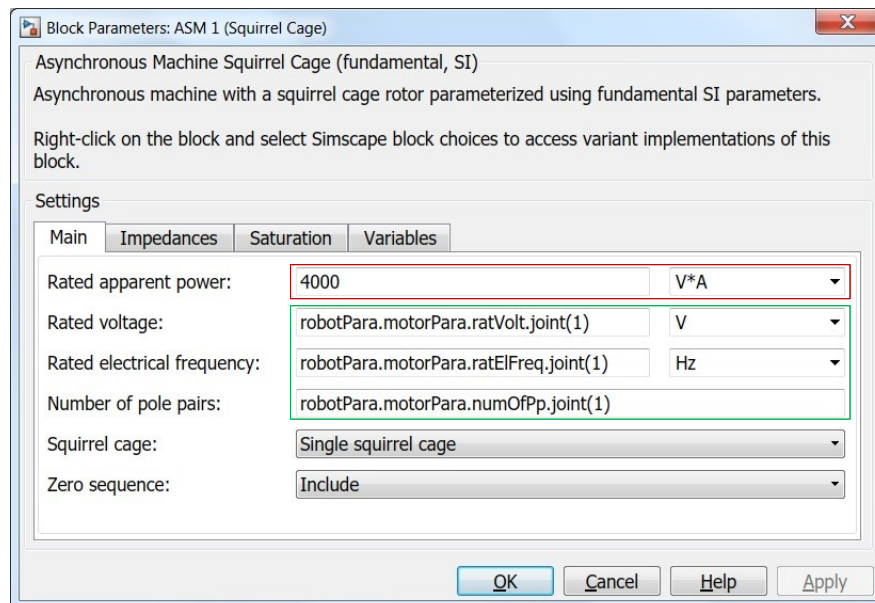


FIGURE 5.10: Screen capture of an exemplary block parameterization

This method provides a structured and centralized compilation of all simulation model parameters in one or more variables which in turn allow a quick and extensive access for viewing and/ or modifications of individual parameters.

## 5.3 MATLAB

As described in the introductory paragraphs of this section (5), the simulation program can be divided into a Simulink/ Simulink Simscape part and a MATLAB part. The previous sections 5.1 and 5.2 describe the simulation model's general overall structure, its individual elements and their functional dependencies.

In order to also determine sequential dependencies between the individual elements and functionalities, a general flow chart, shown in FIGURE 5.11 below was created.

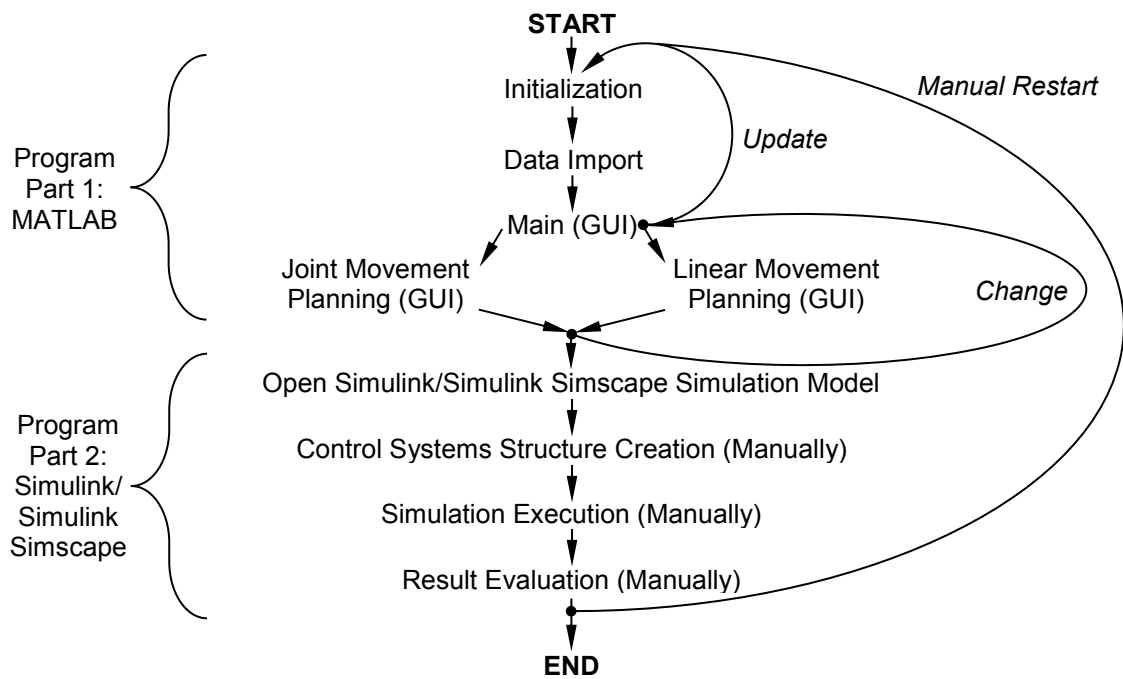


FIGURE 5.11: General Simulation Program Flow Chart

Based on the previously determined main tasks of the MATLAB program part:

- Simulation model parameterization
- Set values calculation and provision

And the main contents and functionalities obtainable from FIGURE 5.2 and FIGURE 5.11, appropriate conceptual designs were elaborated and are presented and discussed within the subsequent sections 5.3.1, 5.3.2 and 5.3.3.

### 5.3.1 Simulation Model Variables

In the context of the previously mentioned automatically generated Simulink Simscape Multibody simulation model from the CAD model, using the MATLAB `smimport()` function, the “ABB\_IRB\_2600\_12\_185\_Simscape\_Data.m” file was automatically created. The corresponding and predetermined MATLAB variable of this file is `smiData`.

Furthermore, the predetermined variables `robotModel` and `importInfo` are automatically created from the MATLAB `importrobot()` function, required for inverse kinematics solving (see section 5.3.3).



As it is highly recommended not to change these predetermined, required and automatically generated variables:

- `smiData`
- `robotModel`
- `importInfo`

The decision was made to create new own variables for the MATLAB program part. Therefore, the variable:

- `robotPara`

Shall be used to store the simulation model parameters in SI and/ or derived SI units, imported from the externally provided robot parameters compilation. The creation/ initialization of the variable and the subsequent storing of the parameters shall be accomplished within a separate MATLAB function.

The variable:

- `simVar`

Shall be created in order to act as the main working variable of the MATLAB program part. In detail, the variable shall be passed into and returned from each individual MATLAB function for data transfer. Hence, the variable shall store all required information captured and calculated during the MATLAB program execution, which finally also covers the trajectory set values to be provided to the Simulink/ Simulink Simscape environment. The variable shall also be created by a separate MATLAB function to ensure the comprehensibility of its structure and contents.

Both variables shall be stored at the MATLAB “base” Workspace in order to enable visibility for the user and easy access for the Simulink/ Simulink Simscape environment.

### 5.3.2 Graphical User Interfaces

Referring to the requirements of the simulation model, user inputs/ interactions are at minimum to be performed via the MATLAB Command Window. Due the number and extent of all required individual elements of the MATLAB program part (see subsequent section 5.3.3), controlling the simulation program/ model via the MATLAB Command Windows was considered as inconvenient, inefficient, complex and may require additional syntax/ command knowledge. Based on that, the decision was made that the optional task of the implementation of graphical user interfaces (GUI) shall be accomplished.

Therefore, the graphical interfaces shall be created with the MATLAB graphical user interface development environment (GUIDE) tool, whereby, according to FIGURE 5.11, three GUI shall be created:

- Main GUI
- Joint Movement Planning GUI
- Linear Movement Planning GUI

Each GUI in turn is represented by a `.fig` file which represents the contents and appearance of the corresponding GUI window itself and a corresponding `.m` file. The `.m` file contains the required code for the implementation of the GUI functionalities (which covers the initial GUI parametrization (e.g. labelling of text boxes), callback functions (reactions on user interactions, e.g. user presses a button) and additional sub functions if required). Thus, six `.m/ .fig` files are required for the implementation of the GUI of the MATLAB program part.

Moreover, each GUI shall contain sufficient input filtering such as:

- Input value type check (number, text, etc.)
- Input value limitations/ range check (e.g. axes angular limitations)

For each individual input value of a GUI and an:

- Input completeness check (entirety of required input values)

In the context of all input values of the corresponding GUI.

Additionally, each GUI shall contain required descriptions and/ or descriptive images to provide guidance and comprehensibility.

A sketch of the conceptualized appearance of the Main (G)UI window is shown in FIGURE 5.12 below.

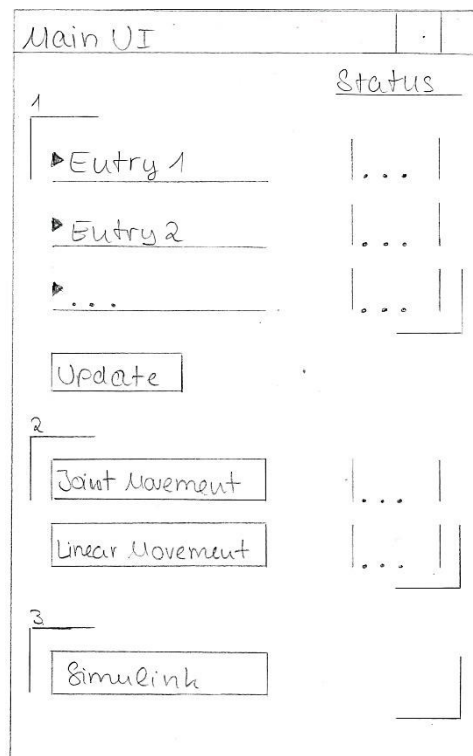


FIGURE 5.12: Freehand sketch of the concept of the Main (G)UI window

A sketch of the conceptualized appearance of the Motion (G)UI windows is shown in FIGURE 5.13 below, whereby the concept shall be valid for both (linear and joint movement) and therefore need to be adapted to the specific requirements (e.g. corresponding description/ descriptive image, number and format of input values, etc.).

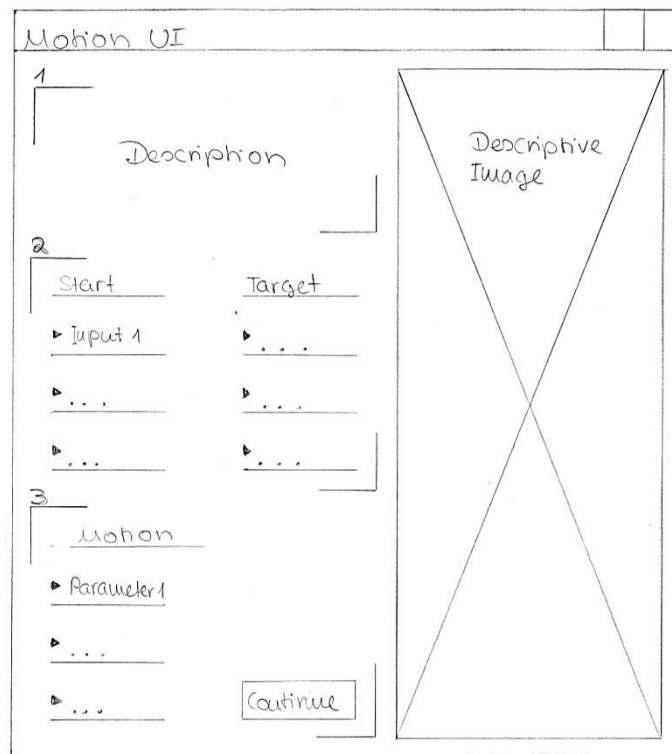


FIGURE 5.13: Freehand sketch of the concept of the Motion (G)UI window

Further preconfigured (graphical elements), e.g. MATLAB dialog boxes, shall be used to display further guidance, information, warnings and errors (e.g. in case of the application of non-appropriate input values).

### 5.3.3 Programs & Program Flow Charts

For the purposes of modularity and comprehensibility, the decision was made that each main task/ procedure of the MATLAB program shall be represented by an individual MATLAB `.m` function file.

Example: `simVar_init.m` shall only contain the `simVar_init()` function.

Some `.m` function files may contain further sub functions which shall only be called locally (within the corresponding function).

Summarizing the contents described in the previous sections 5.2.3, 5.3.1 and 5.3.2, the conceptual design determined the need of four functions for the creation of variables, six functions for the GUI representations and one function for the import of robot parameters.

For the purpose of the installation of a required MATLAB Simulink Simscape Library (see FIGURE 5.2) another separate function shall be created.

Furthermore, the conceptual design of the robotic manipulator's motion reference trajectory design was pending. Therefore, the conceptualization of a linear robotic movement planning was depicted in a flow chart diagram (see FIGURE 5.14) and described below.

The conceptualization of a linear robotic movement planning is presented exemplarily and also valid for the joint movement planning but without the need of solving the inverse kinematics.

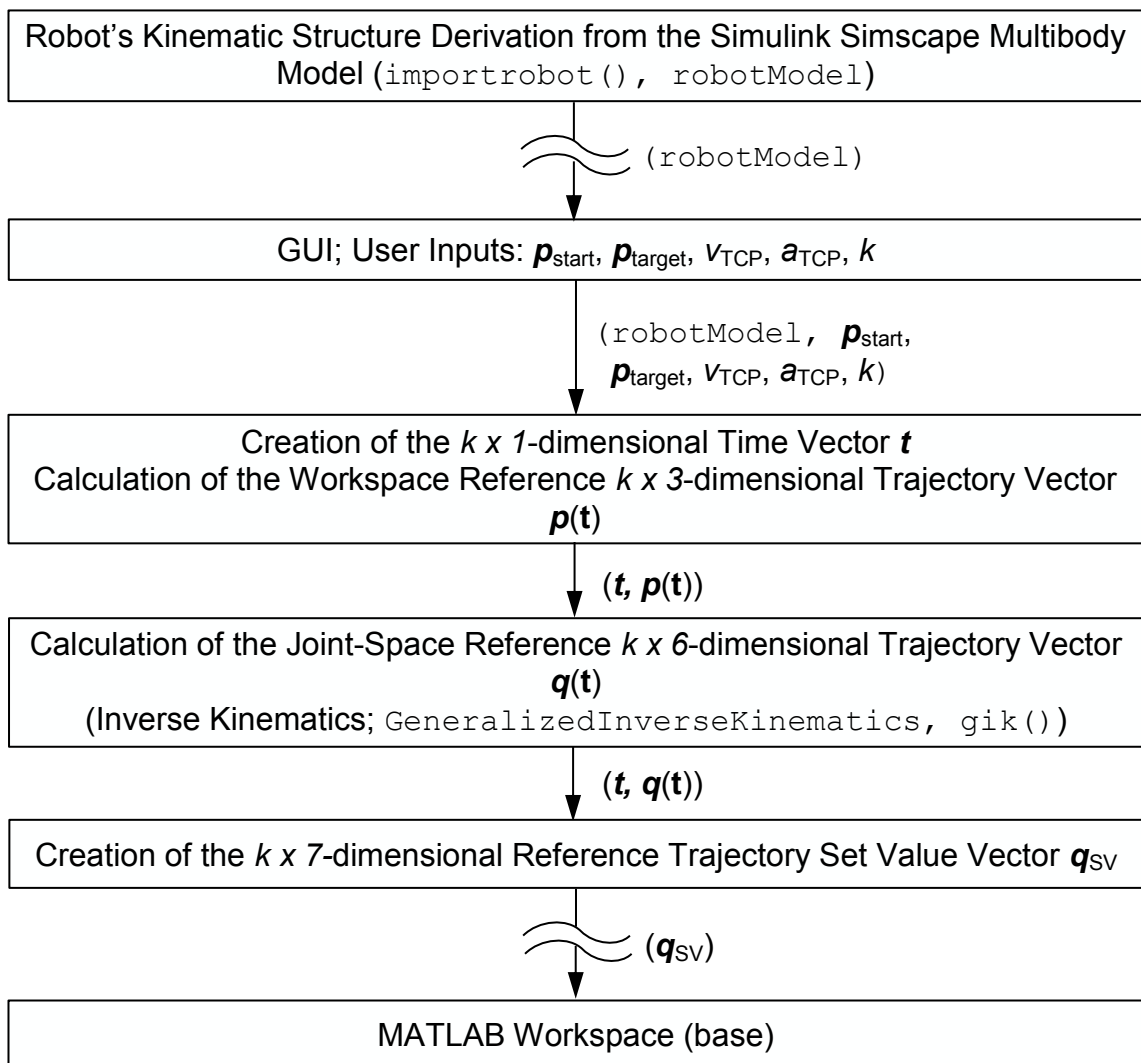


FIGURE 5.14: Flow chart diagram of the linear movement planning

Firstly, the robot's kinematic structure, represented by a MATLAB `RigidBodyTree` object `robotModel`, needs to be derived from the existing Simulink Simscape Multibody model with the MATLAB `importrobot()` function. Following this, the desired starting (A) workspace position  $\mathbf{p}_{\text{start}} = [x_A \ y_A \ z_A]$ , workspace target (B) position  $\mathbf{p}_{\text{target}} = [x_B \ y_B \ z_B]$  and the corresponding motion parameters (velocity  $v_{\text{TCP}}$ , acceleration  $a_{\text{TCP}}$  and interpolation resolution  $k$ ) shall be obtained from the user via the GUI (see FIGURE 5.13)

Subsequently, the workspace trajectory  $\mathbf{p}(\mathbf{t})$  vector shall be calculated from the user inputs within a separate function and accordingly to the elaborated theory (section 4.2.1). It must be considered that the theory of the calculation of the time dependent workspace trajectory vector  $\mathbf{p}(\mathbf{t})$  is formulated analytically (= continuous time). As the simulation model bases on computational calculations, firstly a discrete time series (=  $k \times 1$ -dimensional time vector  $\mathbf{t}$ ) shall be determined and then used to calculate discrete values of the  $k \times 3$ -dimensional workspace trajectory vector  $\mathbf{p}(\mathbf{t})$ .

Following this, a separate function shall be created for inverse kinematics solving in order to calculate the required  $k \times 6$ -dimensional reference joint-space trajectory  $\mathbf{q}(\mathbf{t})$  vector from the workspace trajectory vector  $\mathbf{p}(\mathbf{t})$ . Therefore the MATLAB `GeneralizedInverseKinematics` solver shall be used. Its corresponding MATLAB function `gik()` shall be called with the `robotModel` object and appropriate `ConstraintInputs` objects, to be determined in advance and accordingly to the requirements (e.g. tool orientation).

Then, the reference joint-space trajectory  $\mathbf{q}(\mathbf{t})$  vector shall be united with the time vector  $\mathbf{t}$  in order to create a  $k \times 7$ -dimensional reference trajectory set value vector  $\mathbf{q}_{\text{SV}}$ . This vector shall be stored at the MATLAB Workspace "base" to be accessible for the Simulink/ Simulink Simscape program part and act as control system set values.

Exemplarily, the program flow chart (PFC) of the inverse kinematics function `inverse_kinematics()` is shown in FIGURE 5.15 below.

inverse\_kinematics(simVar,robotModel,robotPara)

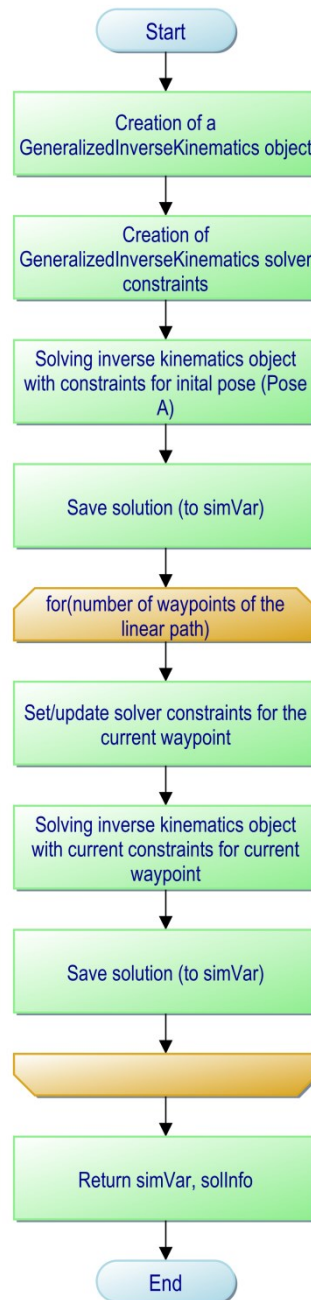


FIGURE 5.15: Program flow chart of the inverse\_kinematics() function

The procedure for the joint trajectory planning shall be same but without the inverse kinematics as the reference joint-space trajectory can be calculated directly from the user inputs.

Finally, a main function shall be created to represent the general program flow, depicted in and obtainable from page 1(18) of Appendix 4. Program Flow Charts.

In order to provide a convenient start of the simulation program (without typing calling parameters) by the user from the MATALB Command Window, the main function shall in turn be called by another superordinate but concise function.

Summarized, the programs to be created, listed and described in TABLE 5.1 below, shall be designed during the accomplishment considering the elaborated definitions and regulations, theory and conceptual design.

TABLE 5.1: Listing and description of the conceptualized MATLAB function(s) (files)

MATLAB .m/ .fig (Function) File(s):	Description:	Corresp. PFC (Appendix 4. Program Flow Charts) Page:
<code>runSim.m</code>	Entry point of the simulation; shall call the <code>main()</code> function and shall be called with <code>runSim</code> ; from the MATALB Command Window.	16(18)
<code>init.m</code>	Shall call all required initialization functions ((e.g. <code>simVar_init()</code> ) and/ or initialize/ install all required variables/ libraries/ data.	4(18)
<code>simVar_init.m</code>	Shall initialize the <code>simVar</code> variable.	17(18)
<code>load_smiData.m</code>	Shall load the <code>smiData</code> variable from "ABB_IRB_2600_12_185_Simscape_DataFile.m".	10(18)
<code>multi_physics_library.m</code>	Shall install the required Simscape Multibody Multiphysics Library.	13(18)
<code>robot_import.m</code>	Shall import and/ or update the robotic system/ structure ( <code>RigidBodyTree</code> ) from the Simulink Simscape model (.slx).	14(18)



TABLE 5.1: Listing and description of the conceptualized MATLAB function(s) (files)

MATLAB .m/ .fig (Function) File(s):	Description:	Corresp. PFC (Appendix 4. Program Flow Charts) Page:
ro- bot_para_xls _import.m	Shall create the <code>robotPara</code> variable and import simulation model parameters from the parameter spreadsheet.	15(18)
main.m	Main function (and corresponding GUI); shall represent the general program flow. Shall call all other necessary functions in order to gather user inputs and to calculate and provide the required data for the Simulink simulation.	11(18)
main_ui.m		12(18)
main_ui.fig		
joint_move_u i.m	Shall obtain and filter required user inputs for the trajectory planning of a joint movement of the robotic manipulator (includes the corresponding GUI).	6(18)
joint_move_u i.fig		
joint_traj_p lanning.m	Shall calculate the joint space trajectory of a joint movement with a $\sin^2$ acceleration profile in full synchronous mode	7(18)
lin_path_ui. m	Shall obtain and filter required user inputs for the trajectory planning of a linear movement of the robotic manipulator (includes the corresponding GUI).	8(18)
lin_path_ui. fig		
lin_traj_pla nning.m	Shall calculate the workspace trajectory of a linear movement with a $\sin^2$ acceleration profile.	9(18)
in- verse_kinema tics.m	Shall solve the inverse kinematics of the robotic manipulator for each waypoint of a linear movement ( $\mathbf{p}(t) \rightarrow \mathbf{q}(t)$ ) based on Cartesian start and target user inputs using the MATLAB <code>gik()</code> solver function.	6(18)

## 6 ACCOMPLISHMENT

The accomplishment generally based on the basic ideas, structures, methods and solutions elaborated in the conceptual design (section 5). Under continuous consideration of the definitions and regulations (section 3) and, if required, the methods and knowledge gained from the theory (section 4), the accomplishment was conducted. Adaptions were made whenever necessary in order to meet the requirements (Appendix 3. List of Requirements) satisfyingly and to increase the simulation model's quality. Adaptions as well as substantial deviations from the individual corresponding conceptual design are mentioned accordingly.

Additional required unspecific/ general information, knowledge and help/ guidance were obtained from The MathWorks Inc. (2019b), The MathWorks Inc. (2019c), The MathWorks Inc. (2019d) and Glöckler (2018) during the MATLAB and Simulink/ Simulink Simscape programming.

Due to the repetitive character of the most of the tasks processed during the accomplishment procedure (mainly caused by the repetitive structure of the robotic manipulator's simulation model), accomplishments are primarily presented and documented in an exemplary manner within this section.

### 6.1 CAD Model

Used CAD software: Dassault Systèmes® SolidWorks™<sup>11</sup> Premium 2014, x64-Edition, SP 2.0

Additional Software: MathWorks Simscape Multibody Link Version 6, Release: R2018b, Win64, plug-in for SolidWorks 2001Plus and higher

The first step of the creation of the simulation model was related to the preparation of the manipulator's CAD model. Therefore, the CAD data of the robot links and the end effector were separated from the `.x_t` parasolid file of the robot welding cell provided by the client (TAMK).

---

<sup>11</sup> SOLIDWORKS™ is a trademark of Dassault Systèmes®

Following this, the separated data (see TABLE 6.1) were edited individually in order to define the material, density, weight and colouration of each part. Subsequently, the parts were assembled using appropriate constraints (see TABLE 6.1) to create a fully functional representation of the real robotic system (see FIGURE 6.2). The last step covered the generation of Simscape Multibody environment import files using the MathWorks Simscape Multibody Link plug-in for SolidWorks (see TABLE 6.2). The obtained import files were needed to create the manipulator's Simscape Multibody simulation model automatically, using the MATLAB `smimport()` function. The previously described procedure is also depicted in FIGURE 6.1 below.

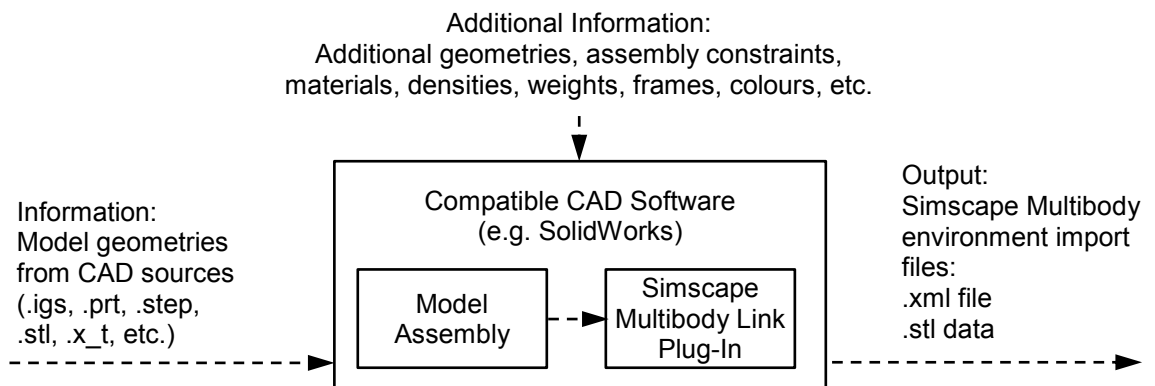


FIGURE 6.1: Procedure of the generation of the Simscape Multibody model import files

TABLE 6.1: List of SolidWorks parts and assemblies of the manipulator

SolidWorks Parts			
No.:	Name:	Type:	Assembly Constraint Type Link $i+1$ to Link $i$ :
0	IRB2600_12_185_base	SolidWorks Part Document (.sldprt)	Revolute (Coincidence of the sur- faces and axes of the CAD model representing the revolute joints) (see FIGURE 6.2)
1	IRB2600_12_185_link1		
2	IRB2600_12_185_link2		
3	IRB2600_12_185_link3		
4	IRB2600_12_185_link4		
5	IRB2600_12_185_link5		Rigid
6	IRB2600_12_185_link6		-
7	Welding_End_Effector		
SolidWorks Assembly			
Name:		Type:	
ABB_IRB_2600_12_185_Simscape		SolidWorks Assembly Document(.sldasm)	

TABLE 6.2: List of Simscape Multibody simulation model input files of the manipulator

Simscape Multibody Simulation Model Files: .stl		
No.:	Name:	Type:
0	IRB2600_12_185_base_Standard_sldprt	.stl
1	IRB2600_12_185_link1_Standard_sldprt	
2	IRB2600_12_185_link2_Standard_sldprt	
3	IRB2600_12_185_link3_Standard_sldprt	
4	IRB2600_12_185_link4_Standard_sldprt	
5	IRB2600_12_185_link5_Standard_sldprt	
6	IRB2600_12_185_link6_Standard_sldprt	
7	Welding_End_Effector_Standard_sldprt	
Simscape Multibody Simulation Model Files: .xml		
-	ABB_IRB_2600_12_185_Simscape	.xml

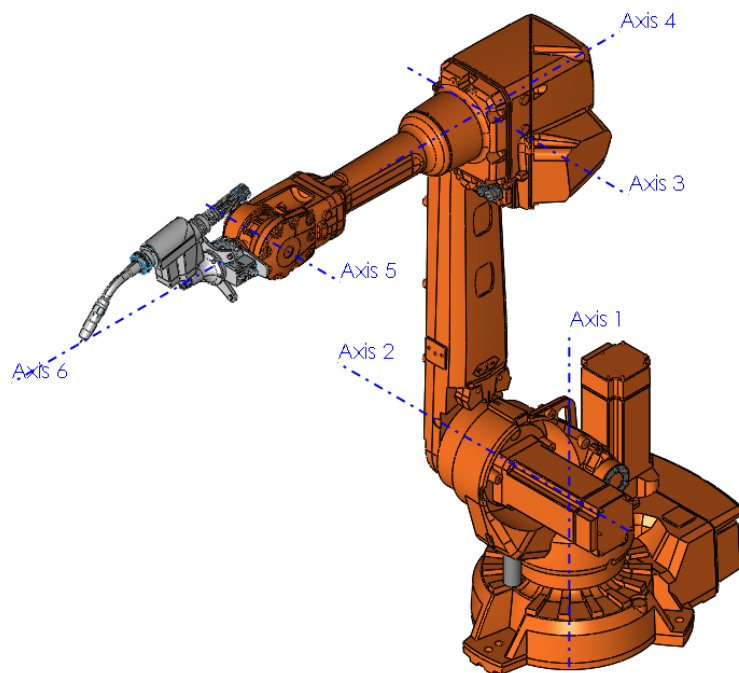


FIGURE 6.2: Screen capture of the SolidWorks assembly of the robot manipulator

## 6.2 Simulink Simulation Model

In contrary to the chronology of the process of the accomplishment of the thesis work, the final general structure of the simulation model is already shown at the beginning of this section. The idea behind is, to firstly give an overview over the final simulation model's Simulink block diagrams individual elements (see FIGURE 6.3), which are then presented and described in detail within the following subsections 6.2.1 - 6.2.4.

The final Simulink/ Simulink Simscape simulation model file "ABB\_IRB\_2600\_12\_185\_Simscape.slx" can be found from the "Simulink Simscape Data" folder of the complete data set (see section 6.5), along with all other required corresponding data (e.g. .stl files).

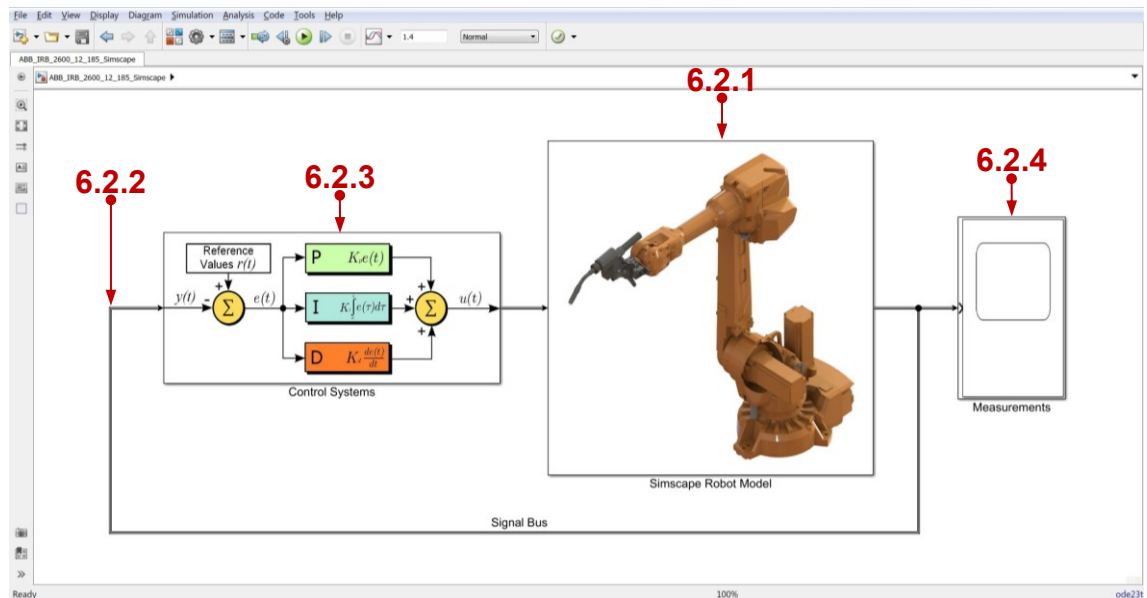


FIGURE 6.3: Screen capture of the final simulation model's Simulink block diagram

### 6.2.1 Simulink Simscape Multibody Robot Model

As conceptualized, the basic Simscape Multibody model of the robotic system was generated automatically from the pre-processed CAD data, more precise the "ABB\_IRB\_2600\_12\_185\_Simscape.xml" file (section 6.1, TABLE 6.2), using the MATLAB `smimport()` function.

The basic Simscape Multibody model (see section 5, FIGURE 5.1) was then elaborated as conceptualized (adding further subsystems, joint actuation, signalling etc.), see FIGURE 6.4 below, and implemented as a subsystem of the overlaying Simulink block diagram (see mark 6.2.1 in FIGURE 6.3).

All corresponding data (block parameters) of the automatically generated Simscape Multibody model are stored in the also automatically generated “ABB\_IRB\_2600\_12\_185\_Simscape\_DataFile.m” file.

The Simscape Multibody robot model itself (FIGURE 6.4) consists of the first (sub) subsystem (Robot Base Subsystem) rigidly linked to the World Frame and all other (sub) subsystems which are sequentially rigidly linked to their predecessors. All (sub) subsystems do have the same general structure and are connected to the signal bus (except the Robot End Effector Subsystem).

The World Frame, Mechanism Configuration and Solver Configuration blocks contain the basic parametrization and references of the Simscape simulation environment (e.g. value and direction of the gravitational acceleration) and are a basic requirement for every Simulink Simscape model.

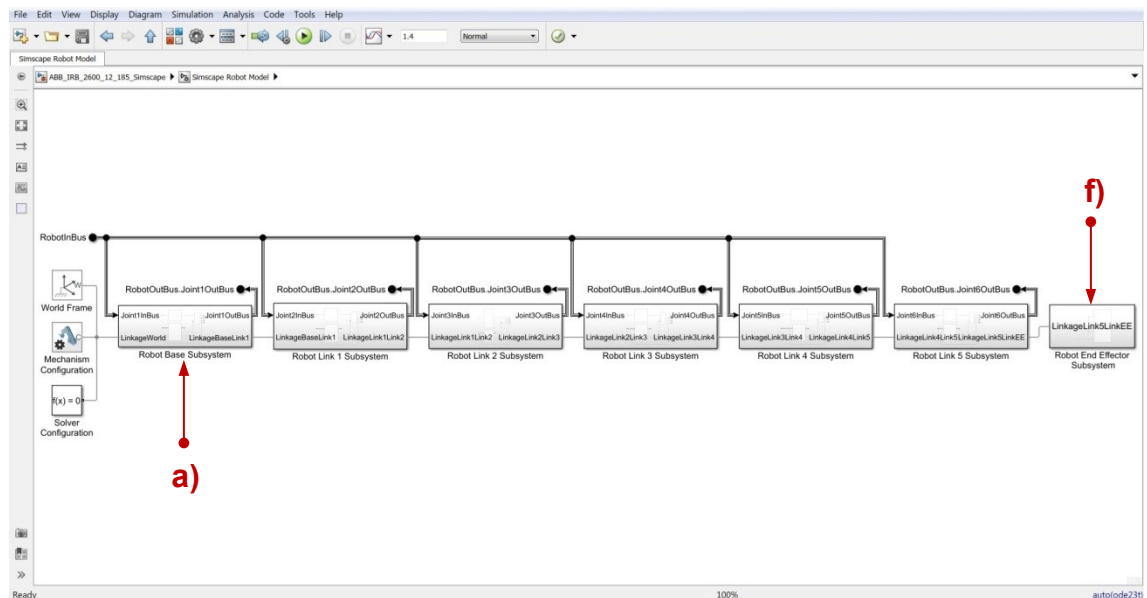


FIGURE 6.4: Screen capture of the final simulation model’s Simulink Simscape robot model

Exemplarily for the other sub (subsystems), FIGURE 6.5 shows the Robot Base (sub) Subsystem of the Simscape robot model (see mark a) FIGURE 6.4).

The Base Body (sub) subsystem of the Robot Base Subsystem contains a Solid block representing the rigid body of the base (link 0) and Rigid Transform blocks for the required appropriate fixed frame transformations (kinematic structure) as automatically generated. The Base Body (sub) subsystem is rigidly connected to its predecessor (World Frame in this case (LinkageWorld)) and to the corresponding joint (Joint 1).

The Joint 1 block (Revolute Joint) represents the real revolute joint of the robot's system and is rigidly connected to the Base Body (sub) subsystem on the input side and to the corresponding kinematic successor (Link 1) on the output side (LinkageBaseLink1). Deviating from the conceptual design, no additional sensor elements/ blocks were applied to measure the joint variables  $q_1$ ,  $\dot{q}_1$ ,  $\ddot{q}_1$  and  $\tau_1$  from the corresponding joint as the block provides these values via internal sensing.

Also deviating from the conceptual design, the Joint 1 Drive System (sub) subsystem does not only apply the motor torque to the Joint 1 block (port "t") but also receives joint velocity feedback ( $\dot{q}_1$ ) (port "w") for interfacing purposes (see description of FIGURE 6.6). Both, the Joint 1 Drive System (sub) subsystem (mark b) FIGURE 6.5) and the Joint 1 block are connected to the signal bus. Signals from different signal domains (e.g. Simscape Multibody and Simulink domain) are converted with PS-Simulink Converter or Simulink-PS Converter blocks before added to or branched off the signal bus.

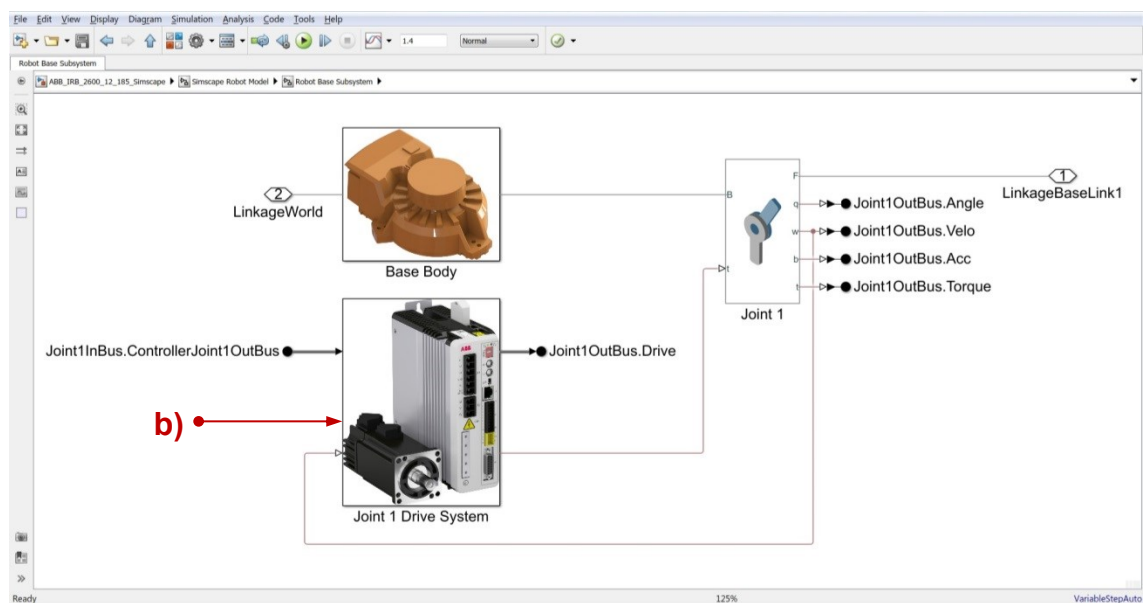


FIGURE 6.5: Screen capture of the Simscape simulation model's Robot Base Subsystem

The block diagram of the (sub) subsystem of the Joint 1 Drive System (mark b) FIGURE 6.5) is shown in FIGURE 6.6 below.

As conceptualized, the ASM 1 Driver (sub) subsystem (mark c) FIGURE 6.6) receives the corresponding controlled value from the Controller Joint 1 subsystem (via the signal bus) and drives the joint actuating ASM 1 block using the three-phase voltage supply (blue coloured electrical domain).

The ASM 1 block represents the real AC asynchronous joint motor and is driven in delta configuration using a Phase Permute (Delta) block in order to gain maximum motor torque. The joint actuation (ASM 1) block provides its (mechanical) torque via the rotational conserving ports “R” (Rod) and “C” (Case) to the Rotational Simscape Interface 1, using the Simscape mechanical rotational domain network (green coloured domain). The joint actuation (ASM 1) torque cannot directly be applied to the corresponding joint as the joint actuation is performed within the Simscape Multibody domain (red coloured signalling domain) and therefore interfacing with joint velocity feedback ( $\dot{q}_1$ ) is required.

The Machine 1 Inertia block and Machine 1 Viscous Damping block were added to the mechanical rotational network to simulate the mechanical characteristics of the joint actuation motor not covered by the corresponding block (ASM 1). The real gearbox of the robotic system is represented by the Cycloidal Transm.1 block also implemented in the mechanical rotational network. As the Cycloidal Transm.1 block does not provide the inertia parameter of the gearbox to be simulated, the Transm. 1 Inertia block was applied.

The Joint 1 Bearing Friction block represents the (linear) friction model of the real bearing of the corresponding joint (Joint 1) and covers breakaway friction as well as Coulomb and viscous friction. Block parameterization is described in the later sections 6.4.2 and 6.4.3.



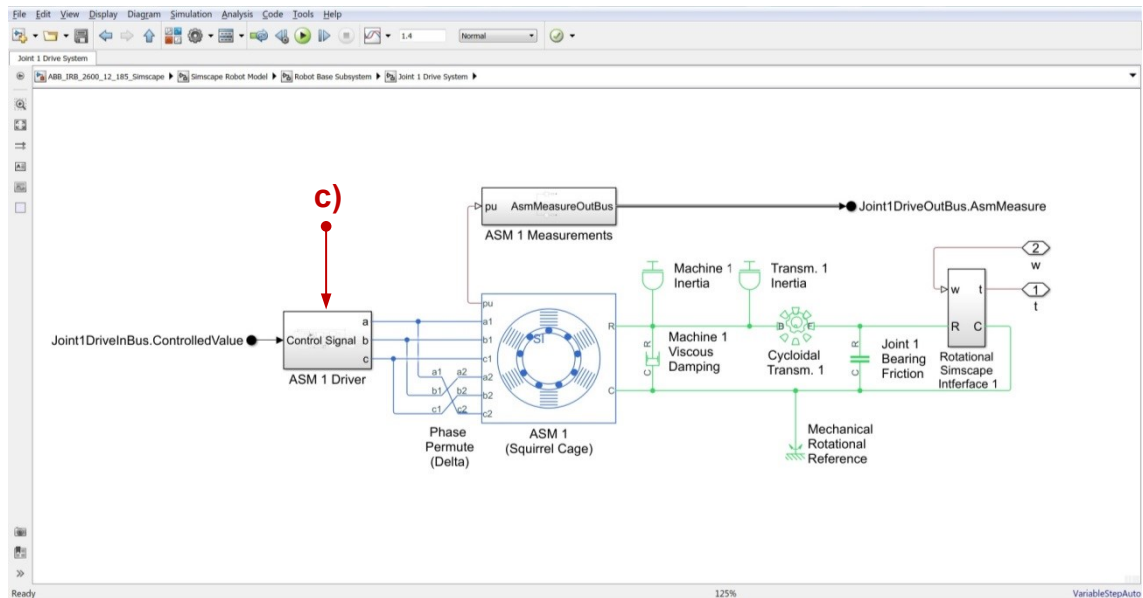


FIGURE 6.6: Screen capture of the Simscape simulation model's Joint 1 Drive Subsystem (1/2)

The block diagram of the ASM 1 Driver (sub) subsystem (mark c) FIGURE 6.6) is shown in FIGURE 6.7 below.

According to the conceptual design (section 5.2.1), the ASM 1 Driver (sub) subsystem was created utilizing the basic function principles and blocks of the MATLAB Simulink Simscape “Asynchronous Machine Scalar Control” (`pe_asm_scalar_control`) example block diagram. During the creation, several adaptations were applied whereby the signal limitation and the rotational direction reversing are the most considerable. The signal limitation was implemented to narrow the expected scalar input value range to -1 to +1, using the Signal Limiter block (Saturation) (mark d) FIGURE 6.7) (Thus: “+1” = “100% power in the positive direction” and “-1” = “100% power in the negative direction”). The Rotational Direction Reverser (sub) subsystem (mark e) FIGURE 6.7) determines the rotational direction of the corresponding joint actuation motor (ASM 1) by permuting two of the three voltage supply phases, depending on the sign of the controlled value (Control Signal).

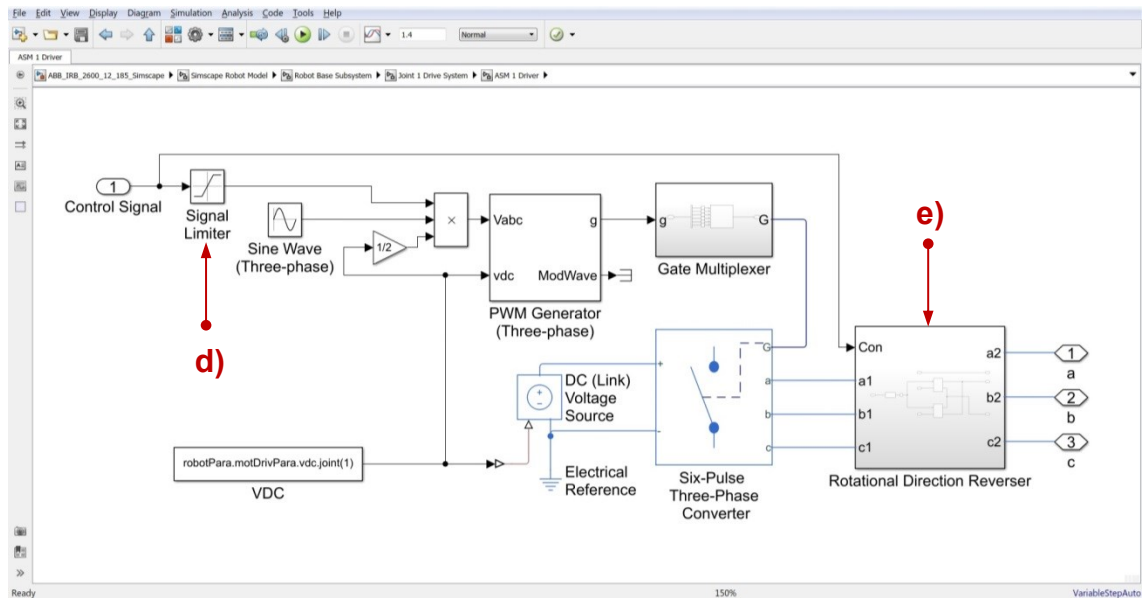


FIGURE 6.7: Screen capture of the Simscape simulation model's ASM1 Driver Subsystem

As the Robot End Effector Subsystem (mark f) FIGURE 6.4) differs from the other (sub) subsystems, its block diagram is shown in FIGURE 6.8 below. Accordant to the conceptual design, it only contains the rigidly connected rigid body (sub) subsystems of link 6 (Link 6 Body) and the end effector/ tool (End Effector Body) in order to represent the robot manipulator's last kinematic element.

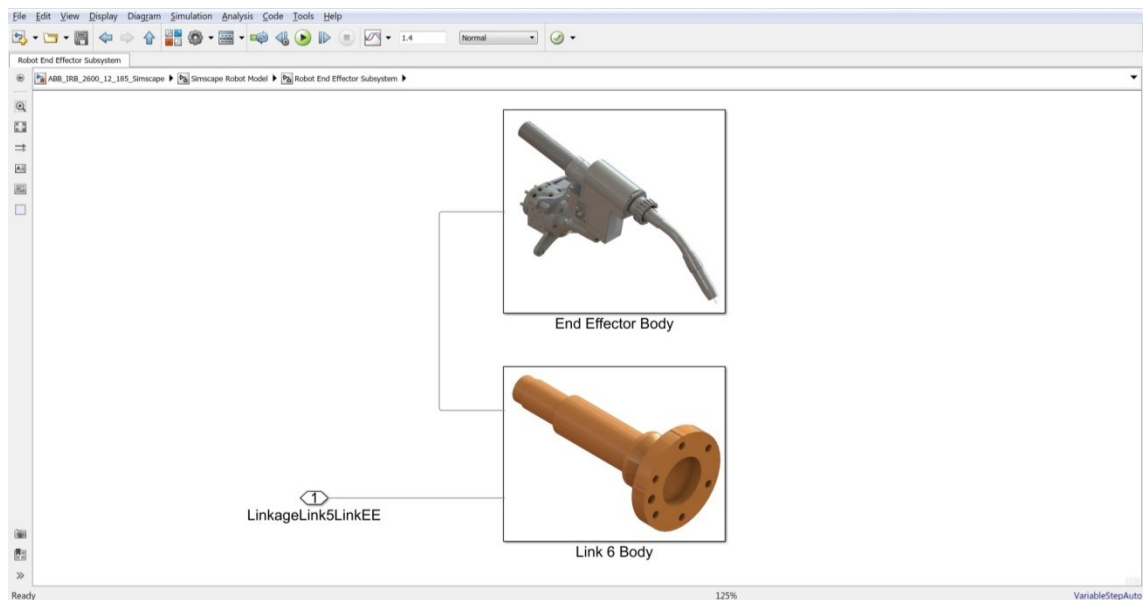


FIGURE 6.8: Screen capture of the Simscape simulation model's Robot End Effector Subsystem

## 6.2.2 Signal Bus

Within the Simulink/ Simulink Simscape simulation model, all signals are routed with the help of a signal bus structure (see mark 6.2.2 in FIGURE 6.3) (exception: values/ parameters directly or indirectly obtained from the MATLAB Workspace (base) using From Workspace blocks). Each (sub) subsystem connected to the signal bus contains InBus blocks (mark g)) to branch off individual signals from the signal bus and OutBus blocks (mark h)) to add signals to the signal bus as exemplarily shown for the Joint 1 Drive System (sub) subsystem in FIGURE 6.9 below.

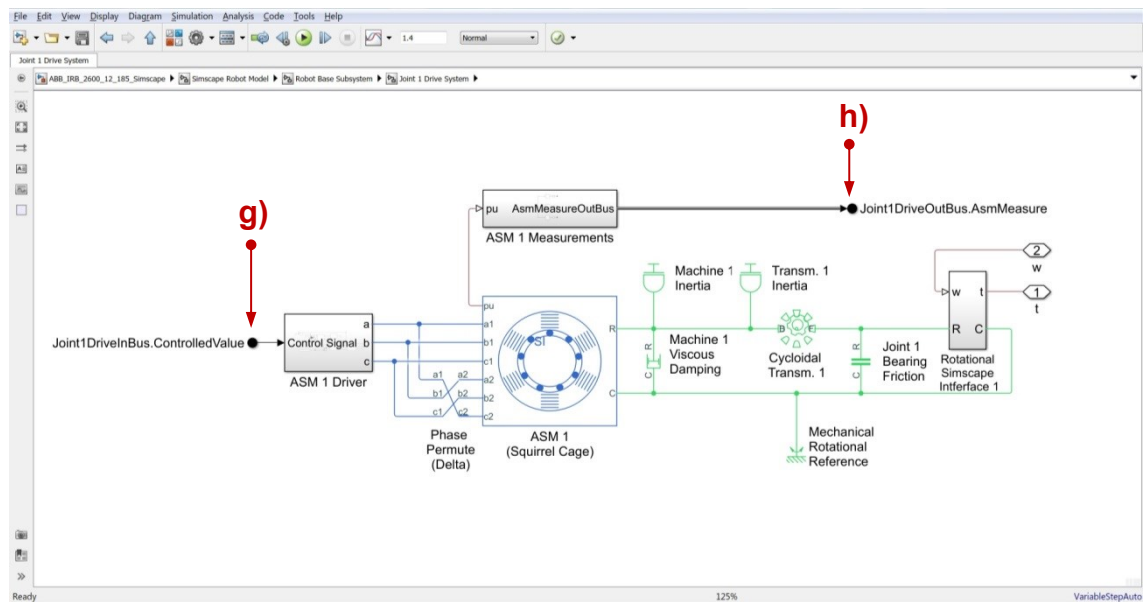


FIGURE 6.9: Screen capture of the Simscape simulation model's Joint 1 Drive Subsystem (2/2)

Sub busses were created within the signal bus in order to bundle individual but related signals meaningfully within a number of subordinated sub busses. FIGURE 6.10 exemplarily shows the structure of the InBus (ControllerSystem-InBus) of the Control System subsystem (mark 6.2.3 in FIGURE 6.3). The incoming individual (joint and motor) signals of each Simscape model's (sub) subsystem (e.g. Robot Base Subsystem) Angle, Velo, Acc, Torque, Drive are bundled within the corresponding sub busses Joint1OutBus – Joint6OutBus.

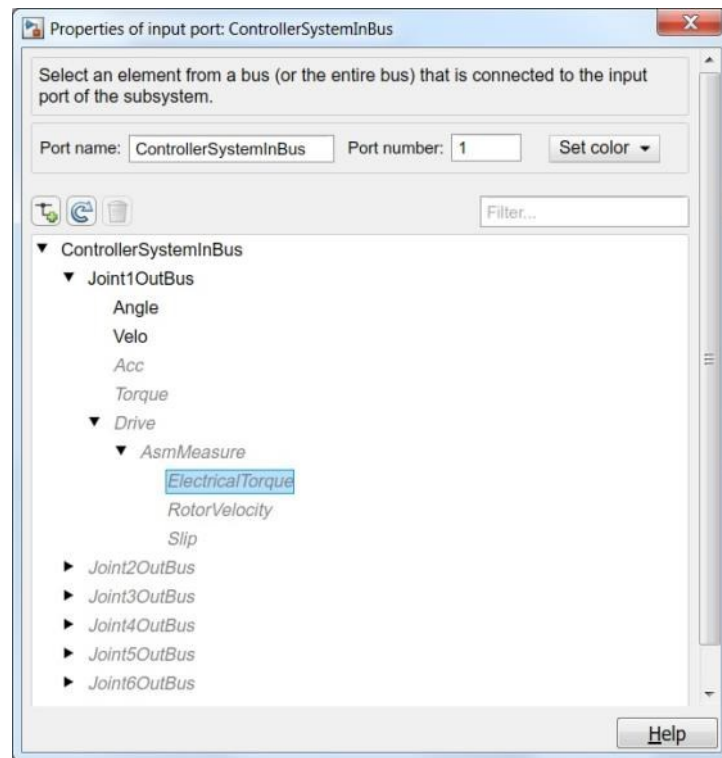


FIGURE 6.10: Screen capture of the signal bus structure of the Control System subsystem In-Bus

The naming of the individual bus signals and sub busses was applied in a consistent and explanatory manner; therefore, no overall listing of the bus systems individual signals is given here.

Alternatively, all signal types measured and available from the signal bus of the final Simulink/ Simscape simulation model are listed in TABLE 6.3 below.

TABLE 6.3: Listing of the available signal bus signal types

Signal Description:	Related Symbol(s):
Joint Position Variables/ Values	$q_1 \dots q_6$
Joint Velocity Variables/ Values	$\dot{q}_1 \dots \dot{q}_6$
Joint Acceleration Variables/ Values	$\ddot{q}_1 \dots \ddot{q}_6$
Joint Torque Variables/ Values	$\tau_1 \dots \tau_6$
Controlled Variables/ Values	$u_1 \dots u_6$
Motor (ASM) Electrical Torque Variables/ Values	$\tau_{1,e} \dots \tau_{6,e}$
Motor (ASM) Rotor Velocity Variables/ Values	$\omega_{1,m} \dots \omega_{6,m}$
Motor (ASM) Slip Variables/ Values	$s_1 \dots s_6$

As already mentioned in section 6.2.1, PS-Simulink Converter or Simulink-PS Converter blocks were implemented for interfacing when adding or branching off bus signals if required, as the signal bus only transfers Simulink domain signals. For error prevention purposes, only SI unit and derived SI unit signal values are transferred within the bus (except signals originally without any unit). Therefore, unit conversion blocks were added wherever required (e.g. within the Measurement System).

### 6.2.3 Control System Structures

Accordant to the conceptual design (section 5.2.2), a superior control system structure, the Control Systems subsystem, was implemented as a subsystem of the overlaying Simulink block diagram (see mark 6.2.3 in FIGURE 6.3).

The Control System's subsystem carries six (individual) decentralized control system structures (sub) subsystems (Controller Joint 1 (Axis 1) – Controller Joint 6 (Axis 6)), one for each of the simulation model's (revolute) joints, which are connected to the signal bus and shown in the subsequent FIGURE 6.11. The subsystem can be used as an environment for centralized control if required.

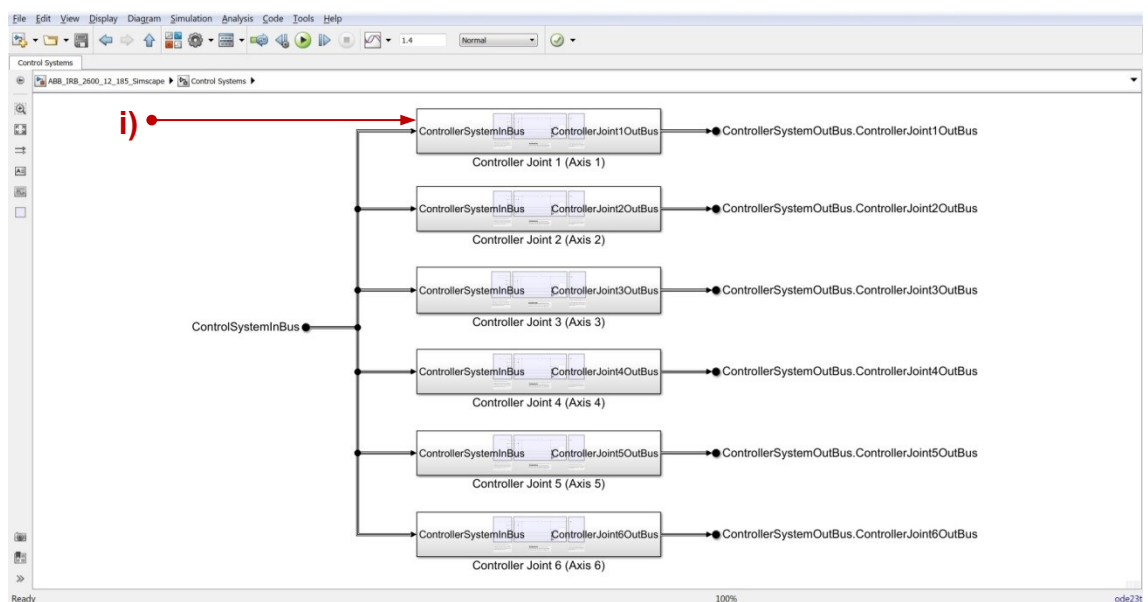


FIGURE 6.11: Screen capture of the Control System subsystem of the Simulink block diagram

Apart from the deviating input and output signals of the individual Controller Joint (sub) subsystems, all (sub) subsystems do have the same structure, exemplarily shown for the Controller Joint 1 (Axis 1) (sub) subsystem (mark i) FIGURE 6.11) in FIGURE 6.12 below.

Each Controller Joint (sub) subsystem is divided into three areas (blue shaded areas, FIGURE 6.12), left: input area (mark j)), middle: controller area (mark k)) and right: output area (mark l)), whereby applied block diagrams must not be necessarily kept inside the areas, since the separation is only meant as a suggestion for the purpose of a clear structure. Explanatory notes are given below each area.

The input area provides the corresponding joint-space position set values (reference trajectory) using a From Workspace block in order to obtain the reference values ( $q_{1SV}$ ) from the `simVar` variable from the MATLAB "base" Workspace. Furthermore, the measured actual values of the corresponding Joint block, joint position  $q_1$  and joint velocity  $\dot{q}_1$  are branched of the signal bus.

The controller area is initially equipped with a simple closed-loop controller structure and a Scope block. The predefined PID Controller block as well as the Scope block were implemented for testing purposes only and do not provide an appropriate control system structure of the simulation model.

The output area contains the signal of the controlled value of the corresponding joint actuation (sub) subsystem (ASM 1 Driver) added to the signal bus.

As each Controller Joint (sub) subsystem is connected to the signal bus, all signals of the bus structure listed in the previous TABLE 6.3 are available within each (sub) subsystem. Following this and deviating from the conceptual design, the implemented Controller Joint (sub) subsystems are MISO instead of MIMO structures, as only one corresponding output (controlled value;  $u_1 \dots u_6$ ) is available for each (sub) subsystem at the contemporary state. This decision was made based on the lack of information of the types and amount of the specific controlled values to be implemented by the individual user and application.

Due to the usage of a signal bus, further signals can be branched off and/ or added to the signal bus and therefore lead out from and/ or added to the Controller Joint (sub) subsystems by the user whenever required.

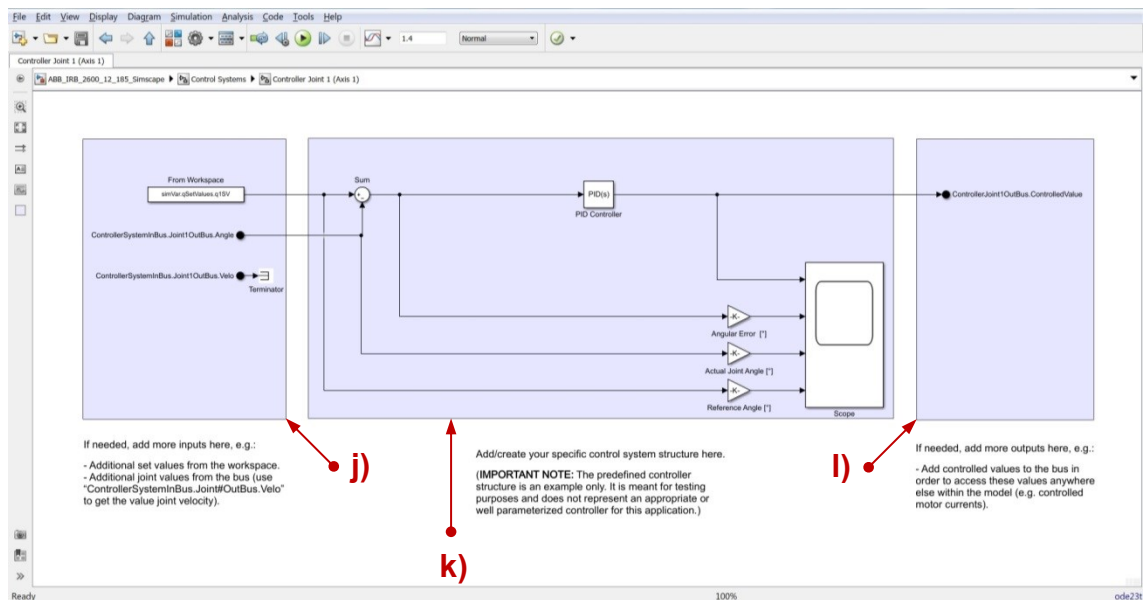


FIGURE 6.12: Screen capture of the Controller Joint 1 (Axis 1) (sub) subsystem

## 6.2.4 Measurements

In contrary to the conceptual design (section 5.2.2), the measurements subsystems were outsourced from the individual subsystems of the Simulink Simscape Multibody model and implemented bundled as a subsystem of the overlaying Simulink block diagram (see mark 6.2.4 in FIGURE 6.3). As all measured signals are available from the signal bus anyways, no additional structures were applied in the context of the implementation of the Measurements subsystem.

The Measurements subsystem consists of the four (sub) subsystems:

- Joint Angles ( $q_1 \dots q_6$ )
- Joint Velocities ( $\dot{q}_1 \dots \dot{q}_6$ )
- Joint Accelerations ( $\ddot{q}_1 \dots \ddot{q}_6$ )
- Joint Torques ( $\tau_1 \dots \tau_6$ )

Whereby the (sub) subsystems are organized by the type of the measured values (angle, velocity, etc.) instead of the origin of the values (corresponding Joint 1-6 blocks) as shown in the subsequent FIGURE 6.13. Necessarily, each (sub) subsystem is connected to the signal bus.



FIGURE 6.13: Screen capture of the Measurements subsystem of the Simulink block diagram

Exemplarily, the structure of the Joint Angles (sub) subsystem (mark m) FIGURE 6.13) is shown in FIGURE 6.14 below.

Measurements are taken with Scope blocks within every (sub) subsystem as they provide displaying (plotting), live viewing, logging, formatting, examining and exporting the captured individual input signals.

In the case of the Joint Angles (sub) subsystem, six Scope blocks were implemented, one for each of the six joint position variables/ joint angles ( $q_1 \dots q_6$ ) to be observed, as shown in the subsequent FIGURE 6.14. Each Scope block receives the corresponding joint-space position set values ( $q_{1SV} \dots q_{6SV}$ , reference trajectories), gained from the `simVar` variable using From Workspace blocks. Furthermore, each Scope block receives the corresponding measured actual values of the joint positions/ joint angles ( $q_1 \dots q_6$ ). This allows the direct comparison of each of the corresponding actual and reference values of each axis as exemplarily shown in the signal plot (FIGURE 6.15) of the Joint Angle Axis 1 Scope block (mark n) FIGURE 6.14).



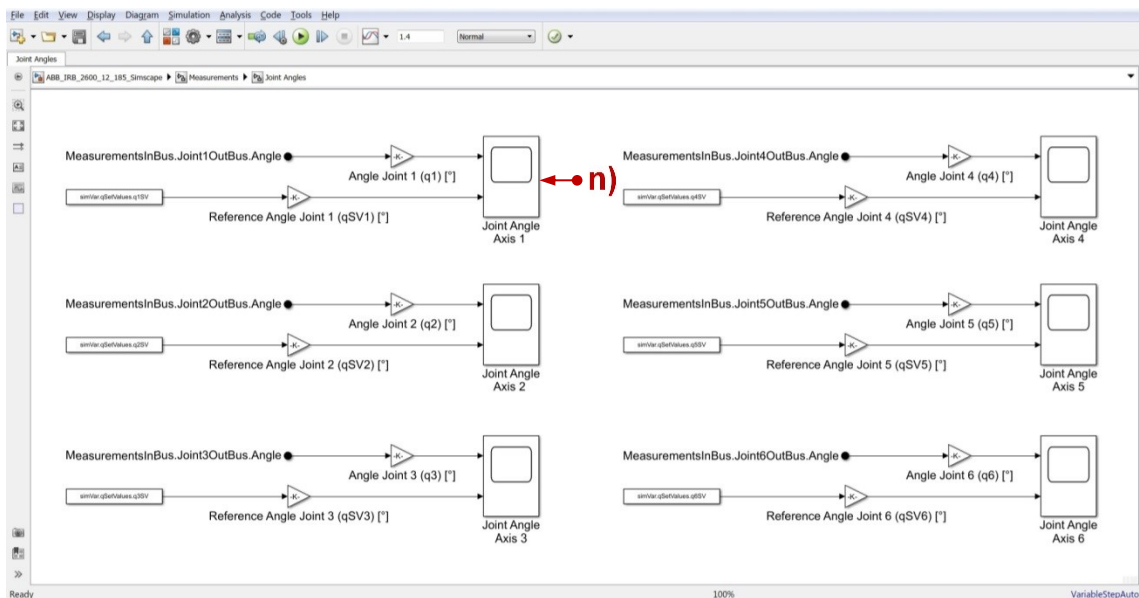


FIGURE 6.14: Screen capture of the Joint Angles (sub) subsystem of the Measurements sub-system

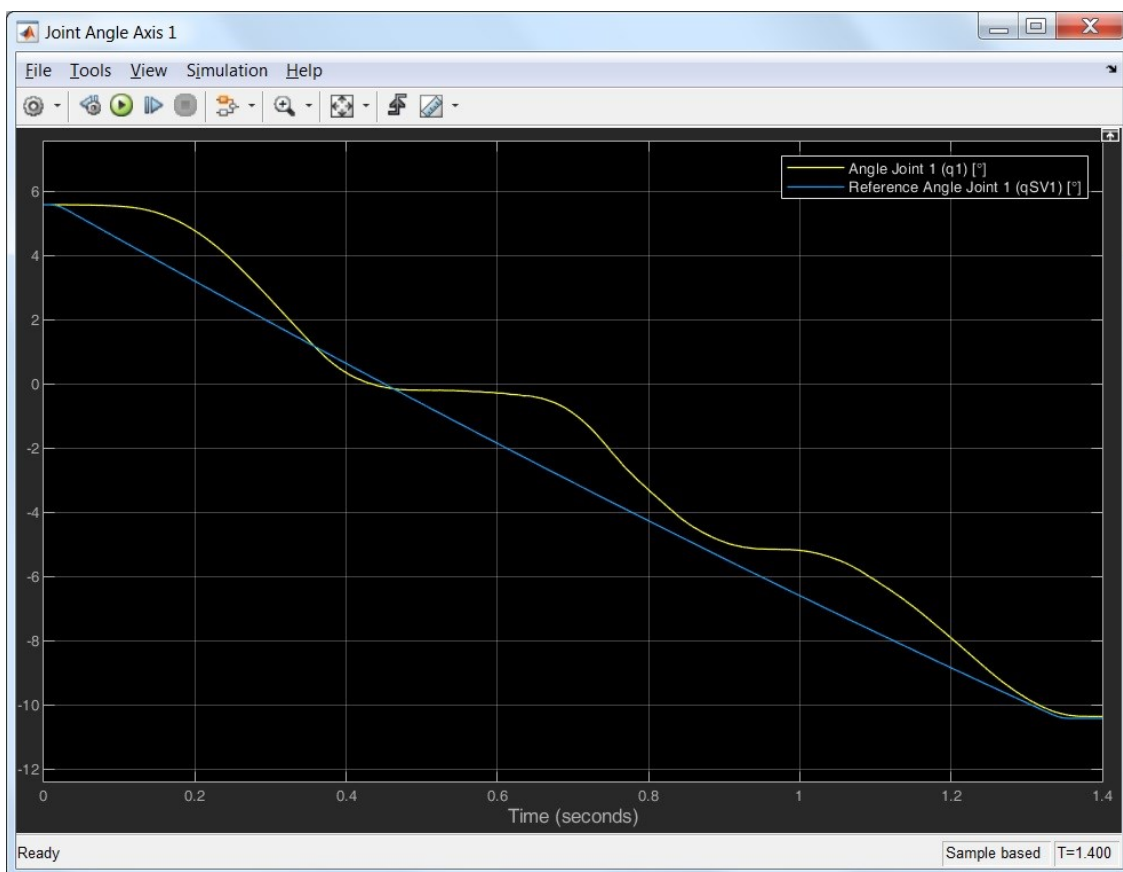


FIGURE 6.15: Screen capture of a Joint Angle Axis 1 Scope block signal plot

In contrary to the Joint Angle (sub) subsystem, the Joint Velocities, Joint Accelerations and Joint Torques (sub) subsystems only contain two Scope blocks each, whereby the individual six signals are bundled related to the principal axes (indices 1-3) and the minor axes (indices 4-6) (see FIGURE 6.16).

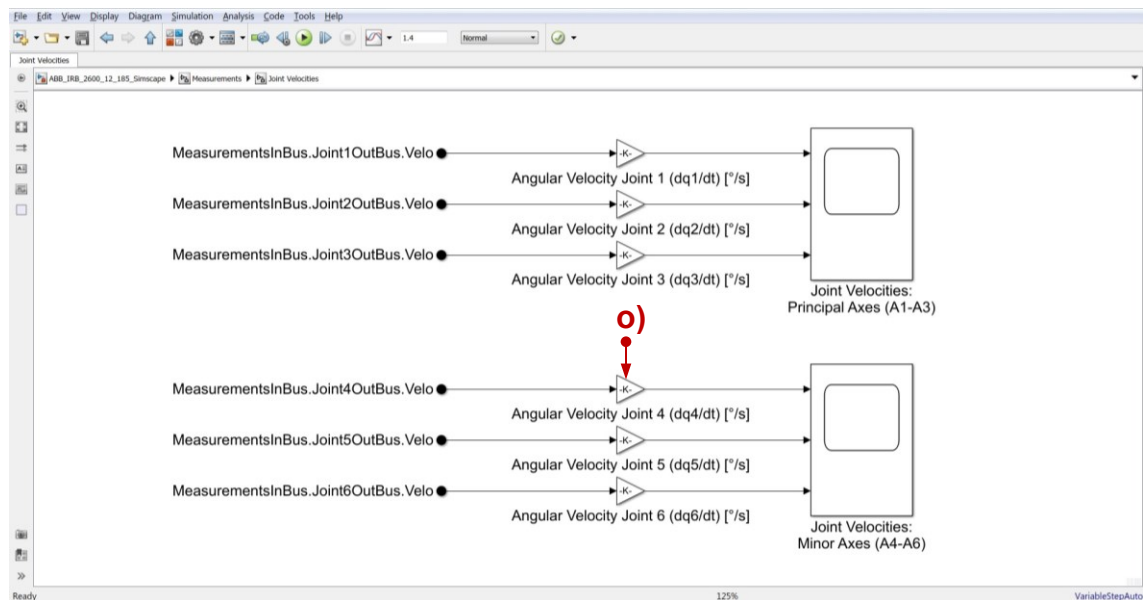


FIGURE 6.16: Screen capture of the Joint Velocities (sub) subsystem of the Measurements subsystem

To increase the comprehensibility of the measured values, the signals of the Joint Angles, Joint Velocities and Joint Accelerations (sub) subsystems are converted from the units [rad], [rad/s] and [rad/s<sup>2</sup>] to the units [°], [°/s] and [°/s<sup>2</sup>], whereas the Joint Torques (sub) subsystem uses the unit [Nm]. The predefined unit conversions can be adapted by changing the gain values of the preceding Gain blocks (see exemplary mark o) in FIGURE 6.16) of each Scope input signal.

The 3D animation/ simulation of the Simulink Simscape Multibody simulation model, represented by the .stl geometry files gained from the CAD model and the calculated kinematics and dynamics, can be viewed from the MATLAB Mechanics Explorers window as exemplarily shown in FIGURE 6.17 below.

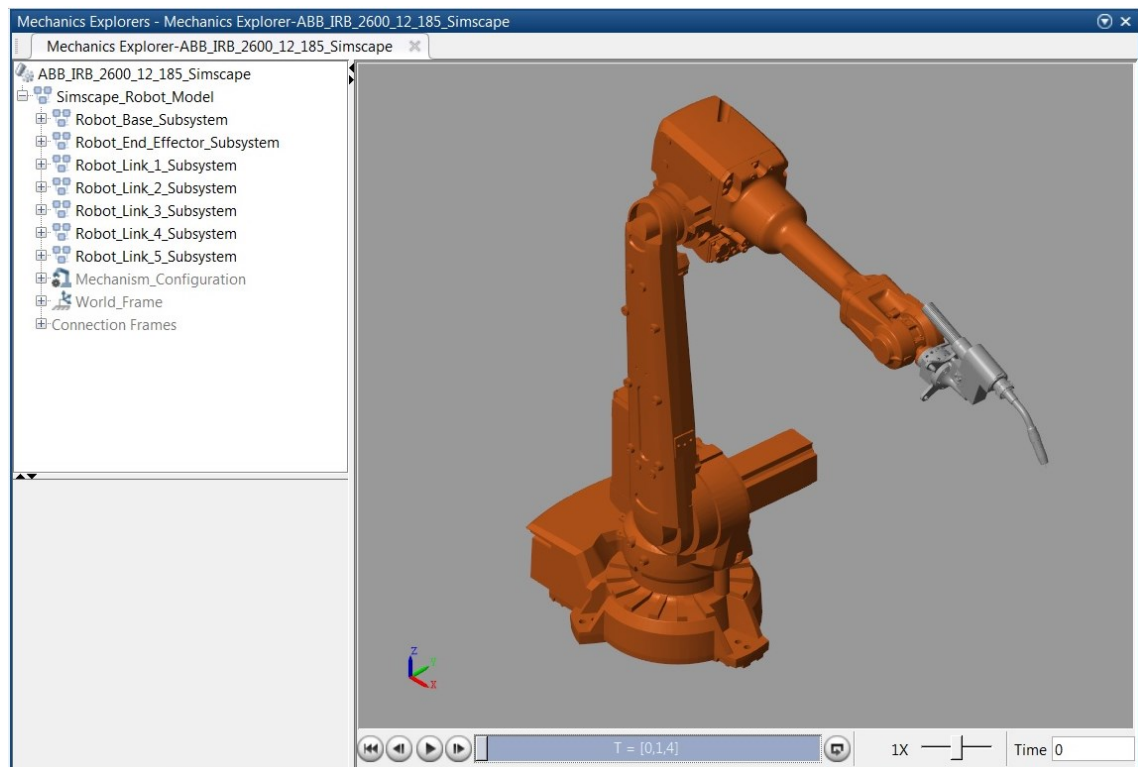


FIGURE 6.17: Screen capture of the robot's Simscape Multibody model simulation animation (Mechanics Explorer)

### 6.3 MATLAB Program(s)

Equally to the structure of section 6.2, the final structure/ flow of the MATLAB Program (part) is presented at the beginning of this section and therefore depicted in FIGURE 6.18 below. The MATLAB General Program Flow chart does only cover the main elements of the program, since a complete semantic description of the MATLAB program requires all 18 individual program flow charts, available from Appendix 4. Program Flow Charts.

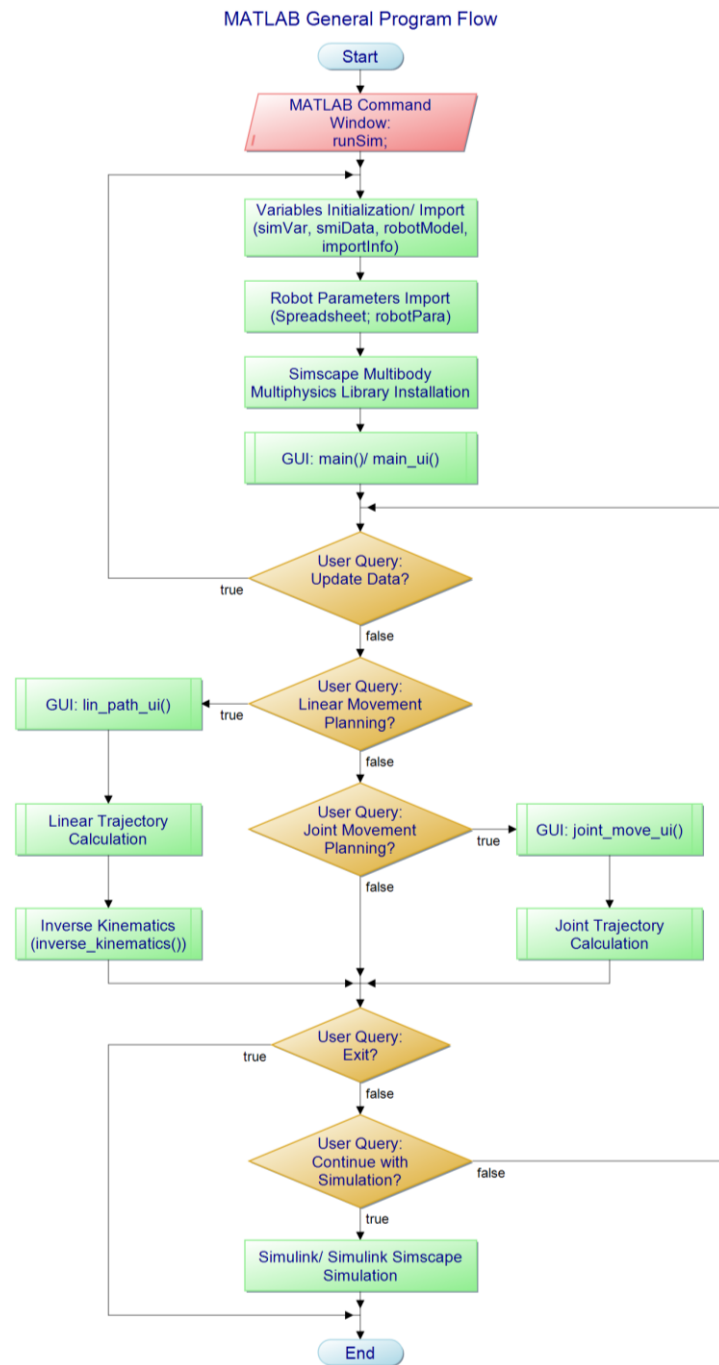


FIGURE 6.18: Flow chart of the MATLAB General Program Flow

### 6.3.1 Program & Program Structure(s)

The MATLAB programming was accomplished by following the basic theoretical programming procedure (section 4.5) and applying the elaborated requirements and specifications of the corresponding conceptual design (section 5.3). The structure, dependencies and interactions of the MATLAB `.m/ .fig` files and all external data (within the data set) are depicted in the subsequent FIGURE 6.19.

The directions of connections refer to the real data flow (from the left to the right: calls, from the right to the left: returns). Further information concerning the shown folders are given in section 6.5.

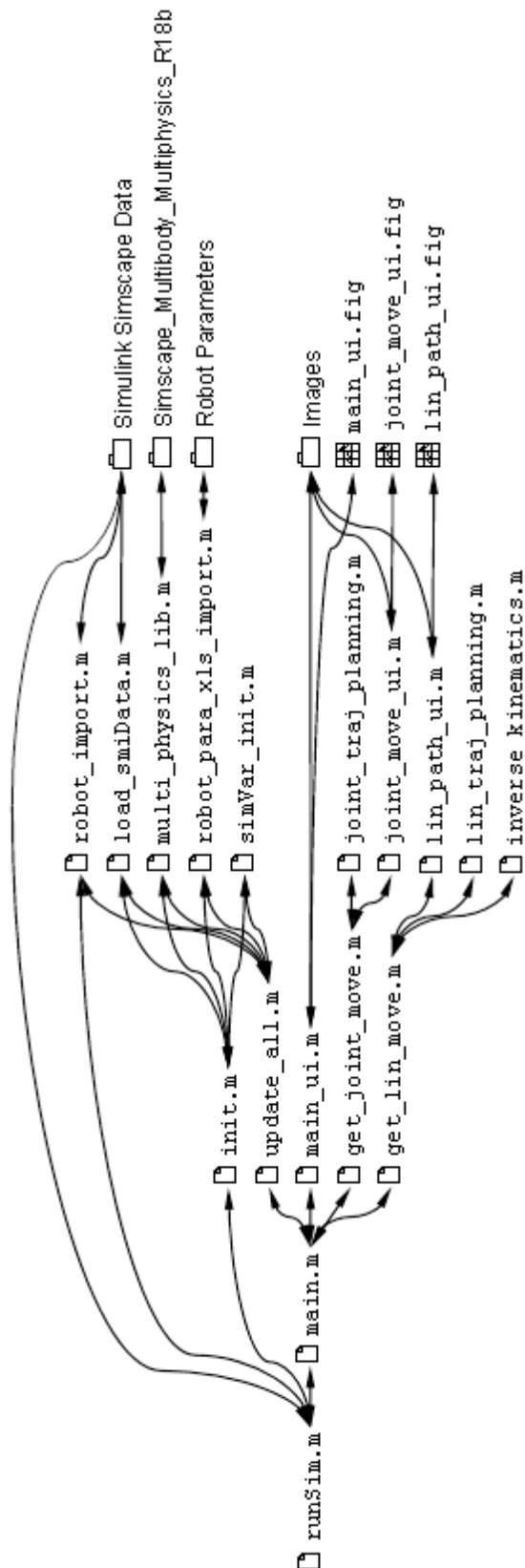


FIGURE 6.19: MATLAB program(s) structure and function/ file dependencies

Comparing the contents of FIGURE 6.19 and TABLE 5.1 reveals that the:

- `update_all.m`
- `get_joint_move.m`
- `get_lin_move.m`

Functions/ files were finally created and implemented but not specified in the conceptual design. Justification: The need of the mentioned functions/ files was not foreseeable at the state of the conceptual design. Hence, no concepts were prepared at that point.

The conceptual designs, including the corresponding program flow charts, were therefore elaborated during the phase of accomplishment. The descriptions of the functions/ files are listed in TABLE 6.4 below.

TABLE 6.4: Listing and description of additionally implemented MATLAB function(s) (files)

MATLAB .m/ .fig File(s):	Description:	Corresp. PFC (Appendix 4. Program Flow Charts) Page:
<code>update_all.m</code>	Update of required variables/ libraries/ data ( <code>simVar</code> , <code>robotPara</code> , <code>smiData</code> , <code>robotModel</code> , <code>importInfo</code> variables and Simscape Multibody Multiphysics Library).	18(18)
<code>get_joint_move.m</code>	Calls <code>joint_move_ui()</code> , then calls <code>joint_traj_planning()</code> to calculate the joint movement trajectory. Finally writes the calculation results into the simulation variable <code>simVar</code> in the required format.	2(18)
<code>get_lin_move.m</code>	Calls <code>lin_path_ui()</code> , then calls <code>lin_traj_planning()</code> to calculate the linear movement trajectory. Finally writes the calculation results into the simulation variable <code>simVar</code> in the required format.	3(18)

All other functions/ files were implemented as conceptualized and described in section 5.3.3, TABLE 5.1.

The limited extent of the document at hand does not allow detailed discussions of the explicit contents and functionalities of each individual function (.m/ .fig file) and the interactions/ dependencies between them. As the corresponding theory, conceptual design and program flow charts are available from the document at hand and function codes are also described by file headers and comments sufficiently, no further explanations are given here.

Excerpts of the final MATLAB code are presented and described more detailed in the later section 6.3.3.

### 6.3.2 Simulation Model Variables

As conceptualized in section 5.3.1, the simulation program requires the entirety of five variables:

- `smiData`
- `robotModel`
- `importInfo`
- `robotPara`
- `simVar`

Which are generated and filled with all required data within the MATLAB program part.

The `smiData` variable is loaded from the `ABB_IRB_2600_12_185_Simscape_DataFile.m` file using the `load_smiData()` function. The `ABB_IRB_2600_12_185_Simscape_DataFile.m` in turn is the automatically generated model data file derived from the Simulink Simscape Multibody Import .xml file (`ABB_IRB_2600_12_185_Simscape.xml`) using the MATLAB `smiimport()` function.

The predetermined variables `robotModel` and `importInfo` are automatically created in the context of the usage of the MATLAB `importrobot()` function. The `robotPara` variable is created by the `robot_para_xls_import()` function. The robot parameter values are read from the `ABB_IRB_2600-12-1.85_Parameters.xlsx` spreadsheet file and stored in the `robotPara` variable (the variable's first level structure is shown in FIGURE 6.20). Hence, the purpose of the `robotPara` variable is the parameterization of the Simulink/ Simulink Simscape simulation model as conceptualized and also described more detailed in the later sections 6.4.2 and 6.4.3.

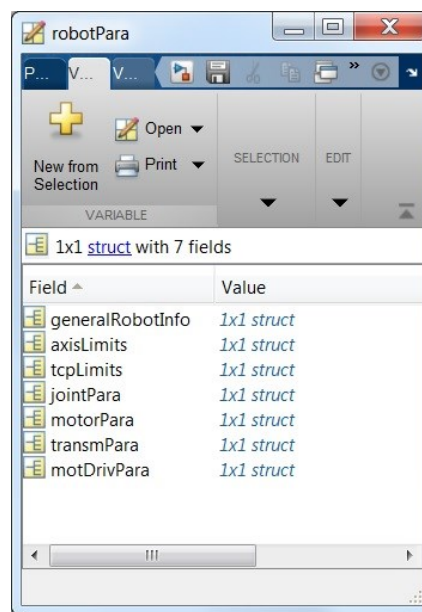


FIGURE 6.20: Screen capture of the first level of the structure of the `robotPara` variable

The `simVar` variable is initialized by the `simVar_init()` function and received and returned from all functions of the MATLAB program part in order to allow all functions to read/ write information from/ to one centralized variable. The `simVar` variable (the variable's first level structure is shown in FIGURE 6.21) also contains and provides the set values (of the reference trajectories) of the control systems structures of the Simulink/ Simulink Simscape simulation model.



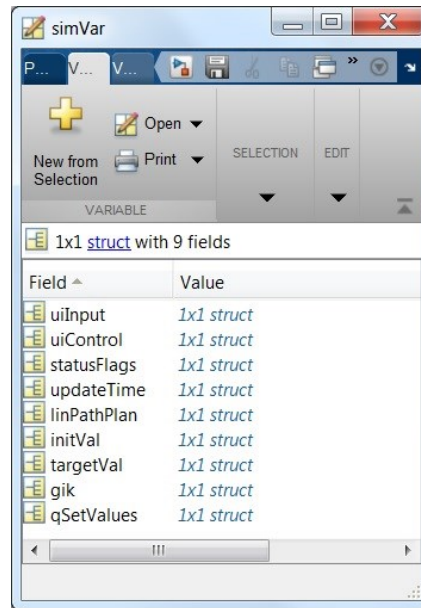


FIGURE 6.21: Screen capture of the first level of the structure of the simVar variable

Summarized, the variables `smiData`, `robotModel` and `importInfo` are only used within the MATLAB program part exclusively, whereby the `robotPara` and `simVar` variables are used in both the MATLAB and the Simulink/ Simulink Simscape program parts.

All variables are made visible/ accessible in the MATLAB “base” Workspace with the transition from the MATLAB to the Simulink/ Simulink Simscape program part, along with the minimum recommended simulation time (in the MATLAB Command Window) as shown in FIGURE 6.22 below.

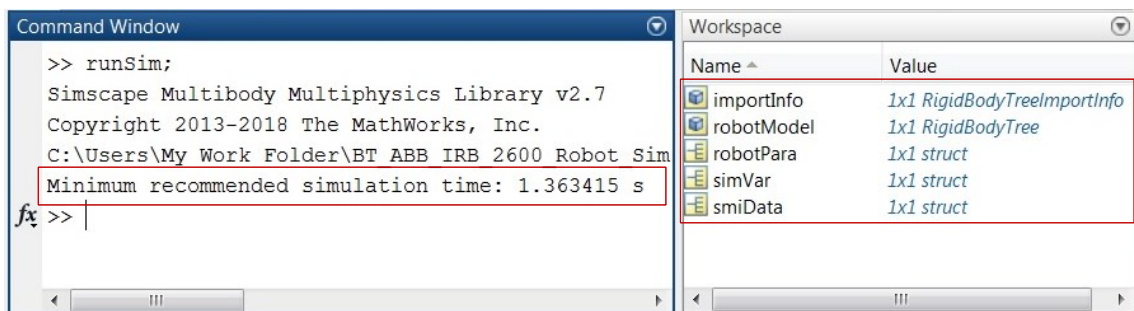


FIGURE 6.22: Screen capture of the MATLAB Command Window and Workspace after the successful MATLAB program execution

All five variables are also described in the TABLE 6.5 below, whereby the `robotPara` variable is described more detailed in the subsequent TABLE 6.6 and the `simVar` variable in the subsequent TABLE 6.7.

TABLE 6.5: Descriptions of the five simulation variables

Name:	Type:	Description/ Purpose:	Initialized/ Created/ Changed by:
<code>simVar</code>	1x1 struct (9 fields)	Contains all required data for the execution of the MATLAB program part. Provides the results of the MATLAB program part to the Simulink/Simulink Simscape program part.	Initialized by <code>simVar_init()</code> , changed by all other functions of the MATLAB program part.
<code>smiData</code>	1x1 struct (3 fields)	Contains the block parameter values of the imported Simscape Multibody simulation model automatically created during the procedure of the execution of the <code>smimport()</code> function.	Created by <code>smimport()</code> , initialized by <code>load_smiData()</code>
<code>robotPara</code>	1x1 struct (7 fields)	Contains values for the parameterization of the block(s) (diagram(s)) of the Simulink/Simulink Simscape simulation model.	<code>robot_para_xls_import()</code>
<code>robotModel</code>	1x1 Rigid-BodyTree	Contains the robotic manipulator's simulation model's kinematic structure (represented by rigid bodies connected by joints) and corresponding parameters.	<code>importrobot()</code>
<code>importInfo</code>	1x1 Rigid-BodyTree-ImportInfo	Contains information concerning the import procedure of the <code>importrobot()</code> function.	

TABLE 6.6: Detailed description of the robotPara variable

Variable:	Fields (First Level):	Description/ Purpose:
robotPara	generalRo- eralRo- botInfo	Contains further subfields (e.g. capacity); contains general information of the real robotic manipulator.
	axisLim- its	Contains further subfields (e.g. range); contains axis/ joint limitations of the Simulink/ Simulink Simscape simulation model equal to the axis/ joint limitations of the real manipulator (e.g. for input filtering in joint_move_ui() and lin_path_ui()).
	tcpLimits	Contains values (e.g. velocity); contains TCP limitations of the Simulink/ Simulink Simscape simulation model equal to the TCP limitations of the real manipulator (e.g. for input filtering in joint_move_ui() and lin_path_ui()).
	jointPara	Contains further subfields and sub subfields (e.g. stateTar); contains values for the parameterization of the revolte joint block(s) (diagram(s)) of the Simulink/ Simulink Simscape simulation model.
	motorPara	Contains further subfields and sub subfields (e.g. ratPow); contains values for the parameterization of the joint motor/ driver block(s) (diagram(s)) of the Simulink/ Simulink Simscape simulation model.
	transmPa- ra	Contains further subfields and sub subfields (e.g. nCdt); contains values for the parameterization of the joint transmission block(s) (diagram(s)) of the Simulink/ Simulink Simscape simulation model.
	motDrivPa ra	Contains further subfields and sub subfields (e.g. vdc); contains values for the parameterization of the joint motor driver block(s) (diagram(s)) of the Simulink/ Simulink Simscape simulation model.

TABLE 6.7: Detailed description of the simVar variable

Variable:	Fields (First Level):	Description/ Purpose:
simVar	uiInput	Contains further subfields and sub subfields; contains inputs of the graphical user interfaces <code>joint_move_ui()</code> and <code>lin_path_ui()</code> .
	uiControl	Contains further subfields (e.g. <code>exeUpdate</code> ); for control functionalities of the main graphical user interface <code>main_ui</code> .
	statusFlags	Contains (flag-) values (either “1” = “true” or “0” = “false”); for the interaction/ control functionalities between the different graphical user interfaces.
	updateTime	Contains the update times of updated/ loaded/ created/ executed data/ libraries/ programs (e.g. Simscape Multibody Multiphysics Library) for the “Last updated:” labels in the <code>main_ui</code> GUI window.
	linPathPlan	Contains further subfields (e.g. <code>pRes</code> ); contains the results of the linear trajectory planning <code>lin_traj_planning()</code> (for <code>get_lin_move()</code> internal use).
	initVal	Contains further subfields (e.g. <code>qStartA</code> ); contains the initial pose (and velocities) of the Simulink/ Simulink Simscape simulation model.
	targetVal	Contains further subfields (e.g. <code>qTargetB</code> ); contains the target pose (and velocities) of the Simulink/ Simulink Simscape simulation model.
	gik	Contains a further subfield ( <code>qRes</code> ); stores the (unformatted) results of the inverse kinematics ( <code>inverse_kinematics()</code> ).
	qSetValues	Contains further subfields ( <code>q1SV...q6SV</code> ); contains the (formatted) set values of the joint angles for the Simulink/ Simulink Simscape simulation model.

### 6.3.3 Motion Planning

Motion planning was implemented within the MATLAB program part, closely following the created flow chart diagram (FIGURE 5.14) and program flow charts described in the conceptual design (section 5.3.3) and using the methods and equations of the corresponding theory comprehensively elaborated and described in the sections 4.2.1 and 4.2.2.

For exemplarily purposes, an excerpt of the code of the `inverse_kinematics()` function (part of the linear movement planning) is shown and described below. The excerpt is related to the core of the `inverse_kinematics()` function, solving the robot manipulator's inverse kinematics with the MATLAB `GeneralizedInverseKinematics` solver.

```

28| gik = robotics.GeneralizedInverseKinematics(
29|         'RigidBodyTree', robotModel, 'ConstraintInputs', ...
29|         {'position', 'aiming', 'joint'});
.
.
.
70| for k=2:length(simVar.linPathPlan.pRes)
...|
73| positionConst.TargetPosition = simVar.linPathPlan.pRes(k,:);
74| aimConst.TargetPoint =          simVar.linPathPlan.pRes(k,:);
...|
77| jointConst.Bounds = [
78|         (simVar.gik.qRes(:,k-1)- maxJointChange)...
78|         (simVar.gik.qRes(:,k-1)+ maxJointChange)];
...|
81| [simVar.gik.qRes(:,k), solInfo] = gik(
81|         simVar.gik.qRes(:,k-1), ...
82|         positionConst, aimConst, ...
82|         jointConst);
83| end

```

In line 28 and 29, the `GeneralizedInverseKinematics System object™` `gik` is created. The `gik` object bases on the robotic manipulator's model's kinematic structure, mapped by a `RigidBodyTree` object, in turn represented by the `robotModel` variable. During the `gik` object creation, a set of the kinematic constraints objects (`ConstraintInputs`), to be applied for the later inverse kinematics solving, need to be predetermined. In the case at hand, the `position`, `aiming` and `joint` constraint objects were predetermined and are created from the corresponding classes accordingly.

The position constraint `position` causes the end effector/ tool (tip of the welding torch) to match the contemporary waypoint of the linear trajectory, whereby the aiming constraint `aiming` causes the tool's z-axis ( $Z_{tool}$ ) to aim at the contemporary waypoint at the same time. An explanatory depiction is shown in the subsequent FIGURE 6.23 .

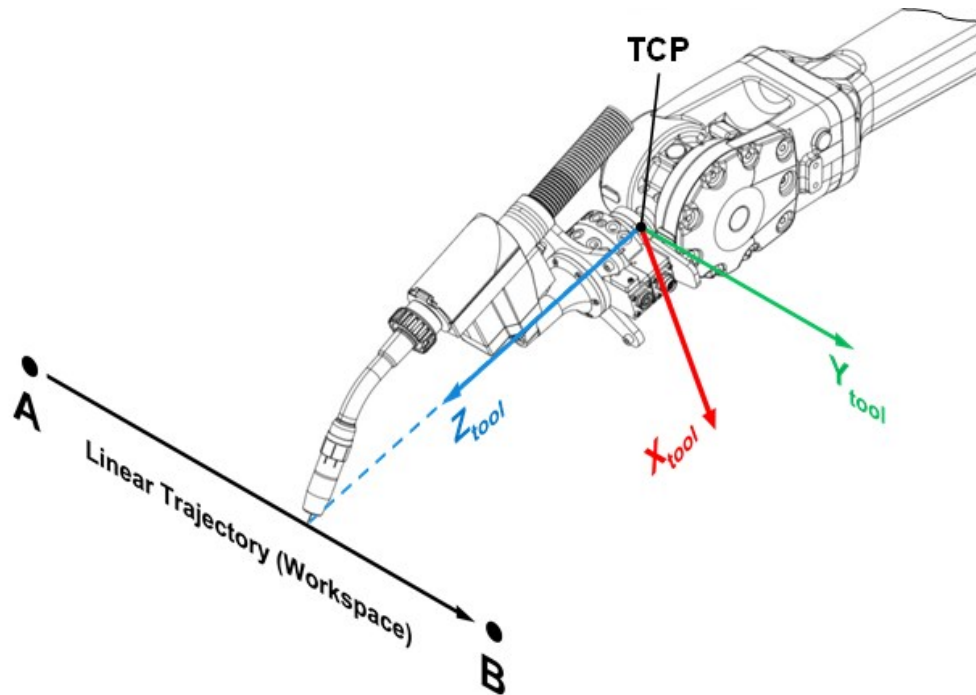


FIGURE 6.23: Depiction of the linear movement's end effector position and pose

As there were no additional constraints defined related to the restriction of the tools pose, the rotation of the tool around its z-axis is not restricted and therefore determined by the inverse kinematics solver. Typically, the z-axis rotation remains unchanged by the inverse kinematics solver ( $q_{Res_{6,k}} = 0^\circ$ ), unless a rotation is absolutely required to reach the contemporary waypoint within the workspace. If required, the tool's z-axis rotation can be constrained/ restricted by the user utilizing predefined constraint creation code available from the `inverse_kinematics.m` file.

Due to the non-uniqueness of the inverse kinematics, the joint constraint `joint` was predefined in order to limit the maximum changes of the angular joint values ( $q_1 - q_6$ ) between each robot's pose related to the waypoint of the linear (workspace) trajectory.

Solving the inverse kinematics is accomplished individually for each waypoint of the linear (workspace) trajectory (stored in the `simVar.linPathPlan.pRes` variable, see FIGURE 6.24) within the for-loop from code line 70 to line 83. As the inverse kinematics solver is a numerical solver, solving always requires an initial guess of the robot manipulator's resulting pose ( $q_{1,k} - q_{6,k}$ ). Therefore, the start pose of the robot manipulator (`qStartA(1) - qStartA(6)` variables) ( $k = 1$ ) is solved once before the solver loop (line 70 to line 83), using the home pose of the manipulator ( $q_{1,1} - q_{6,1} = 0^\circ$ ) as initial guess. This also justifies the for-loop's index  $k$  starting from the value 2.

In line 73 and 74, the position and aiming constraint objects are updated to the related contemporary waypoint ( $k$ ) of the linear trajectory.

In line 77 and 78, the joint constraint object is updated to the contemporary maximum allowed angular joint changes, based on the robot's pose related to the previous waypoint ( $k-1$ ) and gained from the `simVar.gik.qRes` variable.

In line 81 and 82, the `gik` object is solved for the related robot manipulator's pose of the contemporary waypoint ( $k$ ), using the `GeneralizedInverseKinematics` solver. The solving is accomplished using the robot's previous pose ( $k-1$ ) as initial guess. The results ( $qRes(1, k) - qRes(6, k)$ ) are stored in the `simVar.gik.qRes` variable.

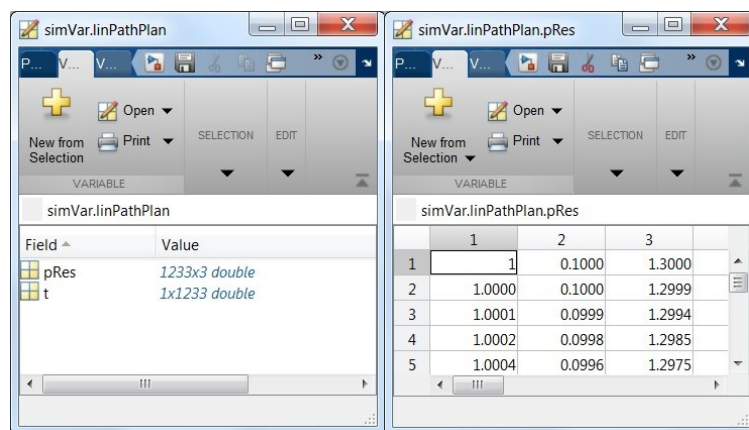


FIGURE 6.24: Screen capture of the `simVar.linPathPlan` and `simVar.linPathPlan.pRes` variables (example)

Within the `inverse_kinematics()` function, the inverse kinematics results are stored in the  $6 \times k$ -dimensional variable `simVar.gik.qRes`.

Deviating from the conceptual design, the inverse kinematics results are not provided to the MATLAB Workspace using the initially conceptualized  $k \times 7$ -dimensional reference trajectory set value variable  $q^{SV}$ .

As the joint-space position set values (inverse kinematics results) of the Joint Controller (sub) subsystem are individually obtained from the MATLAB "base" Workspace using From Workspace blocks, the set values are stored in the six individual  $k \times 2$ -dimensional variables (`simVar.qSetValues.q1SV` - `simVar.qSetValues.q6SV`). These variables are created by the `get_lin_move.m` function, whereby the first column of each of variable contains the explicit time series values and the second column contains the corresponding joint position/ angle values as exemplarily shown in FIGURE 6.25 below.

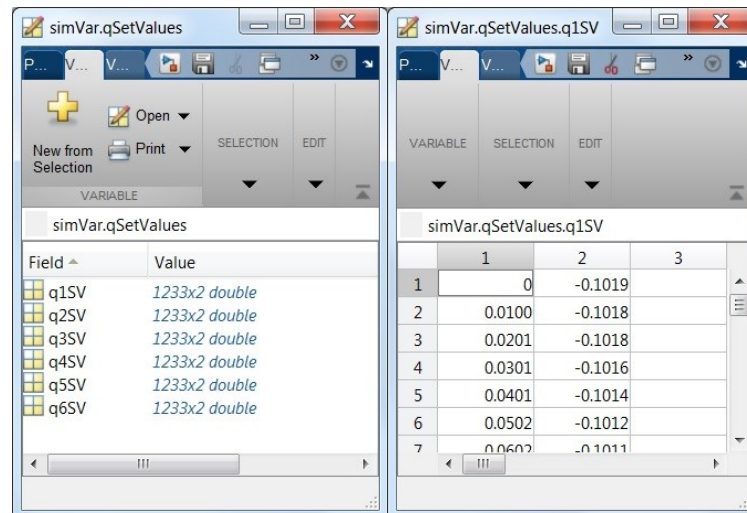


FIGURE 6.25: Screen capture of the `simVar.qSetValues` and `simVar.qSetValues.q1SV` variables (example)

### 6.3.4 Graphical User Interfaces

The graphical user interfaces were implemented closely following the determined conceptual design (section 5.3.2) and the corresponding program flow charts, using the MATLAB GUIDE tool. The three GUI are represented by their corresponding MATLAB `.m` and `.fig` files each:



- Main GUI: `main_ui.m & main_ui.fig`
- Joint Movement GUI: `joint_move_ui.m & joint_move_ui.fig`
- Linear Movement GUI: `lin_path_ui.m & lin_path_ui.fig`

From the user's perspective, the three GUI and a number of further MATLAB message boxes/ dialog boxes implemented map the complete MATLAB program part.

The main graphical user interface `main_ui` is the central GUI of the MATLAB program part and shown in FIGURE 6.26 below. All other program functionalities and GUI are accessed from the `main_ui` window, can be repeated as often as required and also automatically return there, except the case of opening the Simulink simulation model or the exiting of the MATLAB program part.

Status labels (see mark p) FIGURE 6.26) indicate whether the corresponding entry (e.g. `simVar`) is ready ("Ready!" text and a green shaded label) or not ready ("Not Ready!" text and a red shaded label). Deviating from the conceptual design sketch (FIGURE 5.12), Last updated timestamp labels (see mark q) FIGURE 6.26) were added for each corresponding entry in order to allow the user to check when the corresponding entry was updated and/ or a specific procedure was executed.

By pressing the Update button of the `main_ui` window (see mark r) FIGURE 6.26), the update procedure is activated and the `update_all()` function is called. The update procedure is guided by further information provided via MATLAB message boxes/ dialog boxes and leads to the update of all five simulation variables (also the not listed `importInfo` variable) and the Simscape Multibody Multiphysics Library.

After the successful execution, the corresponding Last updated timestamp labels are refreshed accordingly and the program returns to the `main_ui` window.

The joint movement planning GUI `joint_move_ui` is accessed by pressing the Joint Movement button (see mark s) FIGURE 6.26) and shown in FIGURE 6.27.

The linear movement planning GUI `lin_path_ui` is accessed by pressing the Linear Movement button (see mark t) FIGURE 6.26) and shown in FIGURE 6.28.

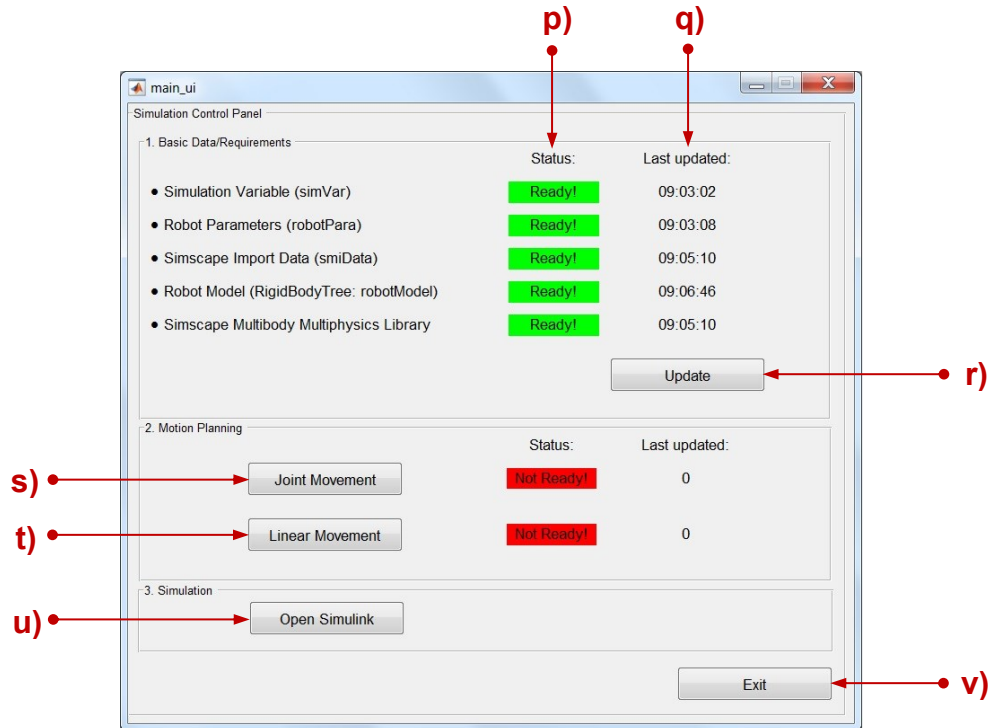


FIGURE 6.26: Screen capture of the MATLAB main GUI `main_ui`

Both movement GUI, `joint_move_ui` and `lin_path_ui` provide specific corresponding descriptions/ instructions and a descriptive image based on the CAD model assembly of the robot manipulator's model. The input areas for the start and target poses/ positions were adapted accordingly to the corresponding movement type. For the increase of the comprehensibility, robot axis angles inputs (Pose, A1-A6) were implemented using the unit [°] and workspace coordinates (Position, [x y z]) using the unit [mm]. Non-SI unit inputs ([%], [°], [mm/s] and [mm]) were also implemented within the Movement Parameters definition area in order to provide common input value formats.

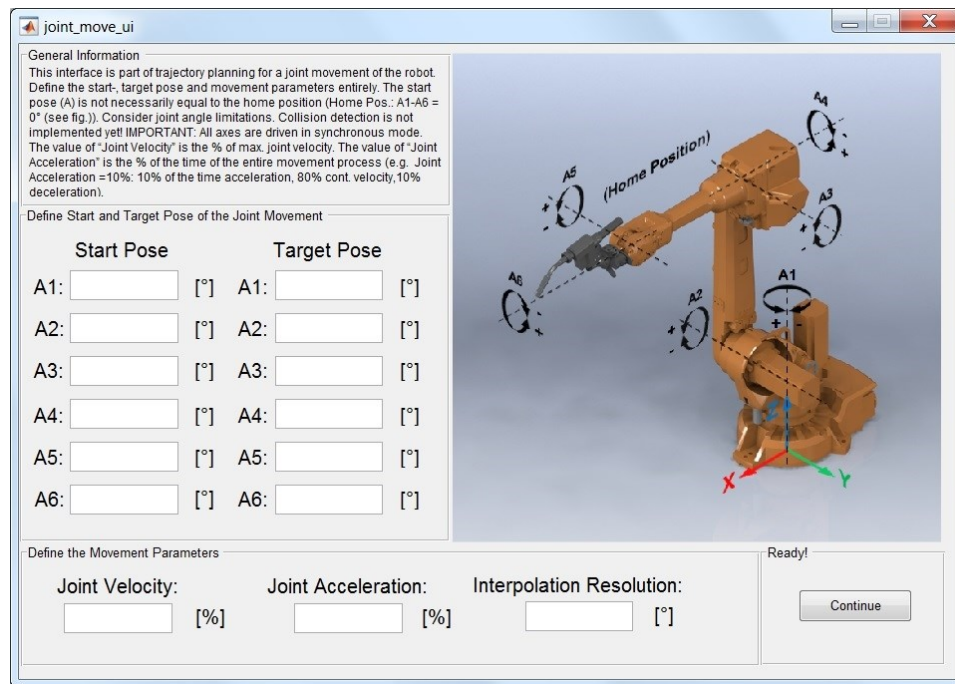


FIGURE 6.27: Screen capture of the MATLAB joint movement GUI `joint_move_ui`

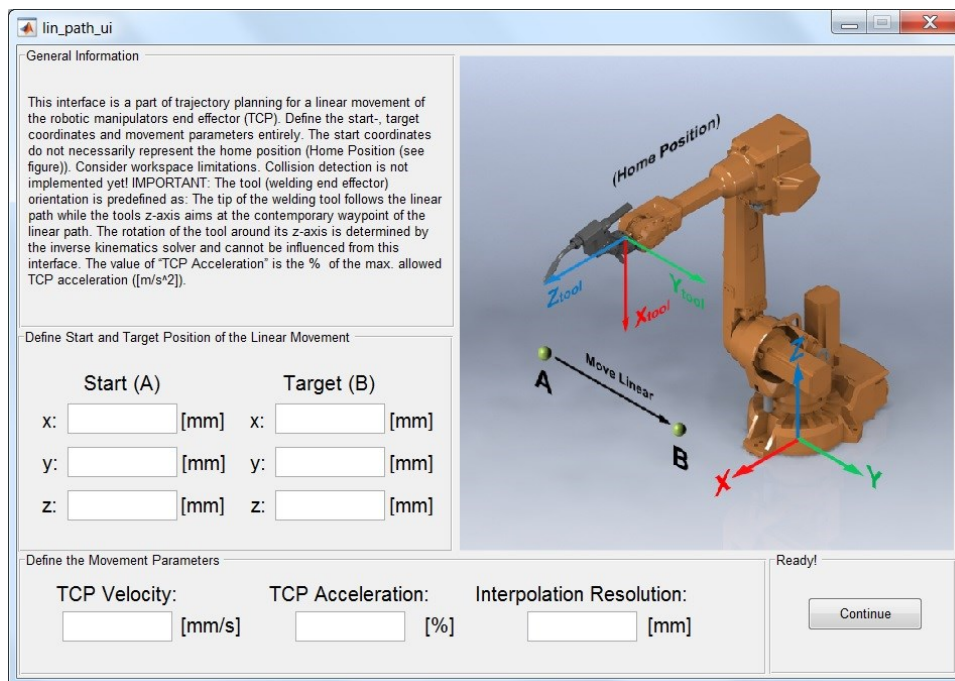


FIGURE 6.28: Screen capture of the MATLAB linear movement GUI `lin_path_ui`

Furthermore, all inputs of the movement GUI are filtered for being a number and being within the allowed boundaries, whereby the corresponding boundaries are gained from the `robotPara` variable (and therefore from the parameter spreadsheet). All input procedures are looped and show error or warning messages as long as inputs are faulty and/ or incomplete, exemplarily depicted in FIGURE 6.29 below.



FIGURE 6.29: Screen capture of an invalid input MATLAB error message box

In the case of correct and complete inputs, pressing the Continue button of the movement GUI will cause closing the related GUI window and the execution of the related trajectory calculation. After the successful trajectory calculation, the program returns automatically to the `main_ui` window and the corresponding Last updated timestamp labels are refreshed accordingly.

Motion Planning can only be prepared for either the Joint Movement or the Linear Movement at a time. Hence, only the latest executed motion planning result is available, whereas former results are deleted and the corresponding Status Label is set to the “Not Ready!” state.

The Simulink/ Simulink Simscape program part is accessed by pressing the Open Simulink button of the `main_ui` window (see mark u) FIGURE 6.26). With the initiation of this procedure, the MATLAB program part is terminated, all variables are made visible/ accessible in the MATLAB “base” Workspace and the minimum recommended simulation time is printed to the MATLAB as shown in the earlier FIGURE 6.22.

The MATLAB Program part can only be exited without errors from the `main_ui` window using the Exit button (see mark v) FIGURE 6.26). In the case of the cancelation of the MATLAB program using the Exit button, all variables are deleted, thus, no variables are made visible/ accessible in the MATLAB Workspace.

#### 6.4 Simulation Model Parameters

The task of the simulation model parameterization covered firstly the acquisition of all required and obtainable parameters of the model.

Secondly, the preparation and compilation of the entirety of parameters of the Simulink Simscape simulation model was accomplished. Thirdly, the conceptualized method of the provision of the parameters to the simulation model was applied.

#### **6.4.1 Parameter Acquisition**

The subsection of the parameter acquisition is divided into another two sub subsections, the approximation of the industrial robot's CAD model part/ body masses and the acquisition of all other applied parameters. This separation was made due to the distinctive characters of the methods of acquisition.

The first attempt made in the context of the acquisition of main parameters of the simulation model based on a request sent to the Finnish branch ABB (Finland) Oy, of the industrial robot's manufacturer ABB Asea Brown Boveri Ltd. in order to obtain:

- Links masses and links CoM coordinates
- (Joint actuation) motor types and their main electrical and mechanical parameters (e.g. rotor inertia, viscous damping coefficients)
- Gearbox types, ratios, efficiencies and inertias

The manufacturer replied that none of the requested data can be shared.

As identification measurements were not covered by the scope of the thesis work, disassembling of the industrial robot was not a realistic option and due to the general lack of freely accessible information, the subsequently explained methods were conducted:

From non-public manufacturer's maintenance and spare parts lists documents of the industrial robot, owned and provided by the client (TAMK), the subsequently listed information were obtained exemplarily for the robot's third axis gearbox:

TABLE 6.8: ABB IRB 2600 gearbox spare part information (axis 3)

Position:	Axis:	Spare Part Number:	Type:	Variants:
3	Axis-3	3HAC028705-004	RV-42N, i=126	IRB 2600 IRB 2600ID

And exemplarily for the robot's axis 4, 5 & 6 (joint actuation) motors:

TABLE 6.9: ABB IRB 2600 motor spare part information (axis 4, 5 &amp; 6)

Position:	Spare Part Number:	AC Motor with Pinion:
4	3HAC030216-003	Axis-4, -5 & -6

In the case of the gearboxes, common internet search engines were used in order to obtain more detailed information. The finding was made that “RV-42N” is a specific model of a 2-stage high-precision cycloidal reduction gear of the RV<sup>®12</sup>-N series of the Nabtesco Corporation.



FIGURE 6.30: Cycloidal reduction gear of the RV-N series of the Nabtesco Corporation (Nabtesco Corporation 2019a)

With the help of the ratio value “i” and the type description “RV-42N” (see TABLE 6.8), the gearbox model and its parameters were searched and found from the official technical datasheet of the Nabtesco Corporation (Nabtesco Corporation 2015) as shown below (FIGURE 6.31, FIGURE 6.32 and FIGURE 6.33).

<sup>12</sup> RV<sup>®</sup> is a registered trademark of the Nabtesco Corporation

Model	Output speed (rpm)			5	10	15	20	25	30	40	50	60
	Ratio code	R Speed ratio		Output torque (Nm) / input capacity (kW)								
		Shaft rotation	Case rotation									
RV-25N	41	41	40	341 / 0.25	277 / 0.41	245 / 0.55	255 / 0.67	210 / 0.79	199 / 0.89	183 / 1.09	171 / 1.28	162 / 1.45
	81	81	80									
	107.66	323/3	320/3									
	126	126	125									
	137	137	136									
	164.07	2133/13	2120/13									
RV-42N	41	41	40	573 / 0.43	465 / 0.70	412 / 0.92	378 / 1.13	353 / 1.32	335 / 1.50	307 / 1.84	287 / 2.15	272 / 2.44
	81	81	80									
	105	105	104									
	126	126	125									
	141	141	140									
	164.07	2133/13	2120/13									

FIGURE 6.31: Manufacturer's rating table of the RV-N series cycloidal reduction gear (Nabtesco Corporation 2015, 8, modified)

T <sub>0</sub> Rated torque (Note 7)	N <sub>0</sub> Rated output Speed	K Rated service life	T <sub>S1</sub> Allowable acceleration/ deceleration torque	T <sub>S2</sub> Momentary maximum allowable torque	N <sub>S0</sub> Allowable Output Speed (Note 1) Duty ratio: 100%	N <sub>S1</sub> Allowable Output Speed (Note 1) Duty ratio: 40%	Backlash	Lost motion	Angular transmission error (Max.)	Startup efficiency (Typical value)	M <sub>C1</sub> Allowable moment (Note 4)	M <sub>C2</sub> Momentary allowable moment (Max.)	W <sub>r</sub> Allowable radial load (Note 10)	I Reduced value of the inertia moment for the input shaft (Note 5)	Weight
(Nm)	(rpm)	(h)	(Nm)	(Nm)	(rpm)	(rpm)	(arc.min.)	(arc.min.)	(arc.sec.)	(%)	(Nm)	(Nm)	(N)	(kgm <sup>2</sup> )	(kg)
245	15	6,000	612	1,225	57	110	1.0	1.0	70	80	784	1,568	6,975	1.71×10 <sup>-5</sup> 6.79×10 <sup>-6</sup> 4.91×10 <sup>-6</sup> 4.03×10 <sup>-6</sup> 3.62×10 <sup>-6</sup> 3.26×10 <sup>-6</sup>	3.8
412	15	6,000	1,029	2,058	52	100	1.0	1.0	60	80	1,660	3,320	12,662	4.43×10 <sup>-5</sup> 1.87×10 <sup>-5</sup> 1.42×10 <sup>-5</sup> 1.07×10 <sup>-5</sup> 1.01×10 <sup>-5</sup> 7.66×10 <sup>-6</sup>	6.3

FIGURE 6.32: Manufacturer's rating table of the RV-N series cycloidal reduction gear (continuation) (Nabtesco Corporation 2015, 9, modified)

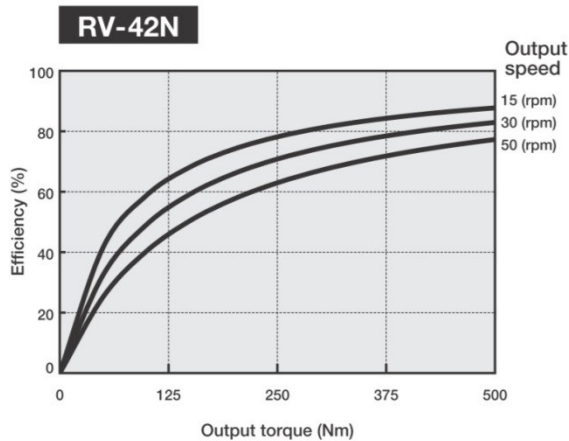


FIGURE 6.33: Manufacturer's efficiency table of the RV42-N cycloidal reduction gear (Nabtesco Corporation 2015, 36)

In the case of the motors, the information of an AC asynchronous motor type was also found from the robot manufacturer's maintenance documents.

Furthermore, the information of the country of origin and the motors net weight (amongst others) were obtained from the industrial robot's manufacturer's website (ABB Asea Brown Boveri Ltd. 2019d):

Country of Origin: Japan (JP)

Product Net Weight: 1.7 kg



FIGURE 6.34: ABB IRB 2600 Axis 4, 5 & 6 AC motor (ABB Asea Brown Boveri Ltd. 2019d)

With the help of the above mentioned information and the information printed to the label of the motor, the motor manufacturer was identified as: TAMAGAWA SEIKI Co., Ltd. As the TAMAGAWA SEIKI Co., Ltd. only offers one applicable AC asynchronous servomotor product series, TBL-I IV Series, the assumption was made that motors with similar characteristics and parameters can be found from the manufacturer's product catalogue (TAMAGAWA SEIKI Co., Ltd. 2019) as shown in the subsequent FIGURE 6.35 and FIGURE 6.36.

電圧 Voltage	フランジ Mounting flange	形式 Model	出力 Output	トルク Torque		電流 Current		回転数 Speed		ロータイナーシャ Rotor inertia		質量 Mass		
				定格 Rated	最大 Max	定格 Rated	最大 Max	定格 Rated	最大 Max	BK無 Standard	BK有* With brake	標準 Standard	ブレーキ付 With brake	
[V]	[mm]		[W]	[Nm]	[Nm]	[Arms]	[Arms]	[min <sup>-1</sup> ]	[min <sup>-1</sup> ]	【10 <sup>-4</sup> kgm <sup>2</sup> 】		【kg】		
AC200	□40	TSM3101	30	0.095	0.33	1.1	3.4	3,000	6,000	0.023	0.028	0.3	0.5	
		TSM3102	50	0.159	0.56	1.1	3.4			0.033	0.038	0.4	0.6	
		TSM3104	100	0.318	1.11	1.4	4.7			0.062	0.067	0.5	0.7	
	□60	TSM3201	100	0.318	1.11	1.5	4.6	3,000	6,000	0.12	0.17	0.4	0.9	
		TSM3202	200	0.64	2.24	2.2	7.3			0.24	0.28	0.6	1.3	
			TSM3204	400	1.27	4.46	3.5	12.1			0.46	0.50	1.1	1.7
	□80	TSM3301	200	0.64	2.24	2.1	6.9	3,000	6,000	0.45	0.60	1.1	1.8	
		TSM3302	400	1.27	4.46	3.6	12.6			0.83	1.00	1.6	2.3	
		TSM3303	600	1.91	6.69	4.8	16.5			1.21	1.38	2.1	2.8	
		TSM3304	750	2.39	8.36	6.5	22.1			1.50	1.66	2.4	3.2	

FIGURE 6.35: TBL-I IV series compact size AC servomotor basic specifications (TAMAGAWA SEIKI Co., Ltd. 2019, 2, modified)



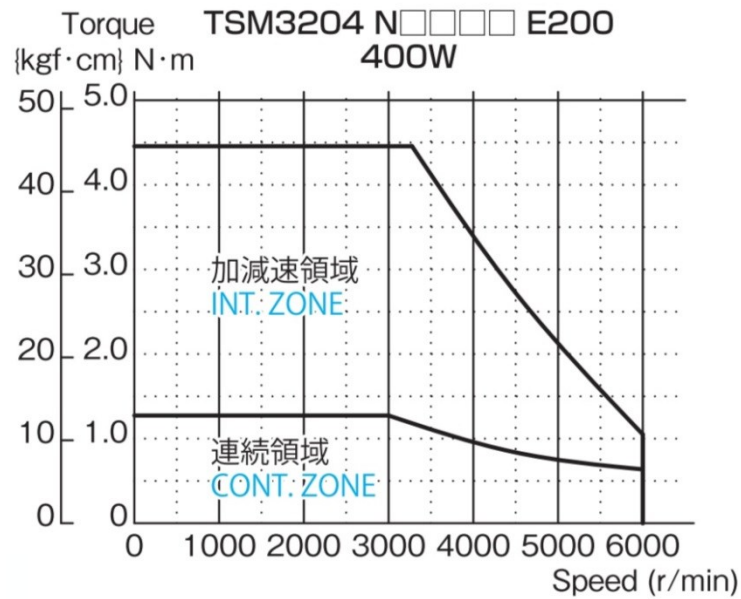


FIGURE 6.36: TSM3204 400W AC200V torque characteristic diagram (TAMAGAWA SEIKI Co., Ltd. 2019, 8)

The described procedures were repeated for all other gearboxes and motors. All obtained parameters were collected, evaluated if necessary and stored at the corresponding entries of the worksheets (“(8) Transmission Parameters” and “(7) Motor Parameters”) of the “ABB\_IRB\_2600-12-1.85\_Parameters.xlsx” spreadsheet (see also section 6.4.2).

### CAD Model Rigid Body (Link) Masses:

Due to the lack of available information and resources, the decision was made to approximate the individual link masses based on the geometrical information available from the CAD model. The approximation based on the approach of an average equally distributed overall density  $\rho^*$ .

Firstly, the overall volume  $V_{\text{tot}}$  of the CAD model of the industrial robot was determined by the summation of the individual link volumes, excluding the tool/end effector.

$$V_{\text{tot}} = \sum_{i=0}^6 V_i = 102.85 \cdot 10^{-3} \text{ m}^3 \quad (6.1)$$

Secondly, the overall mass of the real robotic system without any further applications  $m_{\text{tot}}$  was obtained from the manufacturer's product specifications document (ABB Asea Brown Boveri Ltd. 2019b, 12):

$$m_{\text{tot}} = 284 \text{ kg} \quad (6.2)$$

Subsequently, the average equally distributed overall density  $\rho^*$  was calculated:

$$\rho^* = \frac{m_{\text{tot}}}{V_{\text{tot}}} = \frac{284 \text{ kg}}{102.85 \cdot 10^{-3} \text{ m}^3} = 2761.26 \frac{\text{kg}}{\text{m}^3} \quad (6.3)$$

The average equally distributed overall density  $\rho^*$  was then applied to each individual CAD part within the CAD environment. Furthermore, the weight of the welding torch end effector was gained from a real measurement using a common scale and also applied to the virtual CAD representation.

Based on the available geometrical information and the density  $\rho^*$ , the CAD software automatically calculated the rigid body (link) parameters such as:

- CoM, Mol, Pol
- Frames; (main) axes of inertia

Exemplarily, link 2 of the robot manipulator's CAD model is shown in the subsequent FIGURE 6.37, showing the calculated and displayed frame of the main axes ( $l_x$ ,  $l_y$ ,  $l_z$ ; CoM related).



FIGURE 6.37: Screen capture of link 2 of the robot manipulator's CAD model

The approximated mass of the link 2 part of the simulation model was calculated exemplarily by hand:

$$m_2 = V_2 \cdot \rho^* = 15.67 \cdot 10^{-3} \text{ m}^3 \cdot 2761.26 \frac{\text{kg}}{\text{m}^3} = 43.28 \text{ kg} \quad (6.4)$$

TABLE 6.10 below contains a listing of the individual solid body volumes and approximated link masses of the virtual representations of the manipulator's links.

TABLE 6.10: Robot manipulator's link mass and volume information

i:	Link $i$ :	Solid Body Volume $V_i$ [m <sup>3</sup> ]:	Overall Density $\rho^*$ [kg/m <sup>3</sup> ]:	Approx. Link Mass $m_i$ [kg]:
0	Base	31.82E-03	2761.26	87.87
1	Link 1	25.31E-03		69.88
2	Link 2	15.67E-03		43.28
3	Link 3	23.58E-03		65.11
4	Link 4	6.02E-03		16.62
5	Link 5	0.36E-03		0.99
6	Link 6	0.0087E-03		0.24
$\Sigma$		102.85E-03	-	284
End Effector		-	-	5.25

The acquisition of all other parameters not covered by this section of the document is continued in section 6.7.2 in the context of the optional task of virtual identification measurements.

#### 6.4.2 Parameter Spreadsheet

Based on the general requirements of the modularity, extensibility and convenient usage of the simulation program, the decision was made to provide the parameters of the Simulink Simscape simulation model block diagram blocks indirectly but automatically via variables (FIGURE 6.39) from the MATLAB Workspace as described in the conceptual design (section 5.2.3 and 5.3.1). As the contents of the parameter variables of the simulation model need to be obtained from any source as well, the decision was made to compile and save all required parameters in an external but centralized file. This centralized file in turn is then read during the initialization of the simulation program in order to write the model parameters to the corresponding (MATLAB Workspace) variable `robotPara`. Therefore, a Microsoft<sup>®</sup> Excel<sup>®13</sup> spreadsheet:

“ABB\_IRB\_2600-12-1.85\_Parameters.xlsx”

---

<sup>13</sup> Microsoft<sup>®</sup> Excel<sup>®</sup> is a registered trademark of Microsoft Corporation

Containing the clearly arranged parameter compilation was created. Deviating from the unit definitions and regulations (section 3.1), the units [°], [°/s] and [°/s<sup>2</sup>] were used for some parameter ranges of the spreadsheet to enable the input of values from the manipulator’s manufacturer’s documents directly without any unit conversions.

Exemplarily, the sixth worksheet (Joint Parameters) of the spreadsheet is shown in the FIGURE 6.38 below.

ABB IRB 2600-12/1.85													
(Revolute) Joint Parameters					State Targets		Internal Mechanics			Bearing			
n:	Description:	Corresponding Axis:	Base (B):	Follower (F):	Position Target Value [rad]:	Velocity Target Value [rad]:	Equilibrium Position [rad]:	Spring Stiffness [N*m/rad]:	Damping Coefficient [N*m/(rad*s)]:	Breakaway Friction Torque [N*m]:	Breakaway Friction Velocity [rad/s]:	Coulomb Friction Torque [N*m]:	Viscous Friction Coefficient [N*m/(rad*s)]:
1	Joint 1	Axis 1	Base	Link 1	0,0000	0,0000	0,0000	0,0000	0,0000	0,000001	0,000001	0,000001	0,000001
2	Joint 2	Axis 2	Link 1	Link 2	0,0000	0,0000	0,0000	0,0000	0,0000	0,000001	0,000001	0,000001	0,000001
3	Joint 3	Axis 3	Link 2	Link 3	0,0000	0,0000	0,0000	0,0000	0,0000	0,000001	0,000001	0,000001	0,000001
4	Joint 4	Axis 4	Link 3	Link 4	0,0000	0,0000	0,0000	0,0000	0,0000	0,000001	0,000001	0,000001	0,000001
5	Joint 5	Axis 5	Link 4	Link 5	0,0000	0,0000	0,0000	0,0000	0,0000	0,000001	0,000001	0,000001	0,000001
6	Joint 6	Axis 6	Axis 6	Link 6 (End effector)	0,0000	0,0000	0,0000	0,0000	0,0000	0,000001	0,000001	0,000001	0,000001

FIGURE 6.38: Screen capture of the (8) Joint Parameters worksheet of the parameters spreadsheet

The general structure and contents of the spreadsheet are listed in the subsequent TABLE 6.11.

TABLE 6.11: Structure and contents of the parameters spreadsheet

Sheet No.:	Sheet Name:	Content(s)/ Purpose(s):
1	(1) General Robot Information	Handling capacity, reach, weight
2	(2) Axis Range Limits	General (angular) axis limitations (A1-A6)
3	(3) Axis Speed Limits	General axis angular velocity limitations (A1-A6)
4	(4) Axis Acceleration Limits	General axis angular acceleration limitations (A1-A6)
5	(5) TCP Limits	General TCP velocity and acceleration limitations
6	(6) Joint Parameters	(Revolute) joint(s) parameters: State targets (position, velocity), internal mechanics (equilibrium pos., spring stiffn., damping coeff.), bearings (friction torques, damping coeff.)
7	(7) Motor Parameters	Electrical motor(s) (asynchronous machine (ASM) with squirrel cage rotor (three-phase)) parameters: El. ratings (power, voltage etc.), el. parameters (stator resistance, reactance, etc.) and mechanical parameters (rotor inertia, etc.)
8	(8) Transmission Parameters	Cycloidal transmission (gear box) parameters: teeth numbers (gear ratio), efficiency, inertia, etc.
9	(9) Motor Drivers Parameters	Six-pulse three phase converter parameters: DC link voltage, switching freq., sample time, etc.

The applied method allows convenient and centralized changes of any parameters without the application of changes to the simulation model block diagram. Furthermore, the spreadsheet can be easily modified or extended by adding further worksheets if required.

### 6.4.3 Simulation Model Parameterization

The parameterization of the Simulink/ Simulink Simscape model was implemented exclusively using the `robotPara` variable available from the MATLAB “base” Workspace. This is in accordance with the drafted conceptual design (section 5.2.3) and exemplarily shown for the ASM1 block of the base subsystem in FIGURE 6.39 below.

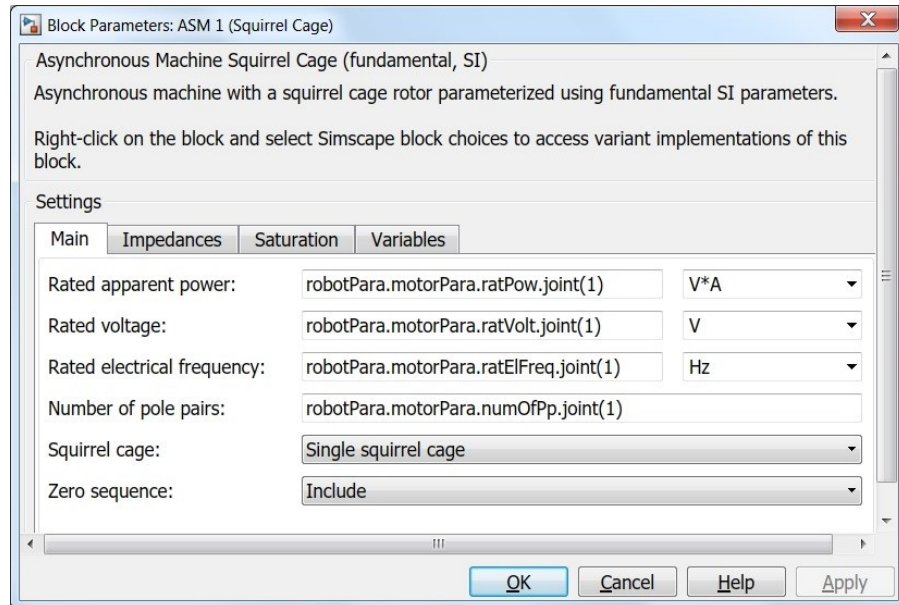


FIGURE 6.39: Screen capture of the ASM1 block parameterization

### 6.5 Data Set File Structure

The final general file structure of the simulation programs data set folder “BT\_ABB\_IRB\_2600\_Robot\_Sim.\_v\_A” is shown in the subsequent FIGURE 6.40.

The amount and types of the individual files of the corresponding subfolders are written in brackets behind/ below the individual related subfolder.

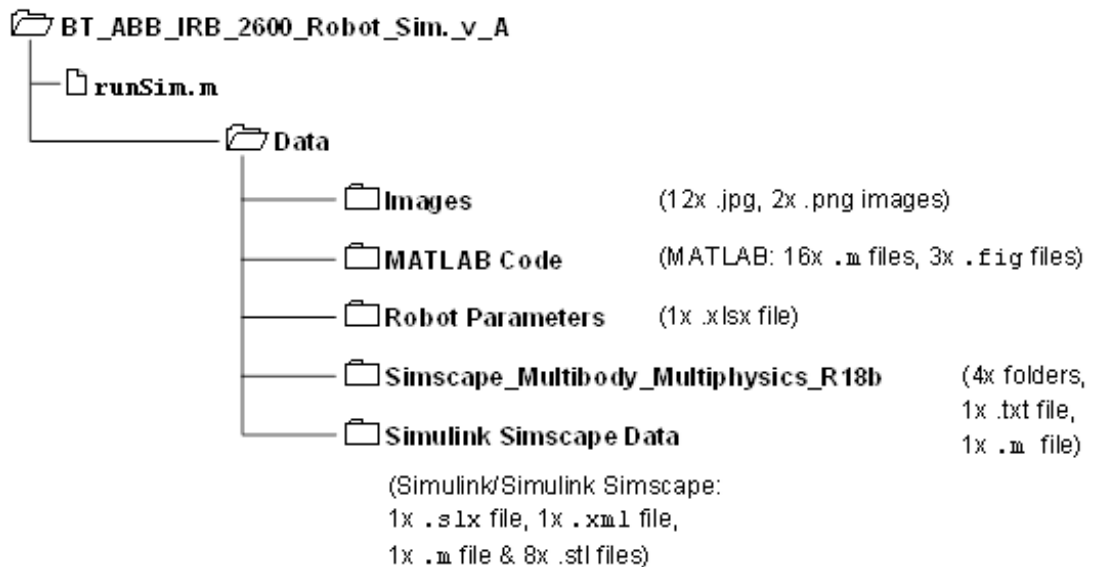


FIGURE 6.40: General file structure of the data set of the simulation model

## 6.6 Operating Manual

In contrary to the task planning of the project plan (Appendix 2. Project Plan), the creation of the operating manual was shifted to an earlier project state in order to provide guidance for the external testing of the preliminary version of the simulation program accomplished by the thesis supervisors and client(s). The operating manual consist of the sections Prerequisites, Introduction, Installation, Operation, Change of Parameters, CAD Model Update, Extensions/ Modifications and Troubleshooting and was designed as an independent document. Despite the fact that the operating manual mainly bases on and particularly refers to the thesis document at hand, it is highly recommended to also consider the contents of the operating manual due to some helpful contents of the manual are not covered by the thesis work document. Furthermore, it is highly recommended to refer to the operating manual before/ during the first use of the simulation model and also whenever errors occur (e.g. during compiling and/ or simulation (solving)).

The operating manual can be found from Appendix 5. Operating Manual.



## 6.7 Optional Tasks

The Optional Tasks section contains the documentation of the elaboration of two optional tasks performed in context of the accomplishment of the thesis work at hand. The accomplishments of the optional tasks are described sufficiently but narrowed to their main contents due to their optional character.

### 6.7.1 Simplified Joint Actuation Motor Model(s)

During testing and debugging of the simulation model (see section 7) equipped with AC asynchronous motor (ASM) models and drivers (version “A”, “BT\_ABB\_IRB\_2600\_Robot\_Sim.\_v\_A”), large computation times and high computational efforts for solving the model were revealed. With the help of several further tests, the high impact of the complexity of the AC ASM models and drivers on the solving time and required resources was investigated.

Based on that, a non-binding agreement of an optional task covering the creation of a second version of the simulation model ((version “B”, “BT\_ABB\_IRB\_2600\_Robot\_Sim.\_v\_B”) was made with the TAMK’s client in order to achieve:

- Decrease of the simulation model’s motor models and drivers complexities
- Decrease of the number of values required for the appropriate parameterization of the motor models and drivers
- Decrease of the computation time and computational efforts in the context of simulation model solving
- Increase of the comprehensibility of the simulation model by the reduction of the overall complexity

Due to the low overall complexity, the small number of required parameters, the high availability of required parameters from freely accessible datasheets, a decreased motor driver complexity and the typical usage for positioning tasks/applications, a (universal and ideal) DC motor type (block) was chosen for the accomplishment of the optional task.

Exemplarily, the “Robot Link 3 Subsystem” of the Simulink/ Simulink Simscape simulation model with the applied DC motor model and driver is shown below (FIGURE 6.41).

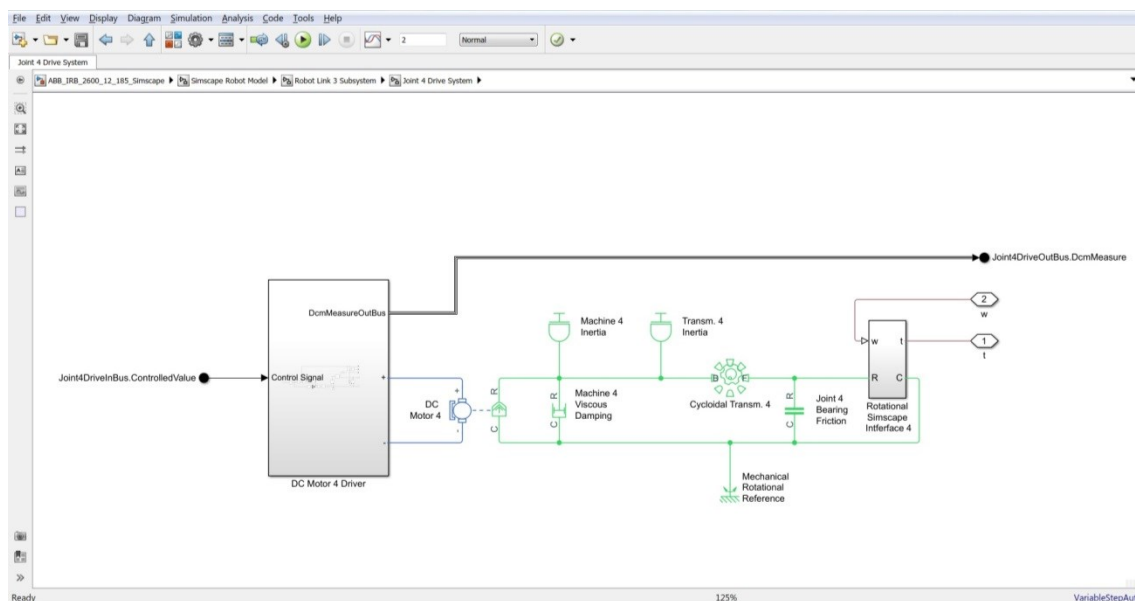


FIGURE 6.41: Simulink screen capture of the Robot Link 3 Subsystem with a DC motor model

In contrast to the AC ASM models, the DC motor models and drivers were directly parameterized with explicit values in the corresponding block parameter windows. The motor model parameters were gathered, compiled and if required, extrapolated from several datasheets (e.g. from ABB Motors and Mechanical Inc.<sup>14</sup>, mainly based on the power rating values of the substituted AC ASM models. The “DC Motor 4 Driver” sub subsystem of the “Robot Link 3 Subsystem” (FIGURE 6.41 above) is shown in the FIGURE 6.42 below.

<sup>14</sup> ABB Motors and Mechanical Inc., formerly “Baldor Electric Company” <http://www.motionusa.com.s3-website-us-east-1.amazonaws.com/baldor/BR1202-F.pdf> (Read on 02.04.2019)

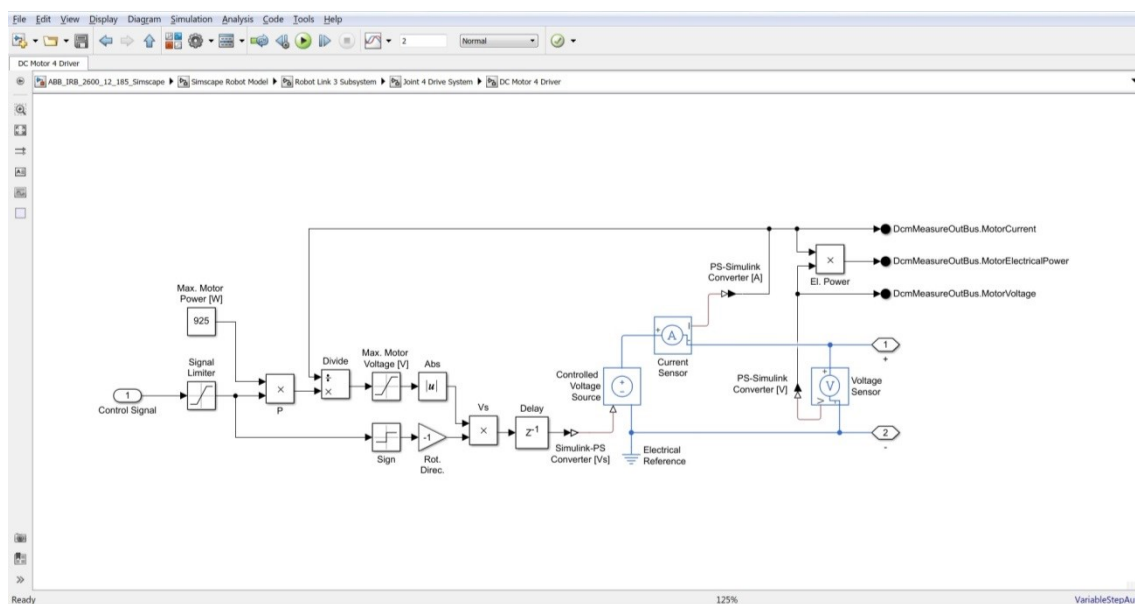


FIGURE 6.42: Simulink screen capture of the DC Motor 4 Driver subsystem

Despite the first impression of the appearance of the subsystem may not necessarily suggests a low complexity, the subsystem consists mainly of simple blocks (e.g. “Divide” or “Abs”). The function principle bases on the limitation of the applicable electric power (calculated from the set value and the maximum motor power). A controlled ideal voltage source draws any required but also measured current. If the maximum applicable electric power is exceeded, the voltage of the ideal voltage source is lowered to decrease the power the appropriate level. Equal to the driver of the AC ASM model, the DC motor driver was designed in a way to expect scalar input values within the range from +1 to -1. The application of DC motor models led to considerable decreases of the computational time and computational efforts for simulation model solving.

### 6.7.2 Virtual Identification Measurements

As the industrial robot manufacturer ABB and developer of the simulation software ABB RobotStudio claims that the simulated virtual robots behave very realistic and similar to their real counterparts (ABB Asea Brown Boveri Ltd. 2019f), the idea emerged to obtain missing Simulink Simscape simulation model parameters from virtual identification measurements conducted in the ABB RobotStudio simulation environment.

Therefore, introductory as well as more sophisticated ABB RobotStudio programming methods were investigated from ABB Asea Brown Boveri Ltd. (2019f) and ABB Asea Brown Boveri Ltd. (2019g) firstly.

In order to obtain missing frictional parameters of the simulation model, the subsequently presented procedure was developed and tested within the ABB RobotStudio 6.08 (license provided from TAMK) virtual environment:

Starting from the mathematical dynamic description of the robotic system (equation (4.14)):

$$\mathbf{M}(\mathbf{q})\ddot{\mathbf{q}} + \mathbf{C}(\mathbf{q}, \dot{\mathbf{q}})\dot{\mathbf{q}} + \mathbf{g}(\mathbf{q}) + \mathbf{f}(\dot{\mathbf{q}}) = \boldsymbol{\tau} \quad (6.5)$$

In order to extract frictional torques only related to the individual joint/ axis investigated, disturbing influences from all other axes and non-frictional torque sources need to be eliminated/ minimized. Therefore:

Reaching a state of constant velocity to eliminate/ minimize all moments of inertia:

$$\mathbf{M}(\mathbf{q}) \underbrace{\ddot{\mathbf{q}}}_{=0} = \mathbf{0} \quad (6.6)$$

Elimination/ minimization of centrifugal and Coriolis torques, e.g. by the alignment of the centers of masses of all moving masses with the investigated axis:

$$\mathbf{C}(\mathbf{q}, \dot{\mathbf{q}})\dot{\mathbf{q}} = \mathbf{0} \quad (6.7)$$

Elimination/ minimization of the influence of the gravitational acceleration, e.g. by the alignment of the investigated axis with the direction of the gravitational acceleration:

$$\mathbf{g}(\mathbf{q}) = \mathbf{0} \quad (6.8)$$

Equation (6.5) is now reduced to:

$$\mathbf{f}(\dot{\mathbf{q}}) = \boldsymbol{\tau} \quad (6.9)$$

Substituting the friction  $\mathbf{f}$  vector by a common static friction model considering viscous ( $\mathbf{F}_{m1}$ ) and Coulomb frictions ( $\mathbf{F}_{m2}$ ) ( $6 \times 6$ -dimensional diagonal matrices, containing the individual constant friction coefficients/ torques) (equation (4.20)):

$$\mathbf{f}(\dot{\mathbf{q}}) = \mathbf{F}_{m1} \dot{\mathbf{q}} + \mathbf{F}_{m2} \text{sign}(\dot{\mathbf{q}}) = \boldsymbol{\tau} \quad (6.10)$$

Exemplarily for the first robotic manipulator's axis (scalar expression):

$$F_{m1,1} \dot{q}_{1,\text{mot}} + F_{m2,1} \text{sign}(\dot{q}_{1,\text{mot}}) = \tau_{1,\text{mot}} \quad (6.11)$$

Conducting motor torque measurements at the time  $t_1$  for two different joint motion velocities, e.g. v50 and v100, using the RAPID<sup>15</sup> `GetMotorTorque()` function in the RobotStudio environment (motor side joint velocities can be read with the help of the `TestSignRead()` function):

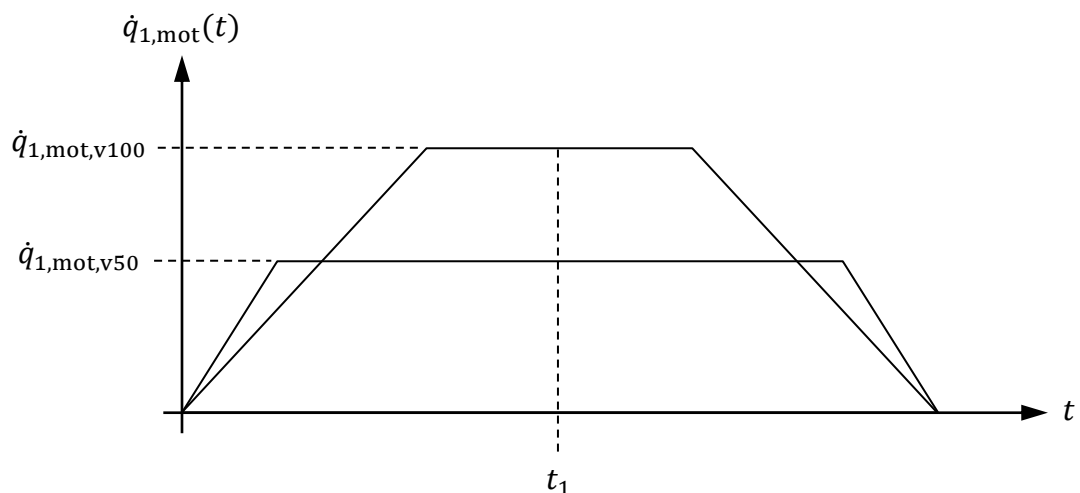


FIGURE 6.43: Exemplary plot of a trapezoidal joint velocity profile of the robot's first axis

<sup>15</sup> RAPID is a high-level programming language of ABB Asea Brown Boveri Ltd. for industrial robot programming

Applying the measurement results (explicit values):

$$\dot{q}_{1,\text{mot},v50}, \quad \dot{q}_{1,\text{mot},v100}, \quad \tau_{1,\text{mot},v50}, \quad \tau_{1,\text{mot},v100} \quad (6.12)$$

To the general common mathematical equation of linear functions:

$$y(x) = mx + b \quad (6.13)$$

Leads to the determination of the individual Coulomb friction torque:

$$F_{m2,1} = \frac{(\tau_{1,\text{mot},v50} \cdot \dot{q}_{1,\text{mot},v100}) - (\tau_{1,\text{mot},v100} \cdot \dot{q}_{1,\text{mot},v50})}{[(\dot{q}_{1,\text{mot},v100} \cdot \text{sign}(\dot{q}_{1,\text{mot},v50})) - (\dot{q}_{1,\text{mot},v50} \cdot \text{sign}(\dot{q}_{1,\text{mot},v100}))]} \stackrel{!}{\geq} 0 \text{ [Nm]} \quad (6.14)$$

And the individual viscous friction coefficient:

$$F_{m1,1} = \frac{\tau_{1,\text{mot},v100} - (F_{m2,1} \cdot \text{sign}(\dot{q}_{1,\text{mot},v100}))}{\dot{q}_{1,\text{mot},v100}} \stackrel{!}{\geq} 0 \text{ [Nms]} \quad (6.15)$$

The described procedure needs to be applied for each other axis individually. Furthermore, the described procedure does not cover any non-linear frictional effects and may suffer from inaccuracies due to superposition with other non-considered effects and changes of frictional values due to the individual robot's poses applied for the identification measurements.

The described procedure of virtual measurements for the identification of frictional coefficients and torques was tested only. Due to a lack of processing time at the end of the accomplishment of the thesis work, the identification was not finished satisfyingly. Therefore, parameter values not determined during the procedures described in section 6.4.1 were set zero or to very small values (e.g. 1E-12) in order to prevent "division-by-zero" errors in the context of numerical solving of the simulation model (e.g. Coulomb friction torques of the joint bearings).

Additional information concerning the topic of missing parameters/ incomplete parameterization are discussed in section 10.

## 7 TESTING AND DEBUGGING

The procedure of testing and debugging was accomplished constantly and parallel to every stage of the creation of the simulation model but more distinct during the stage of code programming. Generally, testing and debugging was mainly characterized by the trial-and-error method aiming at the validation of the investigated individual objects and all possible variations of objects interactions. Testing and debugging was mainly conducted in the MATLAB and/ or Simulink/ Simulink Simscape environment, partly with the help of the included debugging tools.

Parallel-to-creation testing revealed common typical programming bugs evoked by faulty copy-and-paste actions, typing mistakes, faulty indexing, etc.

In addition to the constant testing during the process of the creation of the simulation program, a separate short testing phase was accomplished after the finalization of a first preliminary version of the simulation program. Therefore, the preliminary simulation program version, along with a first version of the operating manual, was provided to the thesis supervisors and client(s) (external testing). Furthermore, testing was also executed by the author (internal testing).

In the case of internal testing, testing and debugging was divided into the sub-tasks of testing and debugging the MATLAB program part on one hand, and the Simulink/ Simulink Simscape program part on the other hand. As internal testing revealed a number of faults, only the most significant are listed exemplarily below:

MATLAB program part:

- Testing revealed a simple but grave unit conversion error within the MATLAB `inverse_kinematics.m` file. In the case of a linear tool movement, the initial robot manipulator's pose ( $\mathbf{q}_A$ ) needs to be derived from the starting point "A" workspace coordinates  $[x_A \ y_A \ z_A]$ , defined by a user input in the unit [mm], using the MATLAB inverse kinematics solver. The MATLAB kinematics solver expects coordinate inputs in the unit [m], but inputs were passed into the solver in the unit [mm].

This led to a conspicuously high level of computation time consumption and computational resources usage for the inverse kinematics solving. The rectification of the unit error caused a drastic decrease of the overall computation time and the usage of computational resources.

Simulink/ Simulink Simscape program part:

- Due to the non-existent appropriate control systems structures, the internal Simulink/ Simscape program testing part was narrowed to the validation of a small number of basic functions. In this context, rotational direction errors of the axes 1, 2, 5, and 6 were identified during observations of the simulation model's animation in the MATLAB Mechanics Explorer (deviations from the definition; section 3.3).

The errors were rectified within MATLAB `get_joint_move.m` file by the alignment of algebraic signs at the corresponding code lines.

At the time of the creation of this document, the results/ feedback concerning the external testing accomplished by the external parties were not provided to the author and thus neither recorded nor rectified.



## **8 OPERATION OF THE SIMULATION MODEL**

The document at hand is primarily meant for the documentation of the accomplished work during the progress of the execution of the bachelor's thesis. Furthermore, the allowed extent of the document is limited.

Therefore, no instructions and/ or further explanations concerning the operation of the simulation model were included in the thesis document itself but can be obtained from the earlier mentioned and comprehensive operating manual to be found from Appendix 5. Operating Manual.

## 9 CONCLUSION

The thesis work, recorded at the document at hand, aimed at the development and implementation of a MATLAB Simulink simulation model of an ABB IRB 2600-12/1.85 six axis articulated arm industrial robot for the purpose of educational use in control system design.

The created simulation model, in its recent state, is considered as a comprehensive and fully functional application that meets the requirements, covers optional accomplished tasks and can be used for the educational purposes it was initially meant for, as it:

- Bases on a Simulink Simscape Multibody simulation model derived from the specific industrial robot's CAD model
- Is in accordance with the main technical specifications of the real industrial robot (axes definitions and limitations, frame definitions, etc.)
- Is in accordance with the common and generally accepted robotic manipulators theory (e.g. DH-formalism)
- Covers a fully kinematic robot model
- Covers a common dynamic robot model considering gravitational acceleration and a linear friction model (viscous and Coulomb frictions)
- Contains detailed and realistic joint actuation models (joint actuation motor types, motor drivers, gearbox types, driveline characteristics, etc.)
- Shows a simplified and assumption based but realistic first parametrization, covering the manipulator's links mass and inertia properties (masses, CoM, Mol, Pol), motor models (electrical and mechanical characteristics) and gearboxes (ratios, inertias, efficiencies)
- Bases on well-documented, sufficiently commented and modular MATLAB codes
- Contains convenient, descriptive and input filtering GUI

- Allows changing, modifying and extending the simulation model's:
  - CAD model
  - Simulink Simscape Multibody model (block diagram)
  - Simulink model (block diagram)
  - MATLAB programs
  - Parameterization
- Covers an additional second simulation model version (Version "v\_B") with simplified DC joint actuation models/ subsystems
- Comes with a comprehensive operating manual covering instructions for the operation, update and change/ modification of the simulation program/ model
- Provides information for future project continuations like a method for virtual identification measurements to obtain frictional parameters from the ABB RobotStudio software

And:

- Provides a ready-to-use control system structures design environment also covering simple predetermined PID controllers for testing purposes
- Allows appropriate observation, recording, storage and export of the simulation results
- Allows comparisons of the Simulink/ Simulink Simscape simulation results to other simulation/ measuring results gained from other sources

Nevertheless, the simulation model suffers major incompleteness and weaknesses such as:

- Incomplete and/ or simplified and/ or estimated and/ or assumption based parametrization
- Not performed simulation model validation due to the lack of appropriate control system structures

Furthermore, the capabilities of the simulation model are limited due to the applied general simplifications and restrictions (section 3.7):

- Ideal rigid bodies such as links, joints, shafts, transmission gears, belts etc.
- A linear friction model
- Missing consideration of backlashes and uncertainties (bearings and transmissions)
- Generally neglected time delays
- Missing consideration of external (secondary) payloads like the end effector supply wiring
- Generally idealised simulation model's elements representations, limited to the level of detail provided by the corresponding Simulink/ Simulink Simscape blocks
- Generally neglected thermal effects (e.g. temperature dependent transmission lubricant viscosity)

## 10 OUTLOOK

Based on the statements related to the incompleteness, weaknesses and limited capabilities of the simulation model made in the conclusion (section 9), continuations of this thesis work are required in order to obtain a completely comprehensive and more accurate simulation model.

Following this, pending future tasks, accomplished in the context of further thesis works, semester projects, in-lecture projects, homework, laboratory works etc., can be coarsely divided into three categories and named as:

Completion (of the parameterization) of the simulation model:

- Acquisition and implementation of more precise information concerning link masses and inertias, motor-, gear/ transmission- and revolute joint parameters such as inertias, electrical parameters, damping/ friction values, gear ratios, etc. from existing data sources and/ or virtual or real (identification-) measurements.

Furthermore, the validation of the simulation model. This could also cover measurements for comparisons between the MATLAB Simulink simulation model, other simulation models and the real robotic system.

In the case of future accomplishment of identification measurements on the virtual or real robotic system, the consideration of Al-Dois, Jha & Mishra (2013) and Verdonck, Swevers & De Schutter (2007) is recommended.

#### Modification of the simulation model:

- Adaption or change of the contemporary applied motor types and their drives in order to reduce the complexity of the simulation model and/ or lessen the computational efforts (e.g. by the modification of the simplified joint actuation motor models presented in section 6.7.1).

Modifications/ changes of the manipulator's CAD model, e.g. CAD assembly constraints, link geometries or the end effector/ tool.

#### Extension of the simulation model:

- Implementation of further simulation model block diagrams and corresponding parameters in order to reduce the number of general simplifications and restrictions.

Implementation of additional motion types such as point-to-point or circular movements. This could also cover the extension of available/ applied velocity and/ or velocity/ acceleration profiles (e.g. S-Curve velocity profile) in the context of motion planning.

Extensions of the solver constraints of the MATLAB inverse kinematics solver `gik()` function are already prepared in the comments of the corresponding MATLAB file and can be applied.

Extensions of the end effectors/ tools capabilities within the coverage of applicable Simulink Simscape domains like pneumatics and hydraulics.

Application of secondary payloads connected to the manipulator's links, e.g. added to CAD model.

The creation of a manipulator's environment within the Simulink simulation model with the help of additional geometries (e.g. a workbench).

## REFERENCES

ABB Asea Brown Boveri Ltd. 2019a. Industrial Robots. IRB 2600. Printed on 19.03.2019. <https://new.abb.com/products/robotics/industrial-robots/irb-2600>

Hyyppä, A. 2015. Älykäs Huuva. Kone-ja tuotantotekniikka Kone-ja laiteautomaatio. Tampereen ammattikorkeakoulu. Opinnäytetyö. Read on 27.02.2019. <http://urn.fi/URN:NBN:fi:amk-2015121520886>

Rodewald, V. 2016. Design of an Intelligent Protection Shield. Mechanical and Production Engineering. Tampere University of Applied Sciences. Bachelor's thesis. Read on 27.02.2019. <http://urn.fi/URN:NBN:fi:amk-2016061312819>

Gerland, B. 2017. Designing and implementing a Robot Gripper using additive manufacturing. Mechanical and Production Engineering. Tampere University of Applied Sciences. Bachelor's thesis. Read on 27.02.2019. <http://urn.fi/URN:NBN:fi:amk-201703273754>

Compton, Z. 2018. Creation of an Augmented Reality App for an Introduction to Industrial Machine Mechanics. Interactive Media. Tampere University of Applied Sciences. Bachelor's thesis. Read on 27.02.2019. <http://urn.fi/URN:NBN:fi:amk-2018053111731>

ABB Asea Brown Boveri Ltd. 2019b. Industrial Robots. IRB 2600. Technical specification - IRB 2600, M2004, Product specification. Printed on 16.02.2019. <https://search-ext.abb.com/library/Download.aspx?DocumentID=3HAC035959-001&LanguageCode=en&DocumentPartId=&Action=Launch>

ABB Asea Brown Boveri Ltd. 2019c. Operating Manual. RobotStudio. 5.14. Document ID: 3HAC032104-001. Revision: E. Printed on 18.02.2019. <https://library.e.abb.com/public/4b4d0a7f1e14fcdac1257c13004f1121/3HAC032104-en.pdf>

Weber, W. 2017. Industrieroboter. Methoden der Steuerung und Regelung. 3rd edition. München: Carl Hanser Verlag München. Printed on 18.02.2019. <https://doi.org/10.3139/9783446435780>

Kelly, R., Santibáñez, V. & Loría., A. 2005. Control of Robot Manipulators in Joint Space. 1st edition. London: Springer-Verlag London Limited. Printed on 21.02.2019. <https://doi.org/10.1007/b135572>

Siciliano, B. & Khatib, O. 2008. Springer Handbook of Robotics. 1st edition. Berlin Heidelberg: Springer-Verlag Berlin Heidelberg. Printed on 21.02.2019. <https://doi.org/10.1007/978-3-540-30301-5>

Siciliano, B., Sciavicco, L., Villani, L. & Oriolo, G. 2009. Robotics. Modelling, Planning and Control. 1st edition. London: Springer-Verlag London Limited. Printed on 21.02.2019. <https://doi.org/10.1007/978-1-84628-642-1>

Bajd, T., Mihelj, M., Lenarčič, J., Stanovnik, A. & Munih, M. 2010. Robotics. 1st edition. Dordrecht: Springer Dordrecht. Printed on 06.04.2019. <https://doi.org/10.1007/978-90-481-3776-3>

Grote, K. H., Bender, B. & Göhlich, D. 2018. Dubbel. Taschenbuch für den Maschinenbau. 25th edition. Berlin: Springer-Verlag GmbH Deutschland. Printed on 06.04.2019. <https://doi.org/10.1007/978-3-662-54805-9>

Pietruszka, W. D. 2014. MATLAB® und Simulink® in der Ingenieurpraxis. Modellbildung, Berechnung und Simulation. 4th edition. Wiesbaden: Springer Fachmedien Wiesbaden. Printed on 18.02.2019. <https://doi.org/10.1007/978-3-658-06420-4>

The MathWorks Inc. 2019a. Products. Overview: Simscape Multibody - Model and simulate multibody mechanical systems. Read on 03.03.2019. <https://www.mathworks.com/products/simmechanics.html>

The MathWorks Inc. 2019b. Support. Documentation: MATLAB - The Language of Technical Computing. Release: R2018b. Printed on 16.02.2019. <https://www.mathworks.com/help/matlab/index.html>

The MathWorks Inc. 2019c. Support. Documentation: Simulink - Simulation and Model-Based Design. Release: R2018b. Printed on 16.02.2019. <https://www.mathworks.com/help/simulink/index.html>

The MathWorks Inc. 2019d. Support. Documentation: Simscape - Model and simulate multidomain physical systems. Release: R2018b. Printed on 16.02.2019. <https://www.mathworks.com/help/phymod/simscape/index.html>

Glöckler, M. 2018. Simulation mechatronischer Systeme. Grundlagen und Beispiele für MATLAB® und Simulink®. 2nd edition. Wiesbaden: Springer Fachmedien Wiesbaden. Printed on 16.02.2019. <https://doi.org/10.1007/978-3-658-20703-8>

Nabtesco Corporation. 2019. Product. Component. RV-N. Read on 06.04.2019. <https://precision.nabtesco.com/en/products/detail/RV-N>

Nabtesco Corporation. 2015. Precision Reduction Gear RV. N series. Technical Information. Read on 06.04.2019. [https://www.nabtesco.de/fileadmin/user\\_upload/Downloads/Produkt\\_Brosch%C3%BCren/RV-N\\_Rev.9\\_EN\\_no\\_drw\\_nabtesco.pdf](https://www.nabtesco.de/fileadmin/user_upload/Downloads/Produkt_Brosch%C3%BCren/RV-N_Rev.9_EN_no_drw_nabtesco.pdf)

ABB Asea Brown Boveri Ltd. 2019d. Products. 3HAC030216-003. Detailed information for: 3HAC030216-003. Read on 06.04.2019. <https://new.abb.com/products/3HAC030216-003/rot-ac-motor-with-pinion>

TAMAGAWA SEIKI Co., Ltd. 2019. Products. Servo Motors. TBL-iIV Series. Catalog. AC Servomotors (TBL-iIV Series). Read on 06.04.2019. [https://www.tamagawa-seiki.com/assets/img/downloads/pdf/servomotor/1699N2EJ\\_shusei.pdf](https://www.tamagawa-seiki.com/assets/img/downloads/pdf/servomotor/1699N2EJ_shusei.pdf)



ABB Asea Brown Boveri Ltd. 2019e. Robotics. RobotStudio. Read on 02.04.2019. <https://new.abb.com/products/robotics/robotstudio>

ABB Asea Brown Boveri Ltd. 2019f. RobotStudio. Getting started: Tutorials for RobotStudio. Read on 17.02.2019. <https://new.abb.com/products/robotics/robotstudio/how-to-use-it/getting-started>

ABB Asea Brown Boveri Ltd. 2019g. RobotStudio. Tutorials for RobotStudio - The world's most used offline programming tool for robotics. Read on 18.02.2019. <https://new.abb.com/products/robotics/robotstudio/tutorials>

Al-Dois, H., Jha, A. K. & Mishra, R. B. 2013. Application of Industrial Robots for Producing Cores in a Foundry: Task Time Optimization. Journal of Applied Science and Engineering 16 (2), 177-186. Printed on 23.02.2019. <https://doi.org/10.6180/jase.2013.16.2.09>

Verdonck, W., Swevers, J. & De Schutter, J. 2007. Dynamic model identification for industrial robots. An integrated experiment design and parameter estimation approach. IEEE control systems 27 (5), 58-71. Printed on 23.02.2019. <https://doi.org/10.1109/MCS.2007.904659>

Virikko, H. & Lamminsivu, R. 2017. Report Guide. Tampere University of Applied Sciences. Tampere. Printed on 10.02.2019. <https://intra.tamk.fi/documents/67978/1644670/Report+guide+2018.pdf/ece93217-5f76-428a-b829-5c2c219f3de3>

Diersen, P. 2017a. Dokumentationsrichtlinien für Bachelor/Master-Arbeiten, Hausarbeiten und Projektberichte. Hannover University of Applied Sciences and Arts. Hannover

Diersen, P. 2017b. Zitierhinweise für Hausarbeiten, Abschlussarbeiten und Projektdokumentationen. Hannover University of Applied Sciences and Arts. Hannover

Diersen, P. 2017c. Konstruktionslehre 1. Lecture script. Konstruktionslehre 1 lecture summer term 2017. Hannover University of Applied Sciences and Arts. Hannover

**DECLARATION OF AUTHORSHIP**

I hereby confirm that this bachelor's thesis at hand is entirely my own work and that I have not used any additional assistance or resources other than indicated. All quotations, paraphrases, information and ideas that have been taken from other sources (including the Internet as well as other electronic sources) and other person's work have been cited appropriately and provided with the corresponding bibliographical references. The same is true of all drawings, sketches, pictures and other illustrations that appear in the text. I'm aware that the neglect to indicate the used sources is considered as fraud and plagiarism in which case sanctions are imposed that can lead to the suspension or permanent expulsion of students in serious cases.

---

Olivier Preuss

Tampere, 16.04.2019

**APPENDICES**

Appendix 1. Thesis Contract.....	130
Appendix 2. Project Plan.....	135
Appendix 3. List of Requirements .....	137
Appendix 4. Program Flow Charts .....	141
Appendix 5. Operating Manual .....	159

## Appendix 1. Thesis Contract

1 (5)



1 (5)

**THESIS CONTRACT: Simulation Model for a Six Axis Articulated Arm Industrial Robot**

The thesis contract has to be done in triplicate. One original is given to the company/community, one to the student/students, and one to Tampere University of Applied Sciences.

Thesis Author(s)		
Name <b>Olivier Preuss</b>	Address	
Email	Telephone	Student number
Degree Programme <b>KV-vaihdon koulutusohjelma/Tekniikan ja liikenteen ala</b>	Study path	

Tampere University of Applied Sciences / TAMK		
Address <b>Kuntokatu 3, 33520 Tampere</b>		Telephone
Thesis Supervisor		
Name <b>Sami Hämäläinen</b>	Email	Telephone
Head of Degree Programme, Supervising Teacher or other TAMK Representative (when needed, see thesis terms)		
Name <b>Markus Aho</b>	Email	Telephone

Bachelor's or Master's Thesis	
Thesis topic/title <b>Simulation Model for a Six Axis Articulated Arm Industrial Robot</b>	Planned completion year of Thesis <b>2019</b>

*Thesis objectives*

Software Usage:

- Operating System: Microsoft Windows 7 and higher (Requirement)
- CAD Software: SolidWorks 2001Plus and higher

OR

- Wildfire, Creo (formerly Pro/ENGINEER) WildFire 2.0 and higher, Creo 1.0 and higher

OR

- Autodesk Inventor 2009 and higher (Requirement)

Simulation Software:

- MATLAB R2018b (Requirement)

Other:

- ABB RobotStudio 6.08 (Optional)

Robot Type:

- ABB IRB 2600-12/1.85 industrial robot (Requirement)

Other Objectives:

- If not defined divergent, the simulation model is based on the SI base units and derived units. (Requirement)
- If not defined divergent, the simulation model is based on the MATLAB programming language (text based) which also covers the Simulink and Simscape programming languages (graphical). Programme code always needs to be commented in a short and concise way. Predefined MATLAB contents like toolboxes, classes, functions, blocks, etc. have to be preferred and used whenever available to accomplish a task. (Requirement)
- As minimum requirement, the interaction between the user and the simulation model is realised via the MATLAB command window in the MATLAB programming language. The creation of a (graphical) user interface for controlling the simulation model is optional. If implemented, the design of the (graphical) user interface is to be determined by the processor but kept simple and intuitive. (Requirement)
- The simulation model includes a physical and graphical representation of the robot based on a CAD model with a sufficient precision implemented via Simulink Simscape. (Requirement)
- The simulation model represents the real robotic system with all its properties such as geometry and dimensions, weights, inertias, frictions, drives behaviours etc. as sufficient as an appropriate effort of the acquisition of the properties allows. In this context, appropriate effort does not cover any measurements taken from the real robotic system in order to obtain simulation system parameters. Required data are acquired from product specifications, datasheets, manuals, technical drawings, software sources, literature, third parties, etc.. In case of not obtainable data, simplifications and assumptions are allowed but are clearly revealed and founded in a sufficient way. (Requirement)
- The simulation model uses the BASE coordinate system as its main coordinate system. In accordance with the common definition, the BASE coordinate system is a right-handed Cartesian coordinate system, whereby the z-axis coincides with the first robot's axis and the x-y-plane coincides with the set-up area of the base of the robot. The BASE coordinate system and the WORLD coordinate system coincide and represent a reference coordinate system which acts as reference for target definitions and end effector orientations. The home configuration of the simulation model is in accordance with the home configuration of the real robotic system defined in the technical specifications available from the robot manufacturer. This accordance is also valid for axis designations, initial angular positions, angular limitations and directions of rotation (signs). (Requirement)
- The simulation model includes a linear path planning which enables a defined movement of the robot model from coordinate "A" to coordinate "B" specified by the user in the reference coordinate system. Coordinates are inserted in the format [x y z], (e.g. [100 300 50]) in millimetres [mm]. Furthermore, the values of the angles of every single rotational joint of the robot model can be directly given into the simulation model by the user in order to act as set point values for the control system structures. Input angle values are inserted in the format [q1 q2 q3 q4 q5 q6], (e.g. [0 0 90 60 0 30]), angular degree [°], and counted absolute from the initial position. The insert format of the orientation of the end effector is

[ $\alpha$   $\beta$   $\gamma$ ], (e.g. [0 0 45]), angular degree [°], in accordance with the common definition of the ZYX Euler angle format. (Requirement)

- The structure of the simulation model is created in a modular way. Maintaining, editing, updating and extending the model is possible with moderate effort. Moderate effort is considered as effort that can be managed by a professional and the information provided in the thesis document, short instruction document and program comments. The program flow and the operation of the model are designed clearly structured. (Requirement)
- The implementation of the simulation model is created in way that allows a comparison between simulation results gained from the MATLAB Simulink simulation and measurements taken from the real robot system and/or the ABB RobotStudio software, especially the values of the single joint angles. (Optional)

Creation of a short concise instruction document for the operation and service of the simulation model. (Requirement)

*Thesis purpose*

The purpose of the thesis is the development of a MATLAB Simulink simulation model of an ABB IRB 2600 six axis articulated arm industrial robot for educational use in control system design. The simulation model shall be used as ready-to-use environment for control system structures designed and implemented by pupils/students. Thus, the simulation model includes all components to execute, monitor and record kinematic and dynamic simulations of the robot, except the controller structures themselves.

*Short description of thesis implementation and timetable*

The thesis will be accomplished by:

1. The determination of all involved parties, the determination of the detailed tasks, requirements and scope and the formation of a contract and registration of the thesis.
2. The processor is responsible for the independent and correct preparation, accomplishment, monitoring and completion of the thesis. Furthermore the processor acts accordingly to processes described in the attached project plan/schedule as close as possible. Additionally, the processor will correspond with and/or report to the determined supervisors whenever major changes and/or necessities occur. The accomplished work is recorded by the processor and the basis for the Bachelor's Thesis document which is created independently and according to the TAMK's Thesis Guidelines.
3. A submission of preliminary contents such as documents and programs to the responsible supervisors/parties for review purposes is desirable. The final thesis document will be submitted via E-Mail in WORD format on schedule by the processor. The implementation will be completed by a maturity test accomplished by the processor and an evaluation of the thesis by the responsible parties.

Timetable: See thesis contract appendices.

### Thesis expenses, Company's or Organisation's supervision and Copyright

*Specification of thesis expenses and agreement on expenses (see thesis terms)*

None.

*Company or organisation representative's role in thesis process (e.g. supervision and tools, see thesis terms)*

Name: Ville Jouppila

*Copyright (see thesis terms)*

No exceptions from the TAMK thesis terms.

### Agreement on Thesis reporting

*Thesis reporting and publishing (see thesis terms)*

In accordance with the TAMK thesis terms.

*Thesis presentation*

In accordance with the TAMK thesis terms.

*Company's/organisation's feedback on thesis (see thesis terms)*

In accordance with the TAMK thesis terms.

The contract parties confirm the above-mentioned thesis details and approve the enclosed thesis terms with their signatures.

Student's signature(s)	
Date	
Student's signature	Print name <b>Olivier Preuss</b>
Supervising Teacher's or other TAMK Representative's signature(s)	
Date 25.03.2013	
Supervising Teacher's signature	Print name <b>Sami Hämäläinen</b>
Supervising Teacher's or other TAMK Representative's signature	Print name <b>Markus Aho</b>

Your information is saved in Oiva. More information: Privacy and Terms of Use <https://oiva.tamk.fi/terms/list>

**APPENDIX 1****TAMPERE UNIVERSITY OF APPLIED SCIENCES' GENERAL THESIS TERMS****(Terms updated on 1 November 2018)**

Tampere University of Applied Sciences (TAMK) appoints a supervisor to the student for the thesis process and strives to support the student at the various stages of the thesis process by providing supervision within the degree programme and studies which support the thesis process. TAMK or the thesis supervisor cannot however be held responsible for the quality, content, completion or possible delays of the thesis. The university of applied sciences or its teachers do not have consulting responsibility regarding the thesis contents or implementation to the cooperation party.

Students may discontinue the thesis project and terminate this contract by notifying of it in written to both the university of applied sciences and the cooperation party. If the student terminates this contract, the university of applied sciences and cooperation party agree on together if it is still possible to implement the thesis project for example by changing the project timetable and contents and having a new student to implement the project or if the thesis project has to be discontinued. All changes in this contract and termination of the contract have to be made in written between the university of applied sciences and cooperation partner. If the student terminates the contract, it does not cause the student or university of applied sciences any liability for damages.

The thesis supervisor and other TAMK staff members are bound by professional secrecy and prohibition of use regarding the student's thesis idea papers, thesis plans, research data and all classified information during and after the thesis process (621/1999: 23§ and 24§).

The working-life representative appoints a contact person for the thesis process and aims at contributing to the advancement of the work by offering the student supervision and information needed to achieve the objectives. Furthermore, the working-life representative will be responsible for agreed thesis expenses, such as postage, fares or other comparable costs. Thesis projects made in working life cooperation may not cause any extra costs for the university of applied sciences. The cooperation party has the obligation to pay the university of applied sciences costs caused by customising the project to the cooperation party's needs or otherwise caused by the cooperation party.

The head of degree programme or another TAMK representative is given on the thesis contract if necessary. The head of degree programme's approval is needed if the thesis project for example uses TAMK's educational supplies, teaching facilities or other resources to a significant degree (the matter is first discussed with the thesis supervisor). There is no need for filling in data if TAMK facilities or computers are used for the thesis process or report writing.

If the cooperation partner requires application of a research permit (for example theses in the field of health care and social services), students follow its guidelines. Students do not need to construct both the thesis contract and research permit.

A thesis report is a public record (Constitution of Finland 731/1999, Act of Openness of Government Activities 621/1999) and it should not contain any classified information. Thesis reports made at Tampere University of Applied Sciences are public. The publicity guarantees objective and fair thesis assessment.

TAMK students are not allowed to write an entirely classified thesis (AOA 2457/4/13, 14.4.2014). In case there is a need to include classified information in the thesis process, it must be carefully considered. Classified information may not be included in the thesis report to be assessed. The thesis report alternatives available are negotiated between the supervisor, student, and working-life representative already at the beginning of the thesis process.

Students commit themselves to keep all confidential information confidential and not to use it for any other purpose than the thesis in accordance with this contract. Students take care of not including any confidential or classified information in the thesis to be published. Students may not disclose confidential information to the university of applied sciences, its supervisors or third parties without the cooperation partner's specific permission.

The thesis author has the right to determine the means of publication for his/her thesis (Copyright Act 404/1961, 2§). The thesis report is either published as a paper version in TAMK library or electronically via Theseus (<https://www.theseus.fi/>). Students take care of not including any confidential or classified information in the thesis to be published. Students may not disclose confidential information to the university of applied sciences, its supervisors or third parties without the cooperation partner's specific permission. If the thesis includes confidential information, the working-life partner has to provide written feedback to support the thesis assessment.

The copyright of the thesis principally belongs to the student. This does not restrict the working-life representative's right to benefit from the knowledge and potential development suggestions included in the thesis. In case the thesis or its appendices include e.g. a separate manual, educational material, software or programming work, visual material, drawings, audio or video data or other equivalent material that the working-life representative needs to utilise in his/her work in a copyright-wise relevant manner, transfer of restricted rights (use, editing and distribution) should also be agreed on separately between the working-life representative and student.

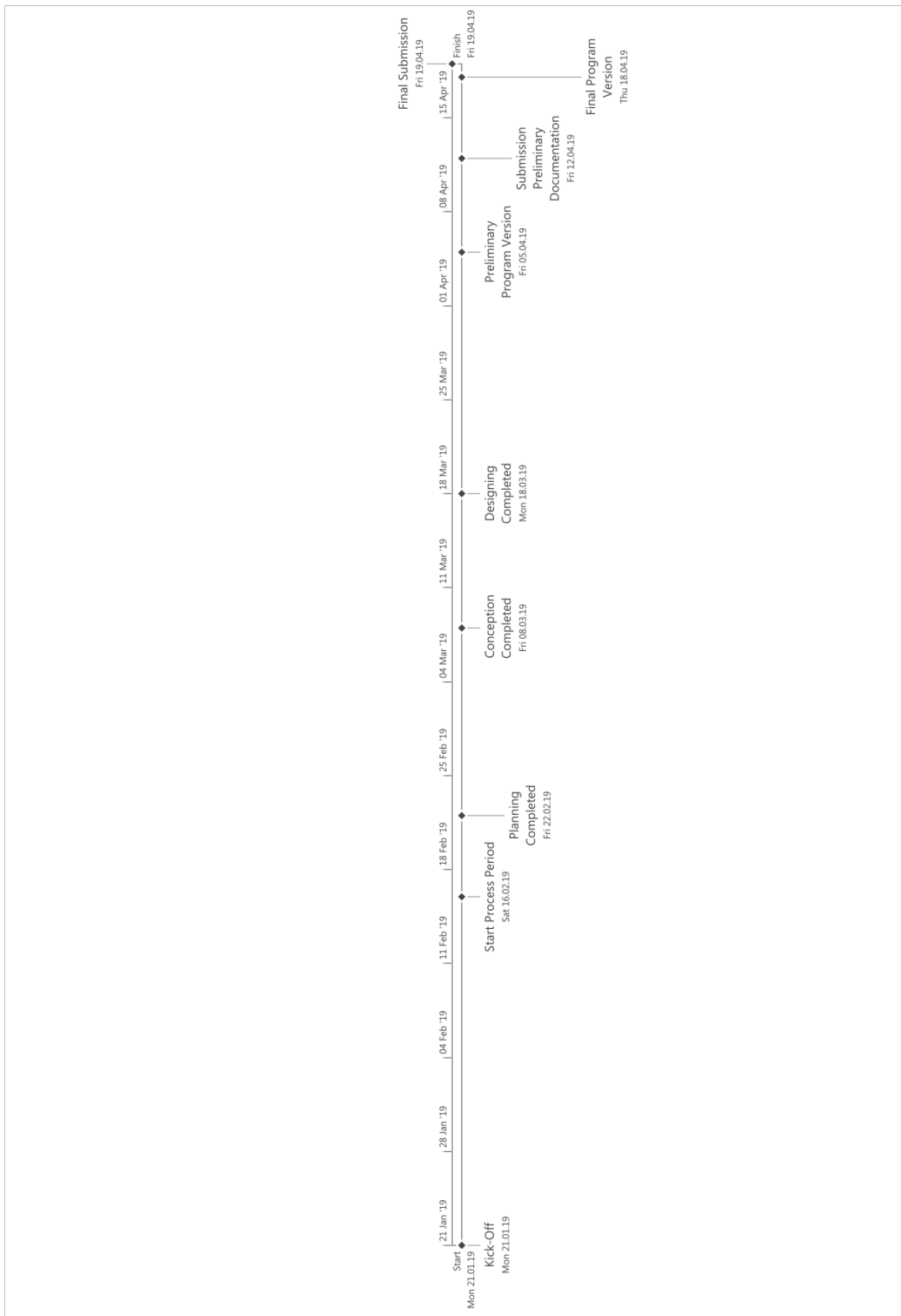
The cooperation party is aware of that the thesis forms a part of the student's studies. The cooperation party understands that as a rule the student is not a professional of the field and that the thesis is not necessarily applicable for the cooperation party's use. The cooperation party assigns background material for the project, such as software, at its own risk. The student and university of applied sciences are not in charge of decrease or destruction of the cooperation party's background material during the thesis project. The contract parties are not liable to one another for potential damages caused by terminating the contract.

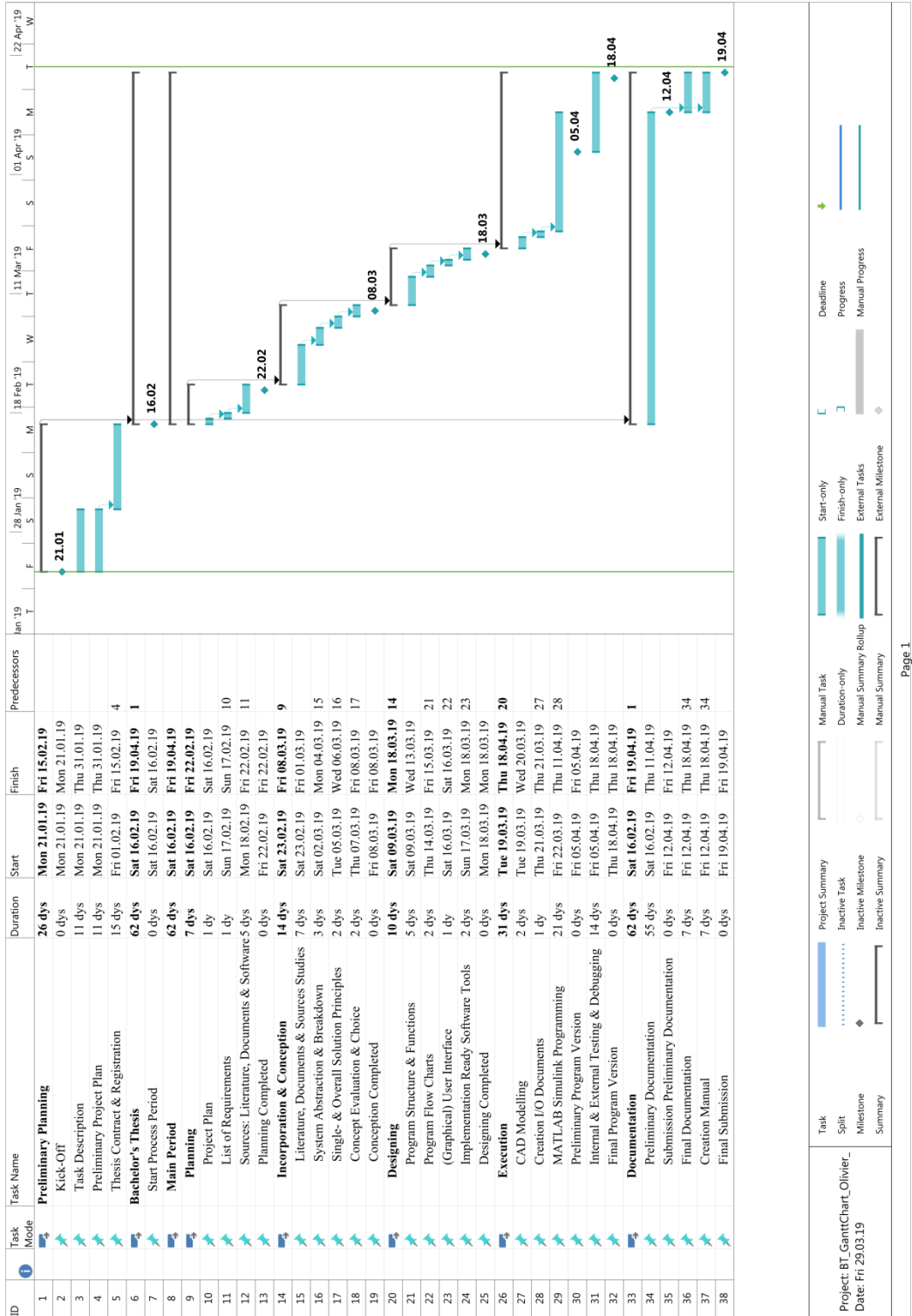
Thesis reports are permanently archived records based on Arkistolaitos' decision (AL/2897/07.01.01.03.01.2014, 18.9.2014).



# Appendix 2. Project Plan

1 (2)





Appendix 3. List of Requirements

List of Requirements			
First Issued: 16.02.2018 (By First Editor)	<b>Project:</b> Bachelor's Thesis: Simulation Model for a Six Axis Articulated Arm Industrial Robot		Client(s): Tampere University of Applied Sciences (TAMK) (Representative: Mr. Ville Jouppila)
	Type: (R = Requirement O = Optional)	No.:	Editor(s): Olivier Preuss (OP)
		Description of Requirement:	Value/Data:
<b>Related Product</b>			
R	RE1	Industrial robot manipulator	ABB IRB 2600-12/1.85 OP
<b>Units</b>			
R	UN1	Units to be applied within the complete	SI base units and derived units OP
R	UN2	Deviations from UN1	Sufficiently described and justified within the documentation (DO2) OP
<b>General Definitions (Manipulator Related)</b>			
R	GE1	Coordinate systems (frames)	Right-handed Cartesian only OP
R	GE2	Base coordinate system	In accordance with the manufacturers definitions OP
R	GE3	World coordinate system	Compliant to GE2 OP
R	GE4	Reference coordinate system	
R	GE5	Tool/end effector coordinate system	In accordance with the manufacturers definitions OP
R		Tool/end effector orientation description	User input: ZYX-EULER angles [ $\alpha$ $\beta$ $\gamma$ ], reference: Base coordinate system (GE2) OP
R	GE6	Home position	In accordance with the manufacturers definitions OP
R	GE7	Axis definitions and designations (labelling, rotational directions, limitations, etc.)	
Approved: OP		Date: 29.03.2019	Page: 1 of 4

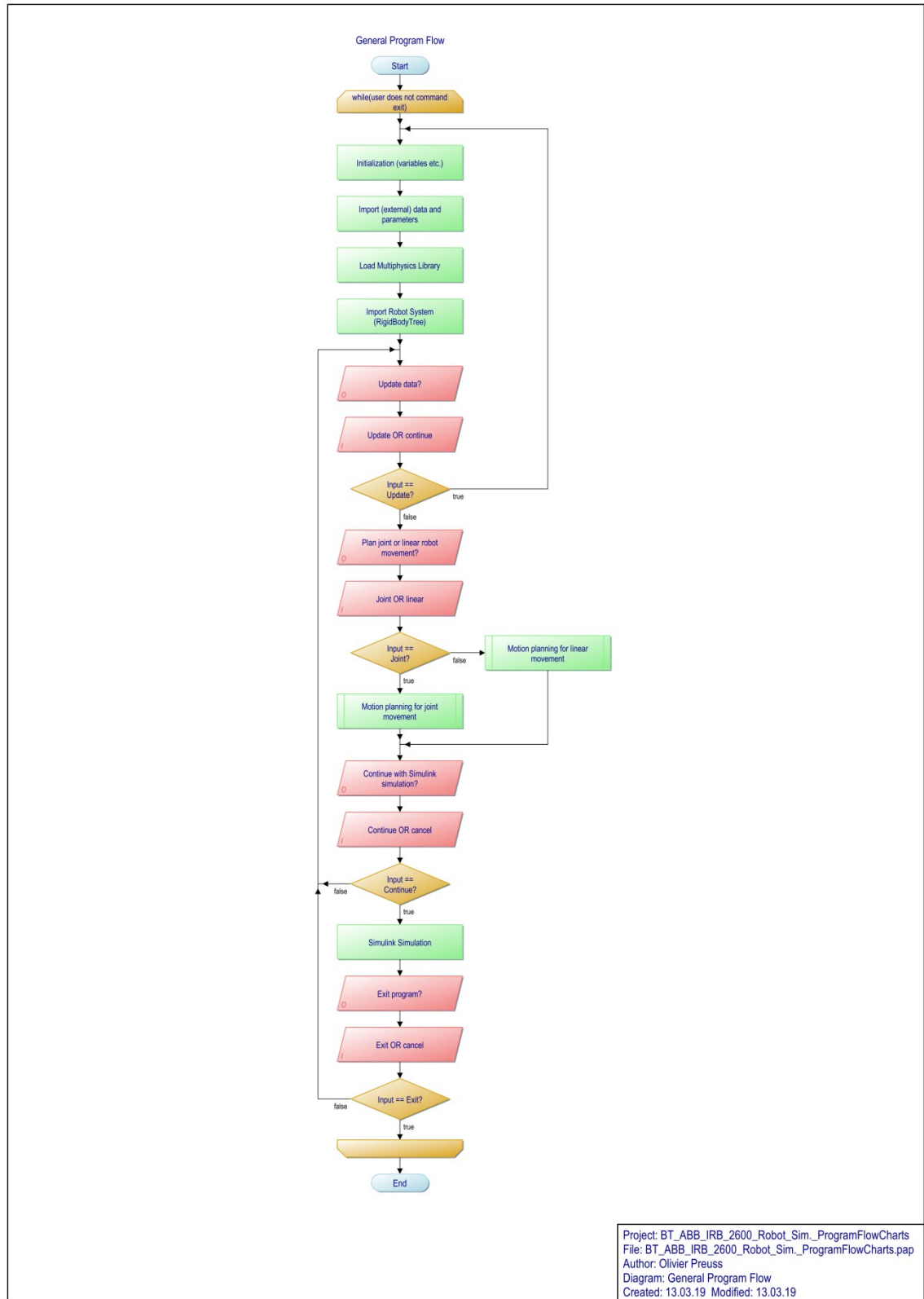
List of Requirements (Continuation)				
Type: (R = Requirement O = Optional)	No.:	Description of Requirement:	Value/Data:	Responsibility:
<b>Simulation Model</b>				
R	SI1	Simulation model implementation	MATLAB Simulink/ MATLAB Simulink Simscape (Multibody)	OP
R	SI2	Kinematic model	Yes (DH-formalism)	OP
R	SI3	Dynamic model	Yes (comprehensive, incl. gravitational acceleration)	OP
R	SI4	Graphical model/ animation	Yes, CAD model based	OP
R	SI5	Control system structures	Environment for the implementation of control system structures: Provision only	OP
R	SI6	Signal measurements	Yes, all signals required for the design of state of the art robot control system structures	OP
R	SI7	Simulation model preparation for maintenance, updates, modifications and extensions	Yes	OP
<b>Parameterization/ Data Acquisition</b>				
R	PA1	Parameter acquisition sources	All available (e.g. datasheets, product specifications, manuals, etc.), exception: Identification measurements from the real robotic system	OP
O	PA2	Virtual identification measurements	Software: See S04, no further definitions	OP
<b>Motion Planning</b>				
R	MO1	Linear TCP movement	From start position "A" $[x_A, y_A, z_A]$ to target "B" $[x_B, y_B, z_B]$ , defined in workspace, reference: Reference coordinate system (GE2)	OP
R	MO2	Joint movement	From start pose "A" $([q_{1A} \dots q_{6A}])$ to target pose "B" $([q_{1B} \dots q_{6B}])$ , defined in joint space, reference: Home position (GE6)	OP
Approved: OP		Date: 29.03.2019		Page: 2 of 4

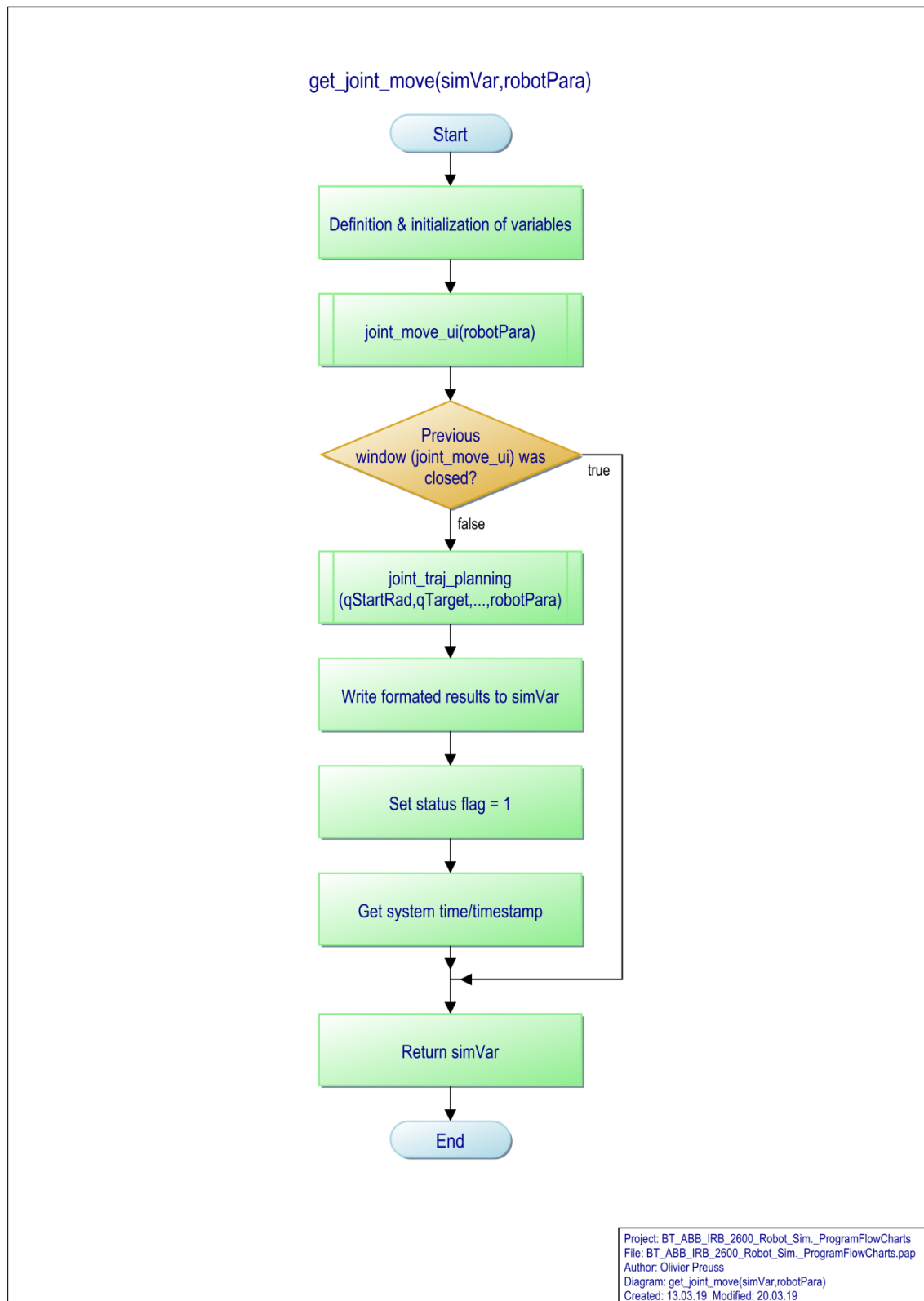
List of Requirements (Continuation)				
Type: (R = Requirement O = Optional)	No.:	Description of Requirement:	Value/Data:	Responsibility:
<b>Programming</b>				
R	PR1	Programming languages	MATLAB programming language and Simulink block diagrams	OP
R	PR2	Preferred usage of predefined MATLAB and Simulink contents (toolboxes, classes, functions, blocks, etc.)	Yes	OP
R	PR3	Code modularity	Yes	OP
R	PR4	Code structure(s)	Program flow chart based	OP
R	PR5	Program prepared for maintenance, updates, modifications and extension	Yes	OP
<b>User Interfaces</b>				
R	US1	Minimum requirement for user inputs (interface)	MATLAB Command Window	OP
O	US2	Graphical User Interface (GUI)	Optional, no further definitions	OP
R	US3	Start and target position format and unit (linear path)	Format: [x y z], unit: [mm]	OP
R	US4	Start and target joint angle format and unit (joint movement)	Format: [q <sub>1</sub> q <sub>2</sub> q <sub>3</sub> q <sub>4</sub> q <sub>5</sub> q <sub>6</sub> ], unit: [°]	OP
<b>Software</b>				
R	SO1	Operating system	Microsoft® Windows® 7 Service Pack 1 or higher	OP
R	SO2	Simulation software	MathWorks® MATLAB® R2018b or higher	OP
R	SO3	CAD software	SolidWorks™ 2001Plus, OR WildFire® 2.0, OR Creo® 13 1.0, OR Autodesk Inventor® 2009 or higher	OP
O	SO4	Other simulation software	ABB RobotStudio 6.08 or higher	OP
Approved: OP		Date: 29.03.2019		Page: 3 of 4

List of Requirements (Continuation)				
Type: (R = Requirement O = Optional)	No.:	Description of Requirement:	Value/Data:	Responsibility:
<b>Economy</b>				
R	EC1	Overall budget	0 €	OP
<b>Documentation</b>				
R	DO1	Thesis document template	TAMK thesis template (2019)	OP
R	DO2	Thesis document specifications	TAMK thesis guidelines (Virikko & Lamminsivu 2017) & HsH thesis guidelines (Diersen 2017a), (Diersen 2017b), (Diersen 2017c)	OP
R	DO3	Simulation model operating manual	Yes	OP
<b>Schedule/Administration</b>				
O	SC1	Submission of a preliminary version of the simulation model	Yes	OP
R	SC2	Submission of a preliminary version of the thesis document	Yes, until 15.04.2019	OP
R	SC3	Final submission of the thesis work	Until 20.04.2019	OP
Approved: OP		Date: 29.03.2019	Page: 4 of 4	

Appendix 4. Program Flow Charts

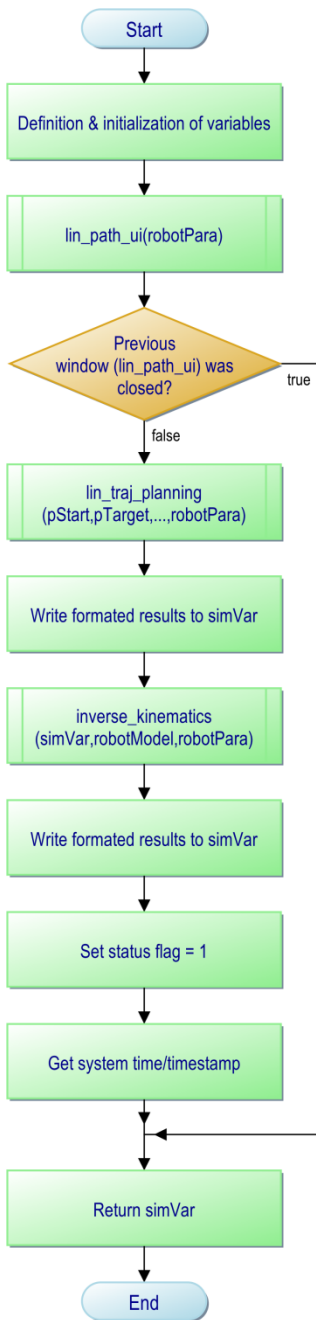
1 (18)



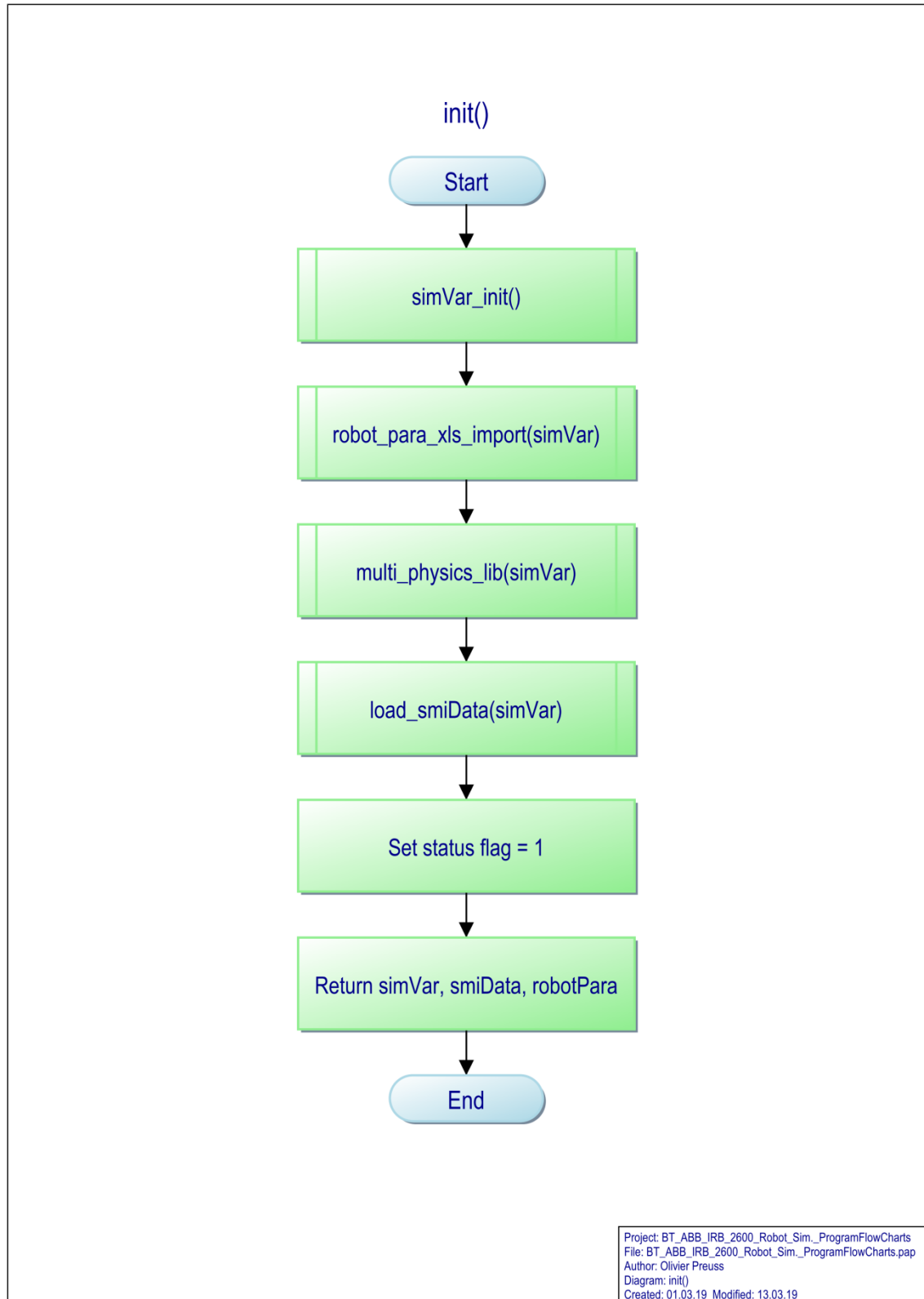




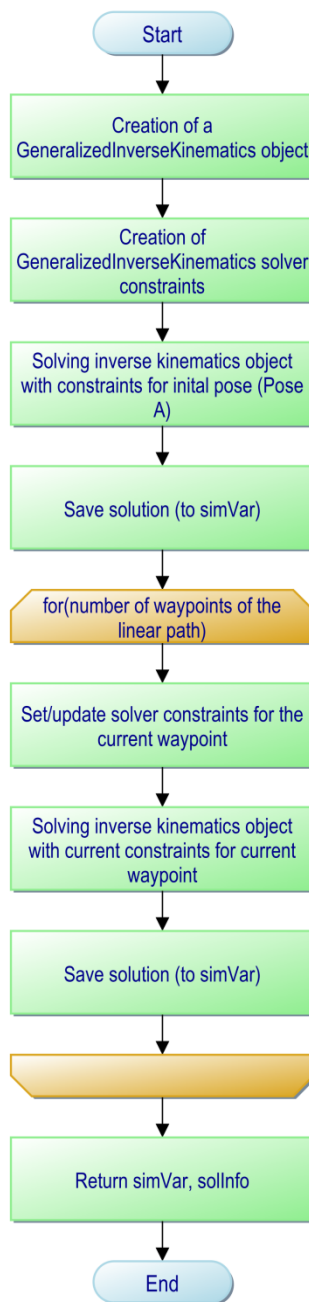
get\_lin\_move(simVar,robotPara,robotModel)



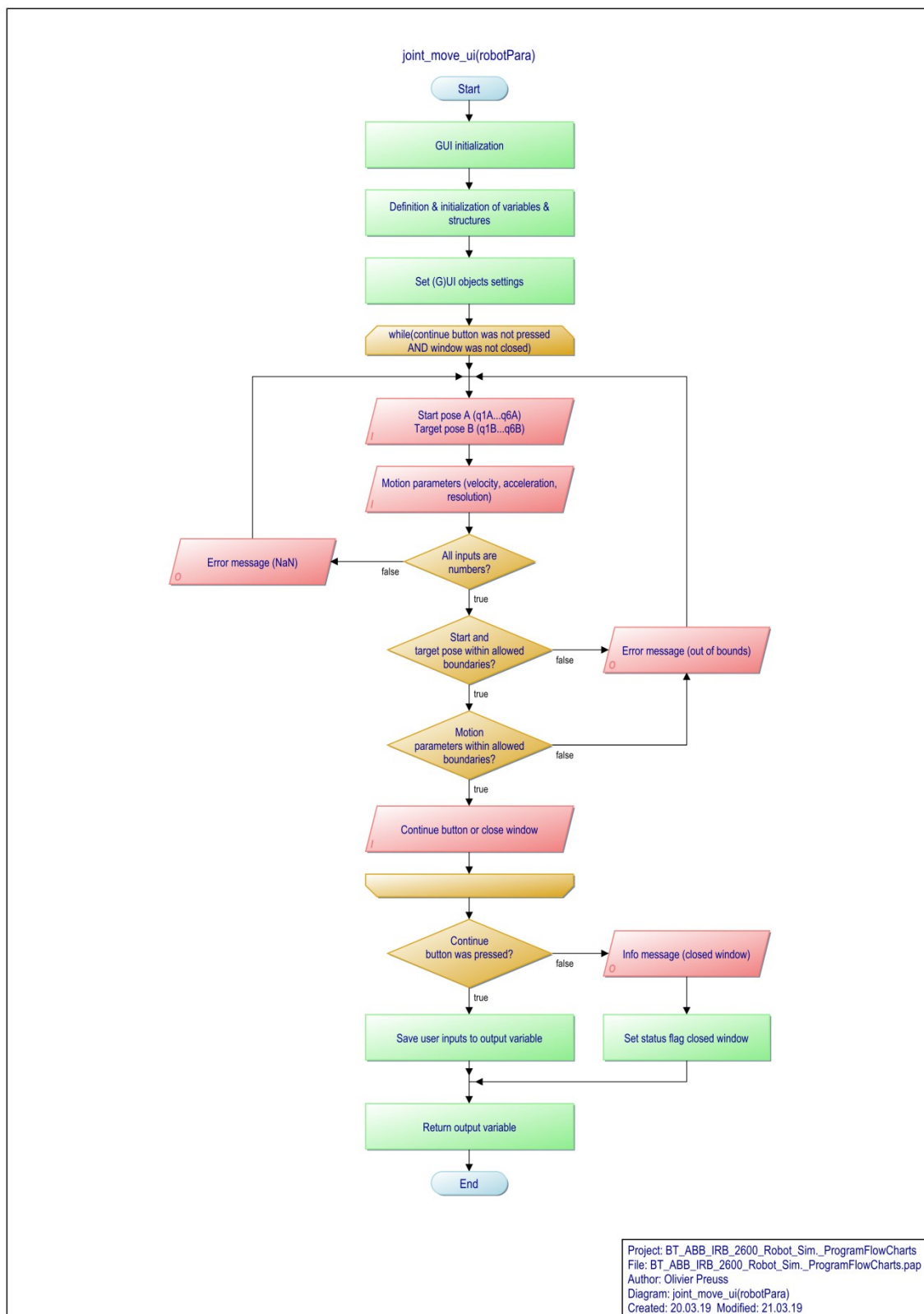
Project: BT\_ABB\_IRB\_2600\_Robot\_Sim\_ProgramFlowCharts  
File: BT\_ABB\_IRB\_2600\_Robot\_Sim\_ProgramFlowCharts.pap  
Author: Olivier Preuss  
Diagram: get\_lin\_move(simVar,robotPara,robotModel)  
Created: 13.03.19 Modified: 20.03.19



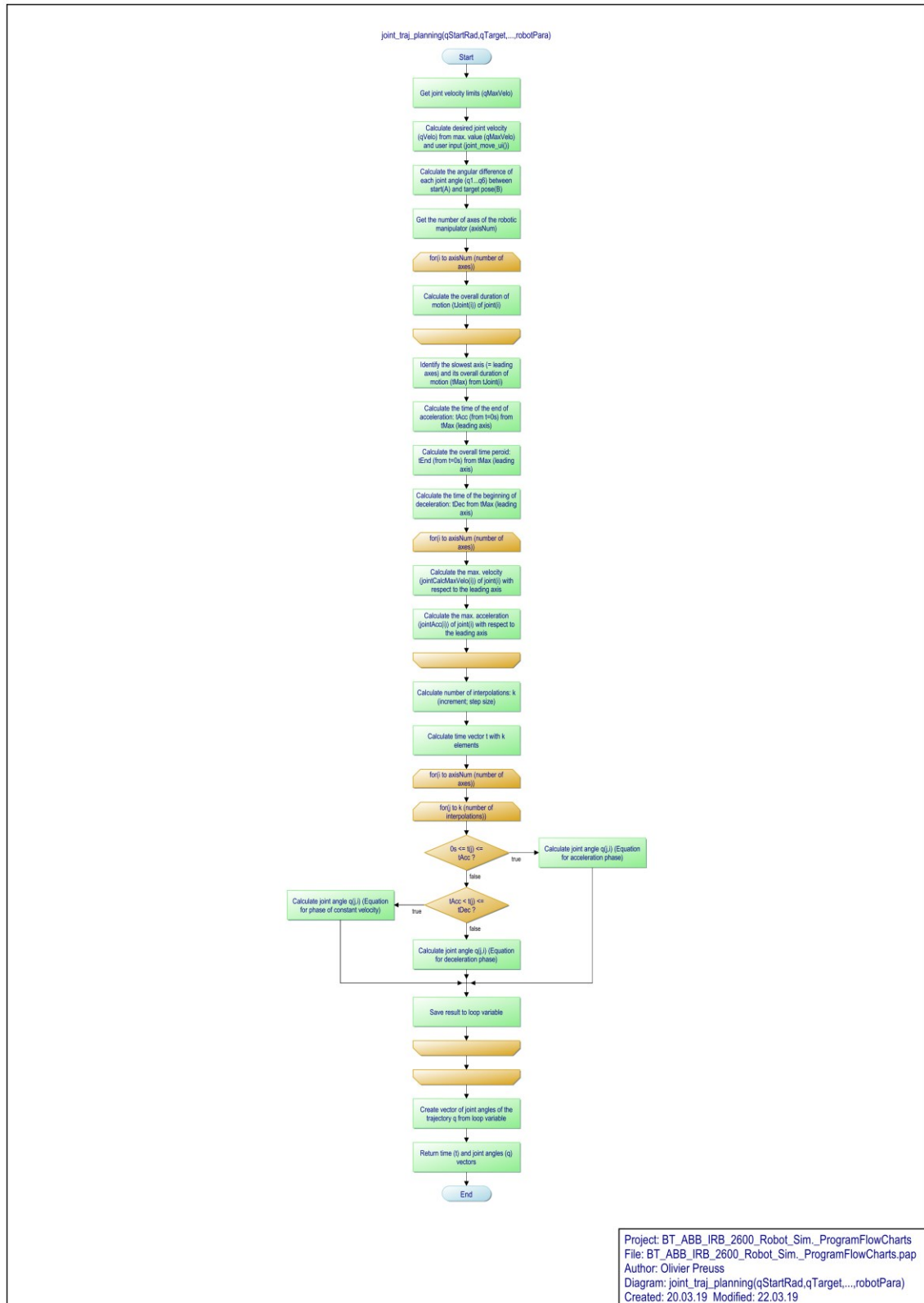
inverse\_kinematics(simVar,robotModel,robotPara)



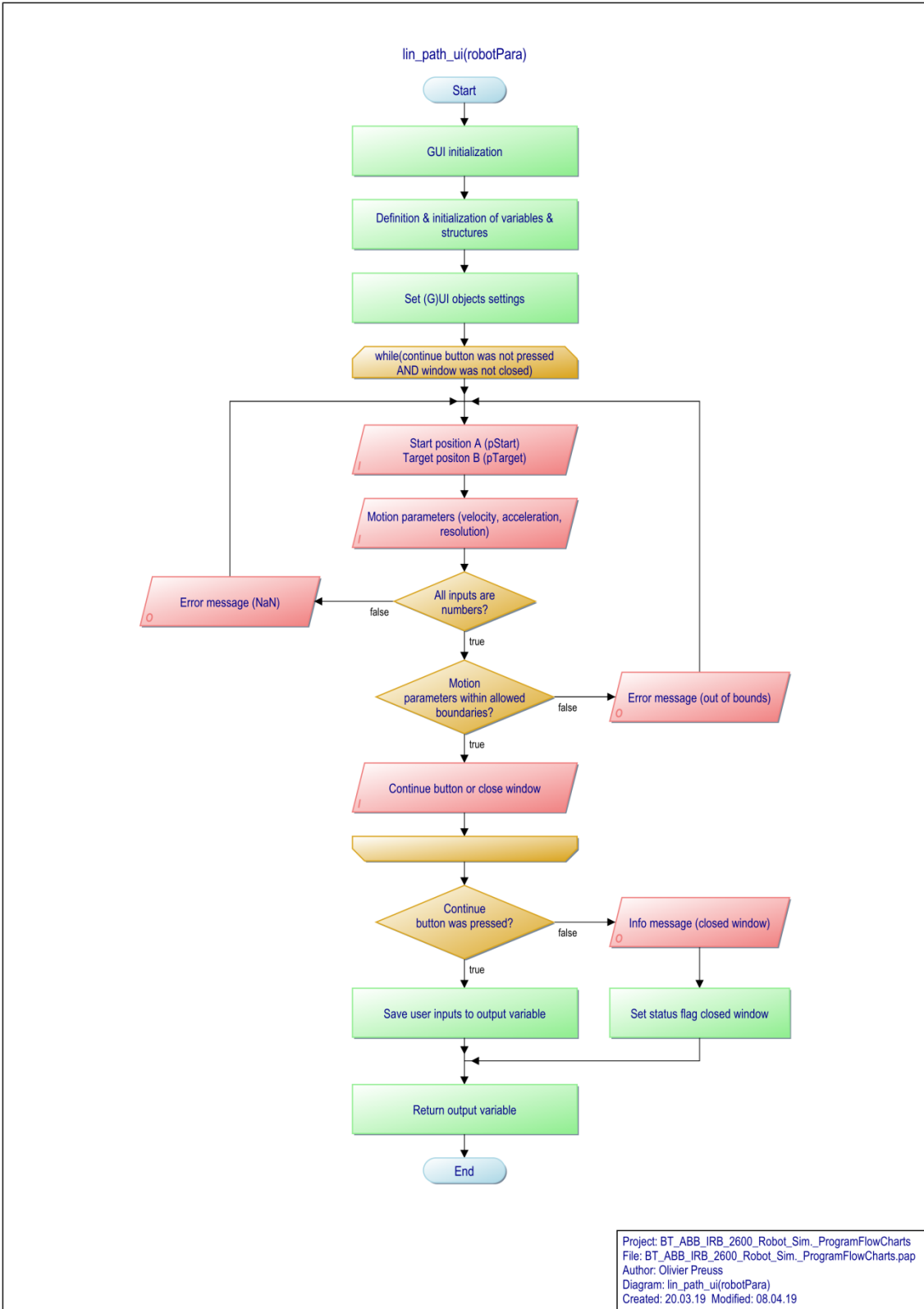
Project: BT\_ABB\_IRB\_2600\_Robot\_Sim\_ProgramFlowCharts  
File: BT\_ABB\_IRB\_2600\_Robot\_Sim\_ProgramFlowCharts.pap  
Author: Olivier Preuss  
Diagram: inverse\_kinematics(simVar,robotModel,robotPara)  
Created: 14.03.19 Modified: 14.03.19



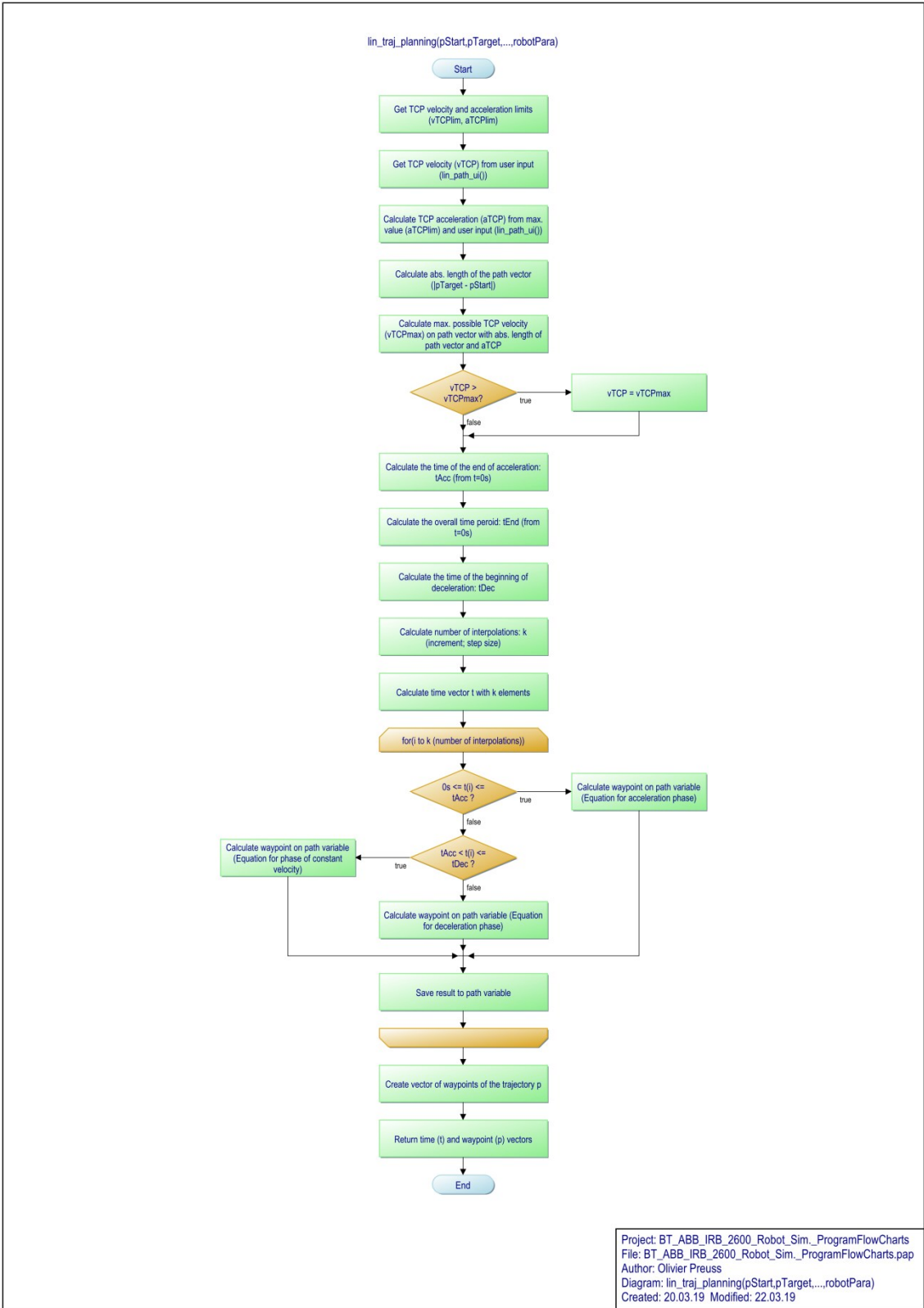
Project: BT\_ABB\_IRB\_2600\_Robot\_Sim\_ProgramFlowCharts  
 File: BT\_ABB\_IRB\_2600\_Robot\_Sim\_ProgramFlowCharts.pap  
 Author: Olivier Preuss  
 Diagram: joint\_move\_ui(robotPara)  
 Created: 20.03.19 Modified: 21.03.19

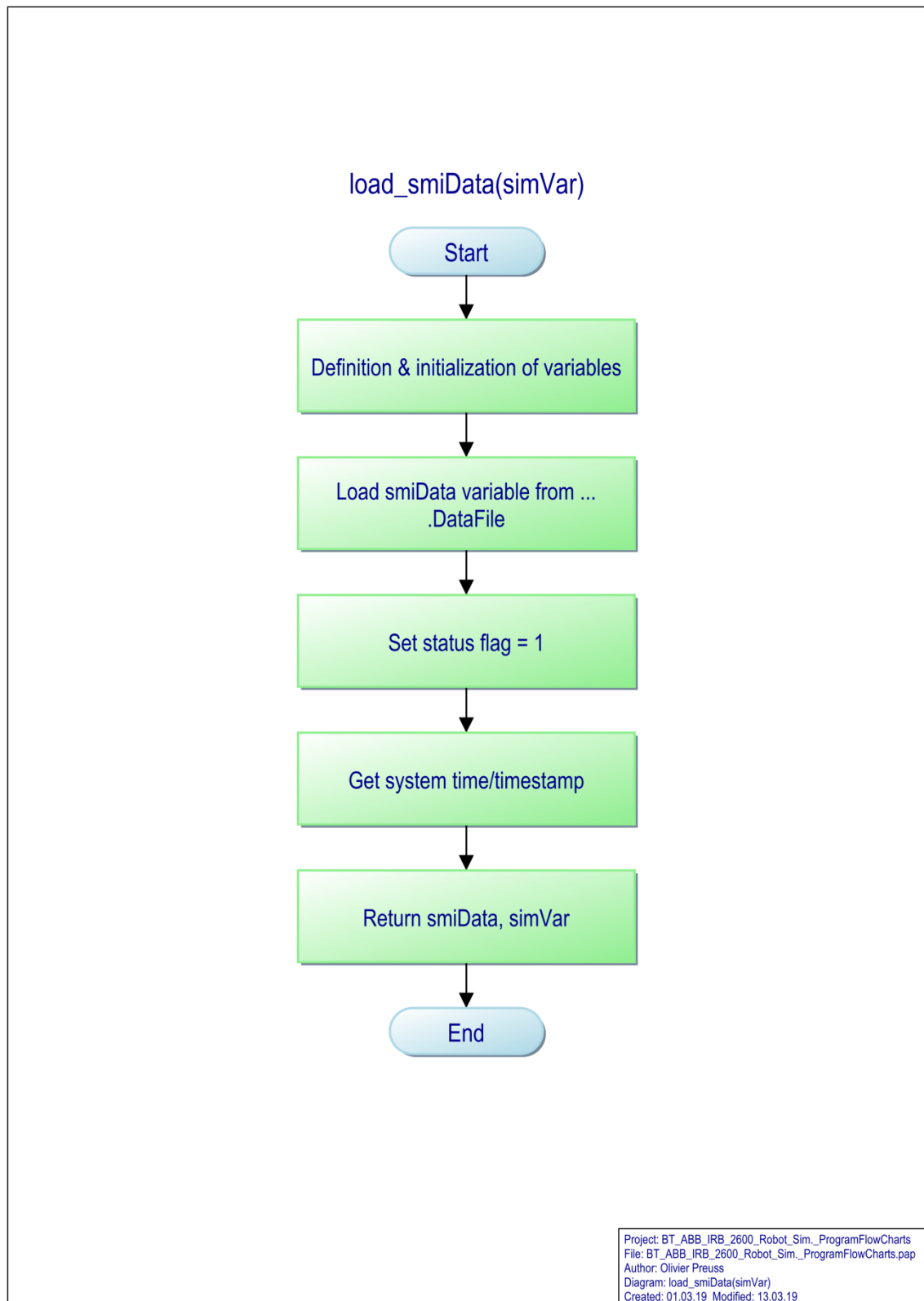


Project: BT\_ABB\_IRB\_2600\_Robot\_Sim\_ProgramFlowCharts  
 File: BT\_ABB\_IRB\_2600\_Robot\_Sim\_ProgramFlowCharts.pap  
 Author: Olivier Preuss  
 Diagram: joint\_traj\_planning(qStartRad,qTarget,...,robotPara)  
 Created: 20.03.19 Modified: 22.03.19

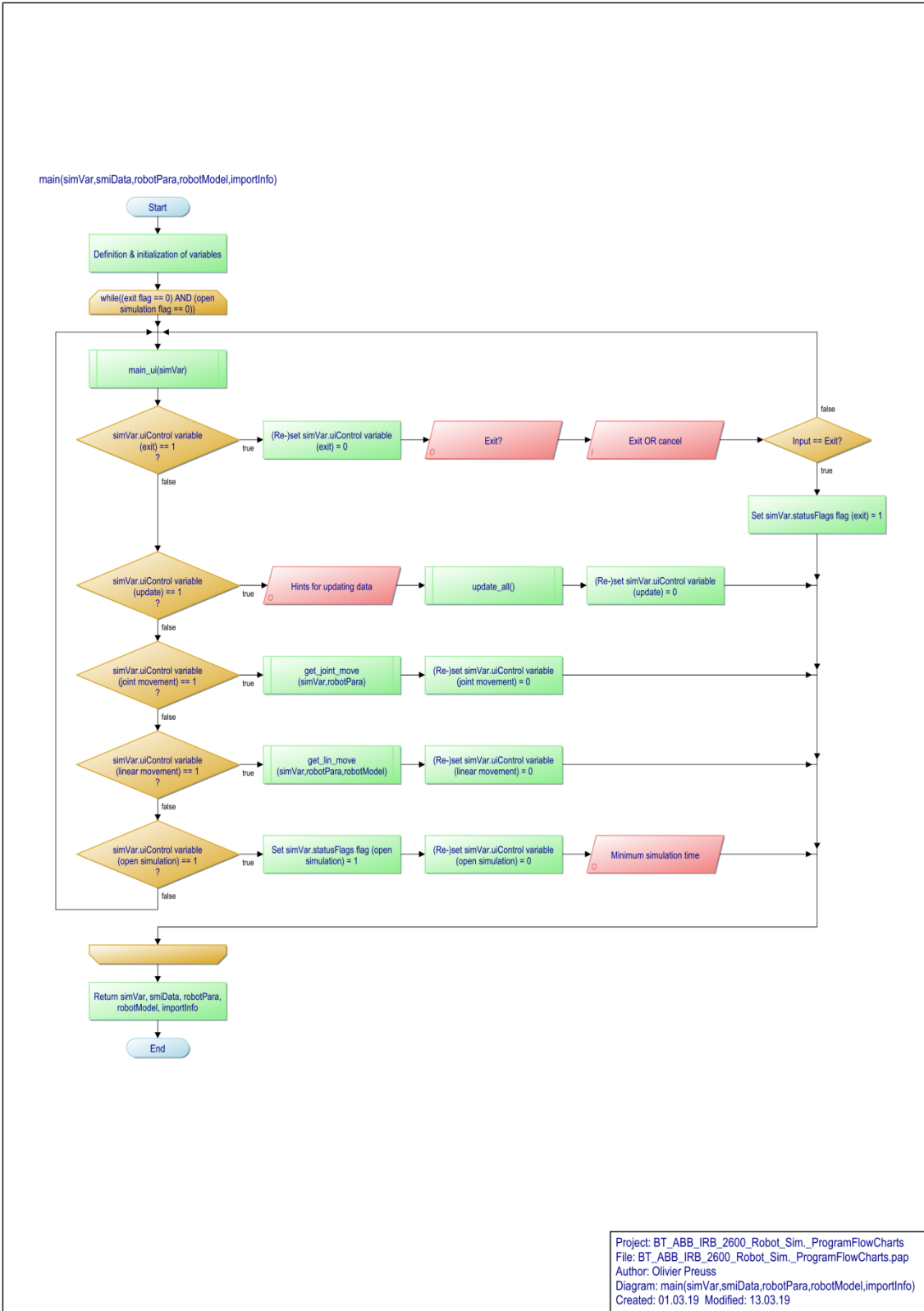


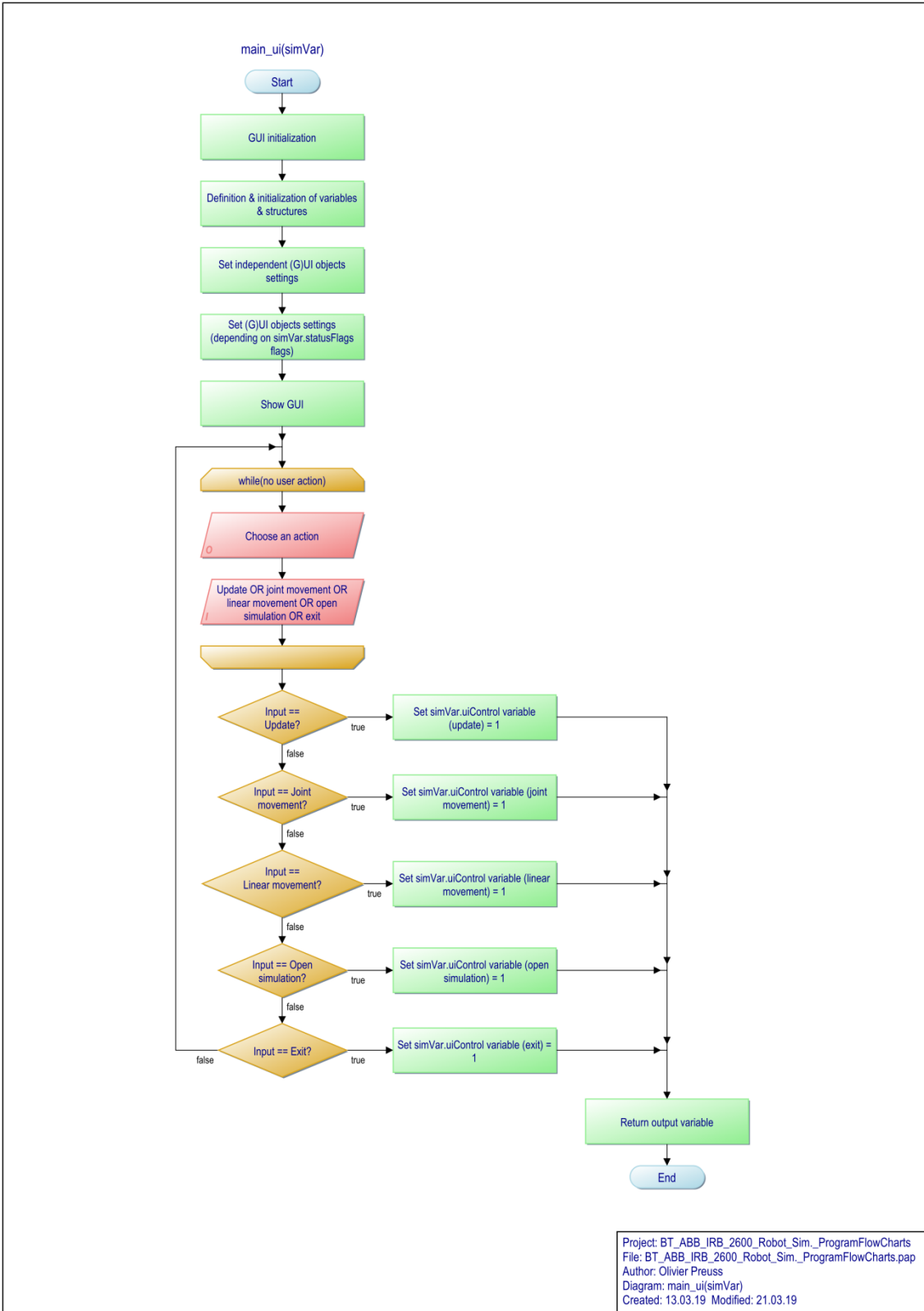
Project: BT\_ABB\_IRB\_2600\_Robot\_Sim\_ProgramFlowCharts  
 File: BT\_ABB\_IRB\_2600\_Robot\_Sim\_ProgramFlowCharts.pap  
 Author: Olivier Preuss  
 Diagram: lin\_path\_ui(robotPara)  
 Created: 20.03.19 Modified: 08.04.19

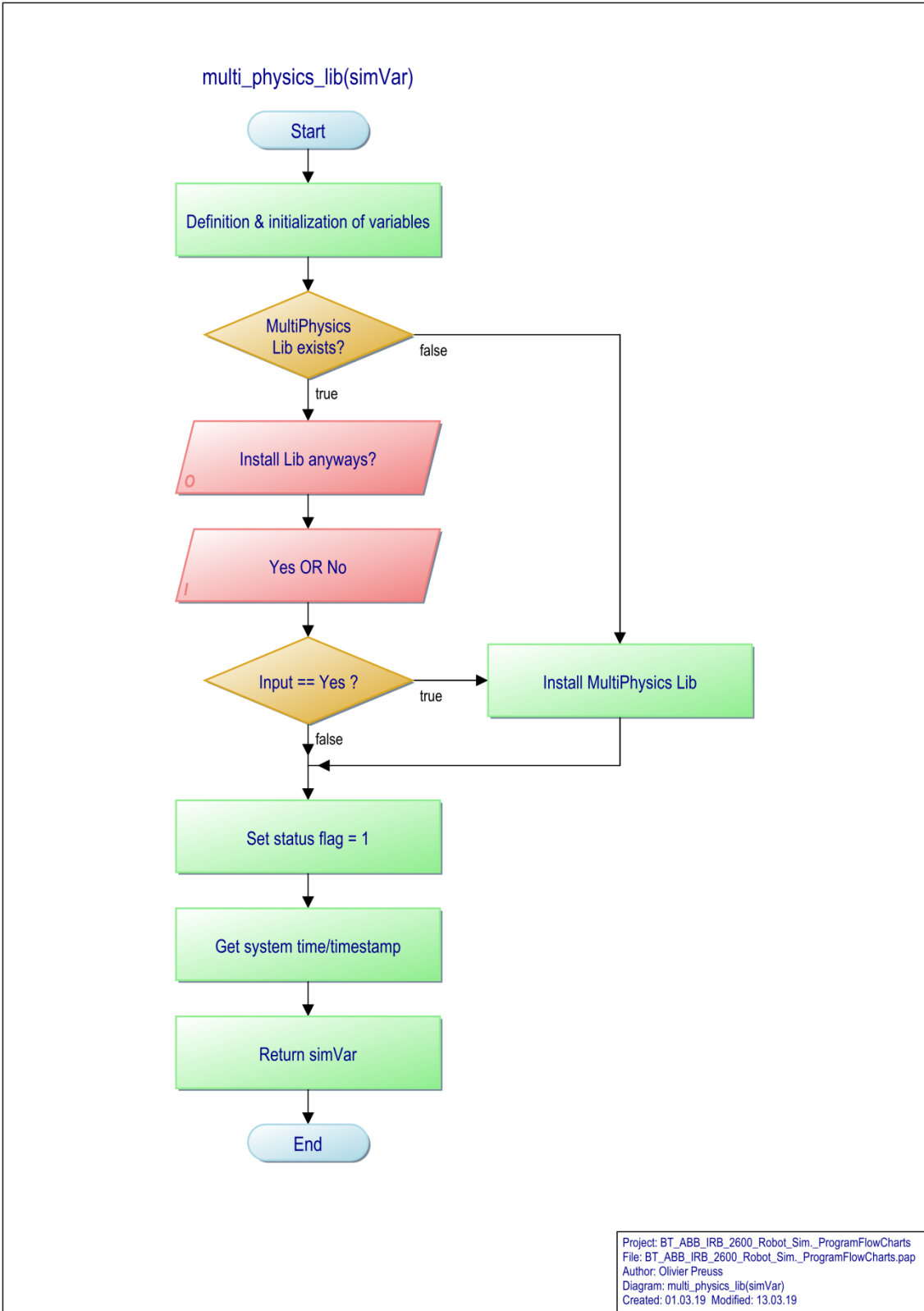




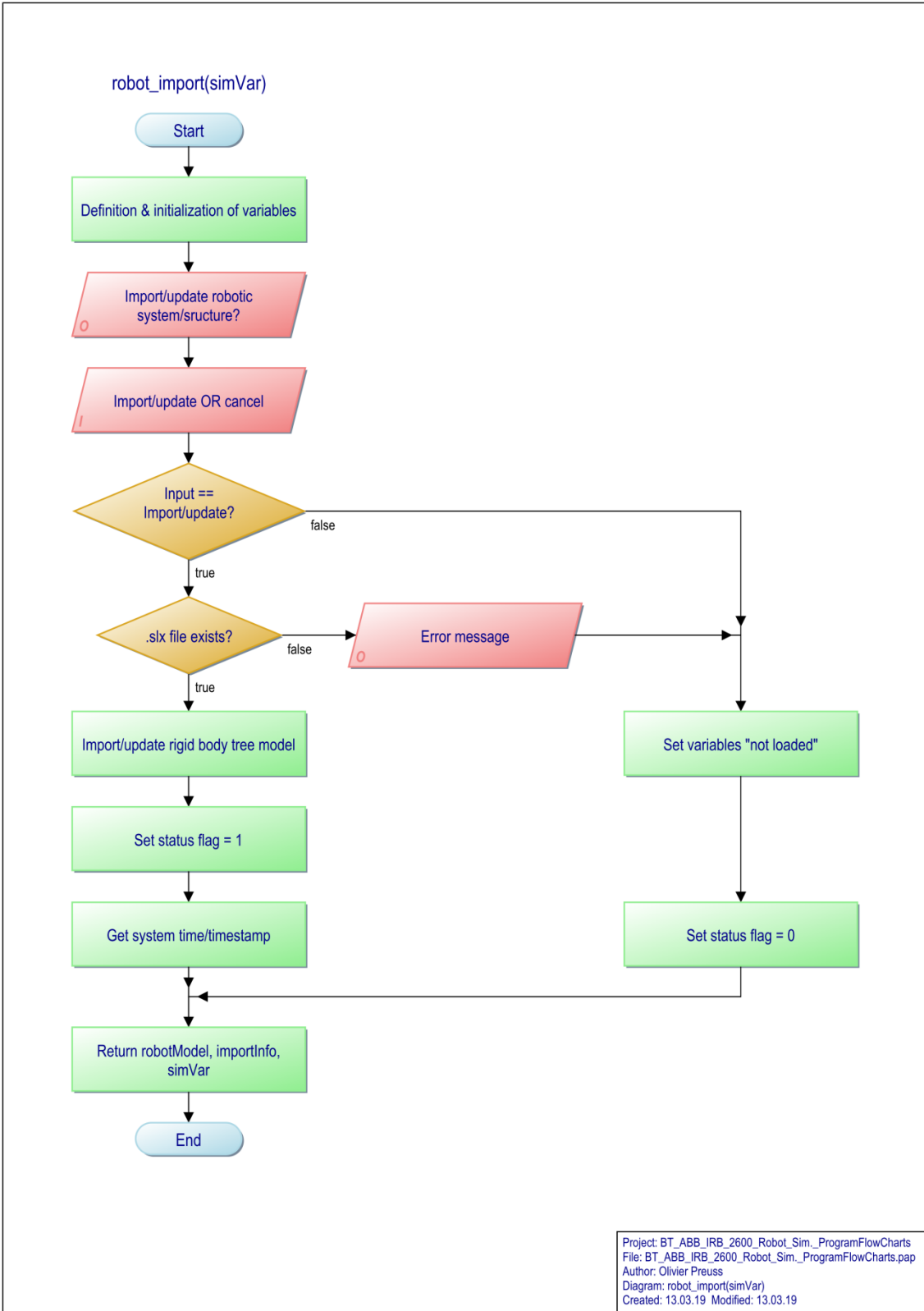


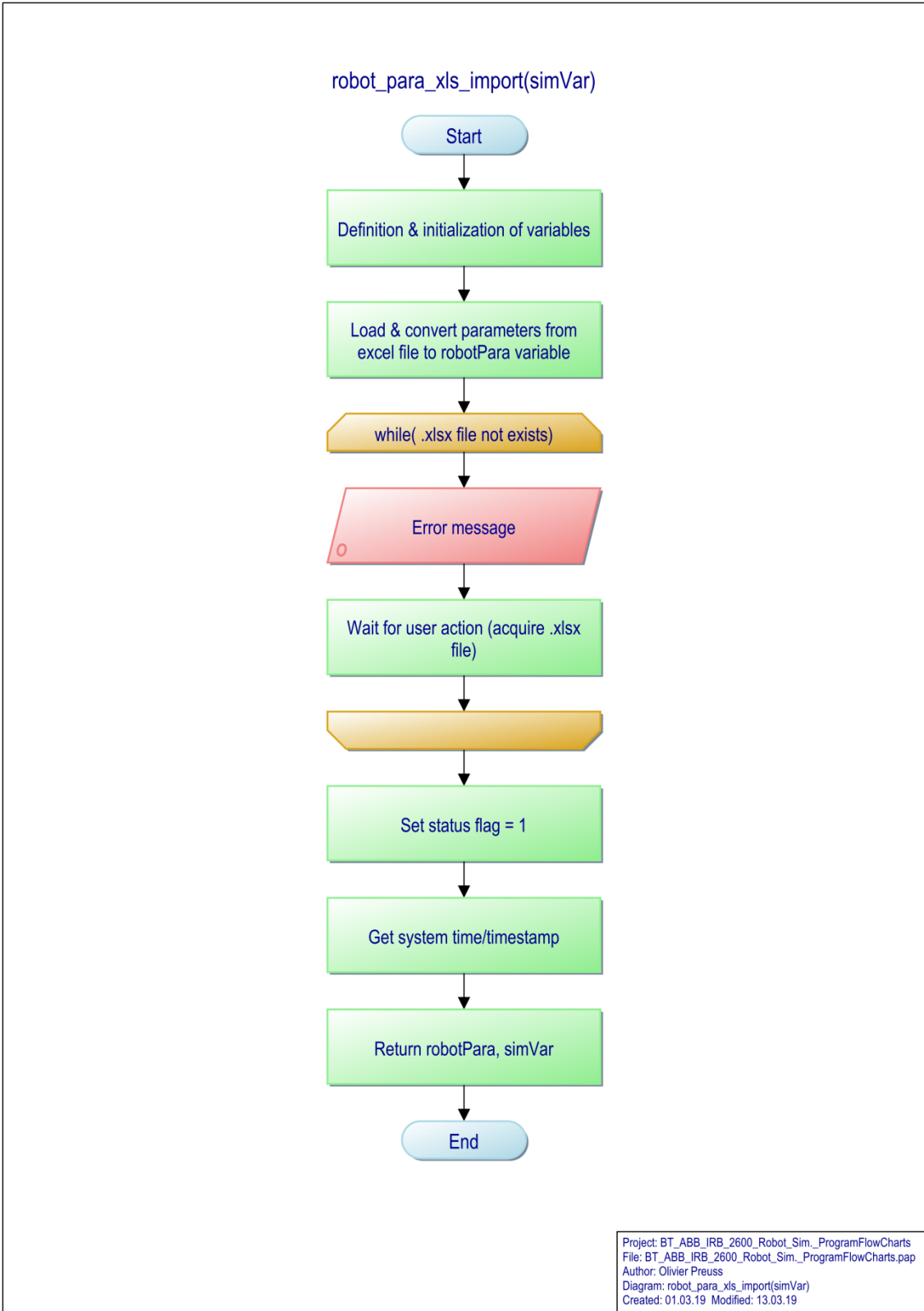


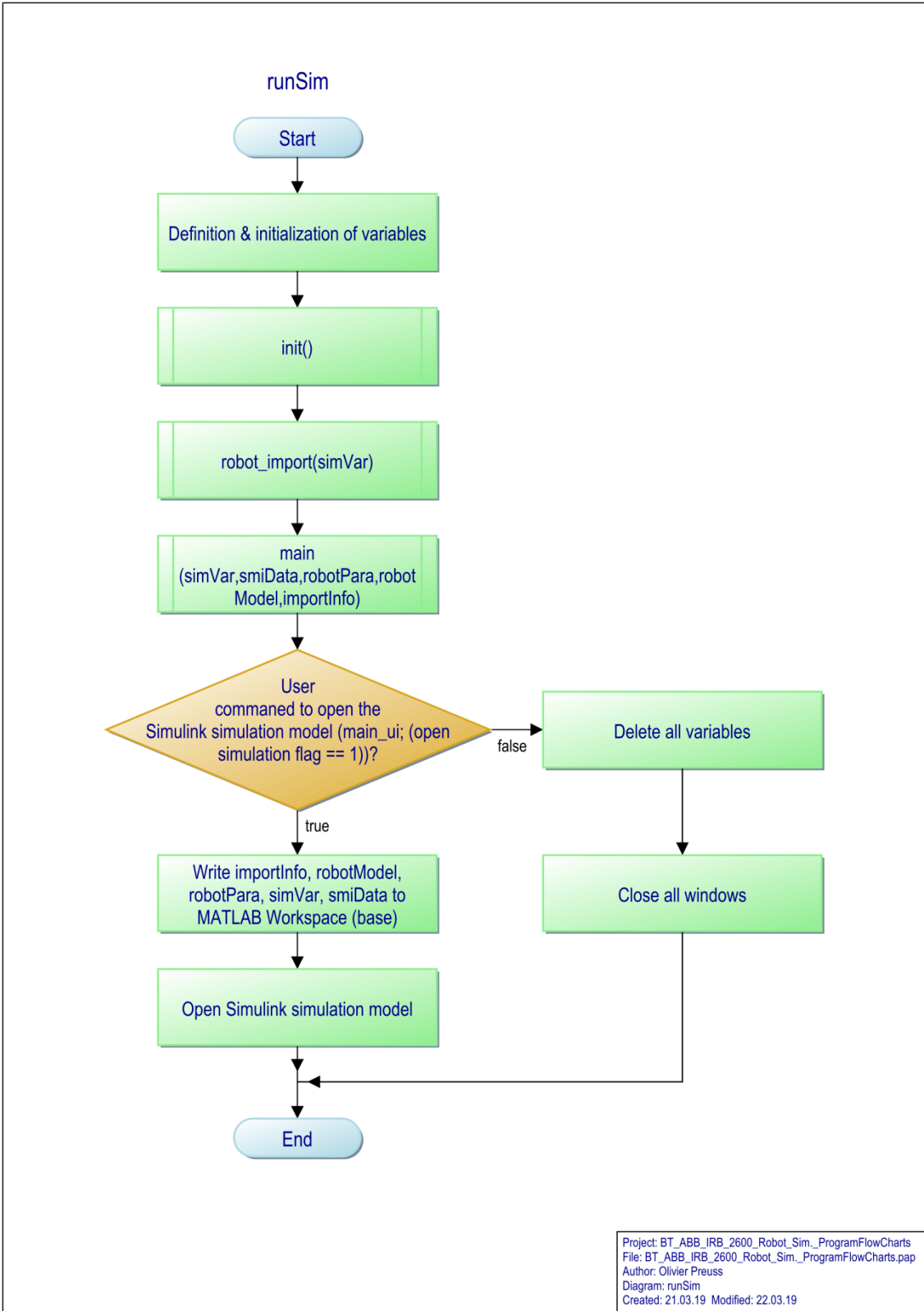




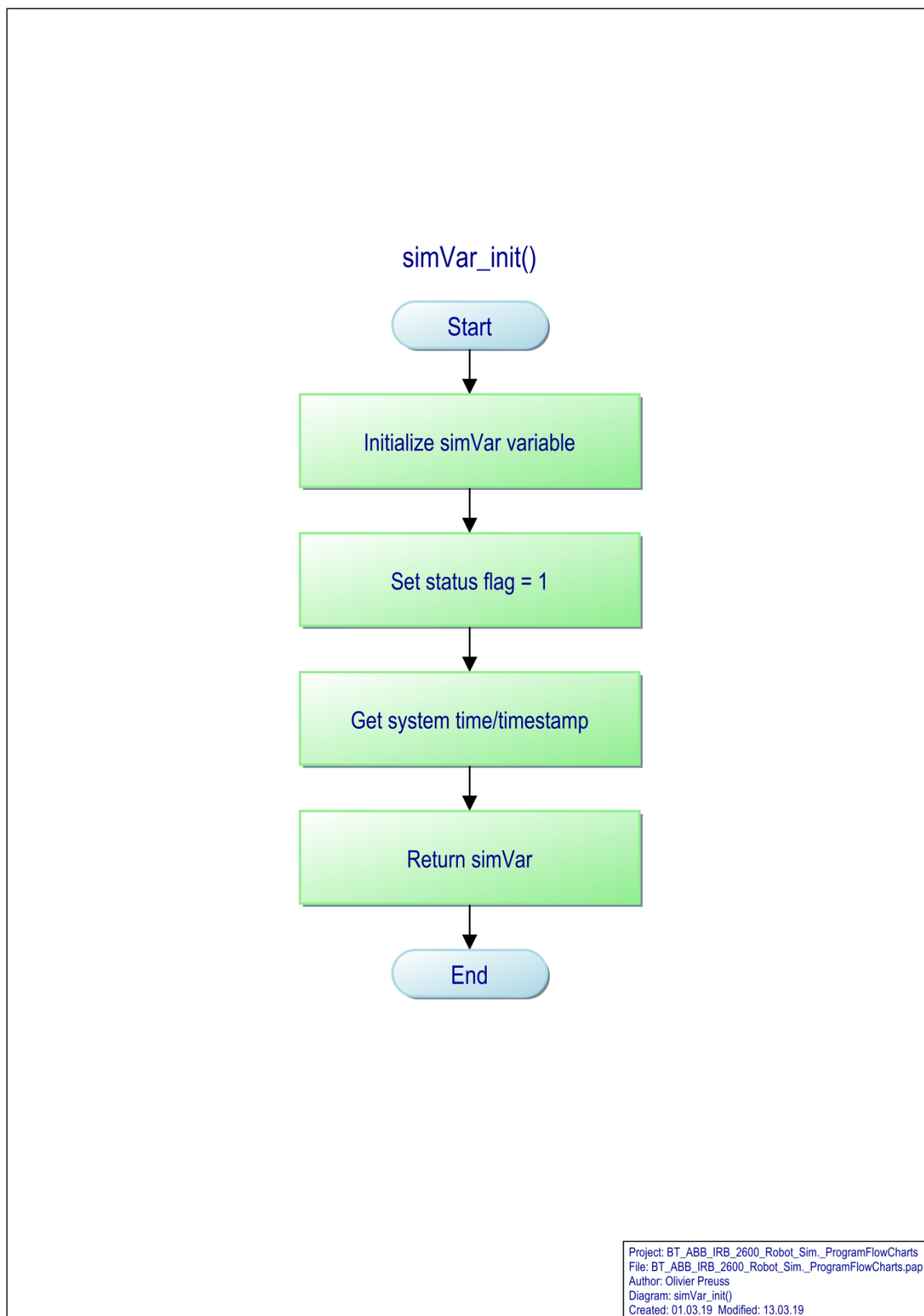
Project: BT\_ABB\_IRB\_2600\_Robot\_Sim\_ProgramFlowCharts  
File: BT\_ABB\_IRB\_2600\_Robot\_Sim\_ProgramFlowCharts.pap  
Author: Olivier Preuss  
Diagram: multi\_physics\_lib(simVar)  
Created: 01.03.19 Modified: 13.03.19

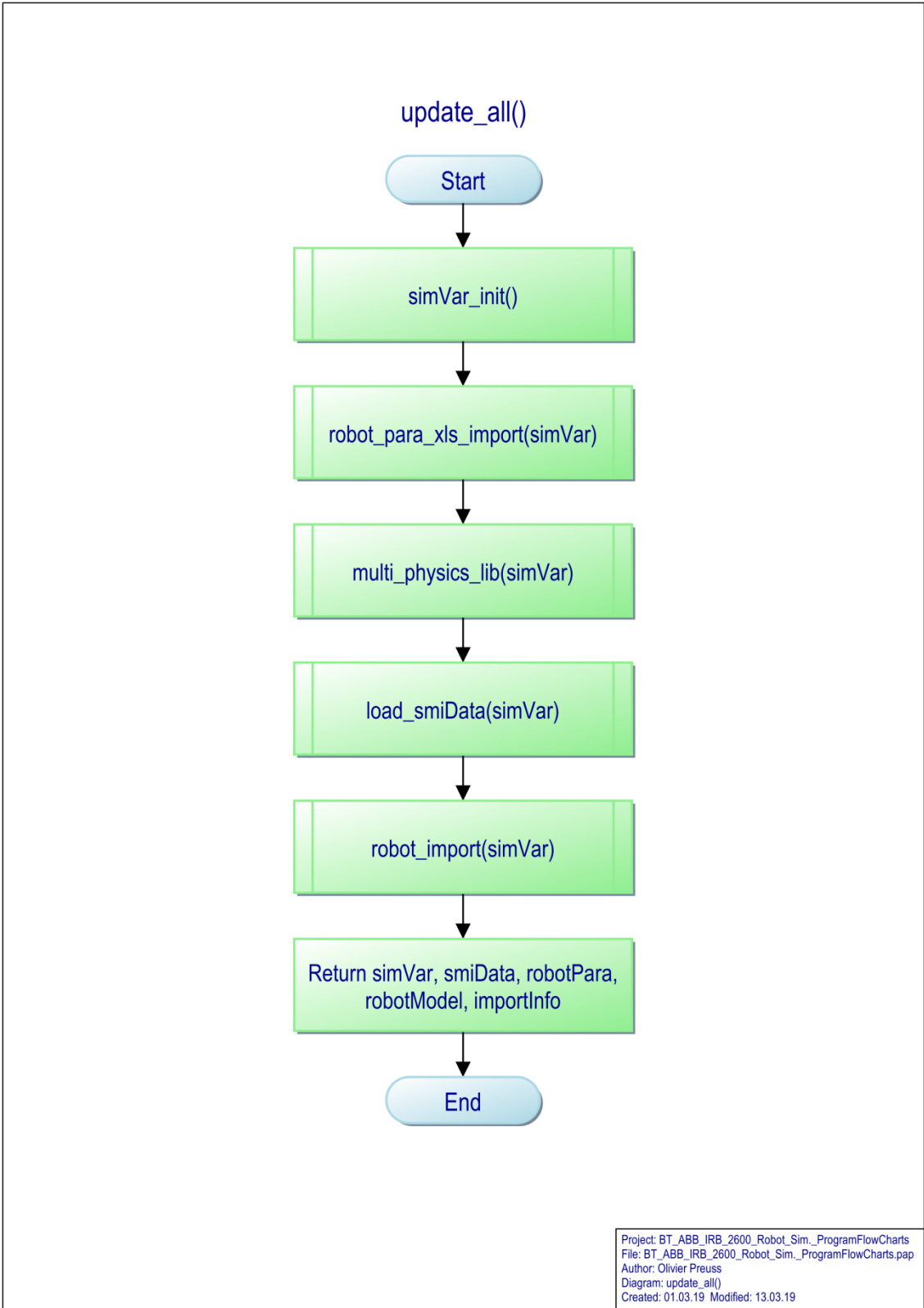






Project: BT\_ABB\_IRB\_2600\_Robot\_Sim\_ProgramFlowCharts  
 File: BT\_ABB\_IRB\_2600\_Robot\_Sim\_ProgramFlowCharts.pap  
 Author: Olivier Preuss  
 Diagram: runSim  
 Created: 21.03.19 Modified: 22.03.19







## Appendix 5. Operating Manual

1 (36)

This page was intentionally left blank in order to make the subsequent operating manual an independent document that can be extracted from the thesis document separately.

# MANUAL

**A Quick Start Guide for the MATLAB® Simulink®/  
Simscape™ (Multibody™)<sup>1</sup> Simulation Model of an  
ABB<sup>2</sup> IRB 2600-12/1.85 Industrial Robot Manipulator**

Author:	Olivier Preuss
First Issued:	05.03.2019
Last Edited:	16.04.2019
Recent Version:	v05
Revision:	A

Corresponding Document:	Preuss, O. 2019. Simulation Model for a Six Axis Articulated Arm Industrial Robot. Mechanical and Production Engineering. Tampere University of Applied Sciences. Bachelor's thesis
-------------------------	---

---



<sup>1</sup> MATLAB®, Simulink®, Simscape™ and Simscape™ Multibody™ are trademarks or registered trademarks of The MathWorks, Inc.

<sup>2</sup> ABB Asea Brown Boveri Ltd.


**ABOUT THIS DOCUMENT**


3 (36)

The purpose of the document at hand is a quick introduction to the start-up, the operation and updating and extending/ modifying the MATLAB Simulink/ Simulink Simscape (Multibody) simulation of an ABB IRB 2600-12/1.85 industrial robot manipulator, initially equipped with a Fronius<sup>3</sup> Robacta Drive CMT welding torch/ end effector. This document refers to the bachelor's thesis: "Simulation Model for a Six Axis Articulated Arm Industrial Robot", by Olivier Preuss, published in April 2019 at Tampere University of Applied Sciences (TAMK) in Tampere, Finland.

Symbol:	Meaning:
	Useful information/ hint.
	Important note, read carefully!

**GENERAL INFORMATION**

	Do not delete or add any data from/ to the simulation data set. Also do not rename, relocate or change the general structure(s) and location(s) of any of folders or files of the data set. (Exceptions: CAD MODEL UPDATE and EXTENSIONS/ MODIFICATIONS (refer to the corresponding manual pages 179 and 182). Always use "Save as" in order to retain an unchanged copy of the simulation.
---	---

	For further information not covered by this manual, refer to the corresponding thesis document.
---	---

- In order to keep this document as short as possible, [hyperlinks](#) are used to redirect to external web sources (provided by manufacturers/ developers/ other third parties) whenever reasonable.
- Any *filenames.m* or *folder names* are written in italicised, coloured font and contain the file extension(s).
- MATLAB related naming and commands are written in `Courier new` font.
- Other "commands", "window names", "button names" etc. are in quotation marks.

---

<sup>3</sup> Fronius International GmbH

**CONTENTS**

0. PREREQUISITES .....	163
1. INTRODUCTION .....	164
2. INSTALLATION .....	165
3. OPERATION.....	166
4. CHANGE OF PARAMETERS.....	177
5. CAD MODEL UPDATE .....	179
6. EXTENSIONS/MODIFICATIONS .....	182
7. TROUBLESHOOTING .....	193

**0. PREREQUISITES**

5 (36)

The usage of the simulation program requires basic general knowledge of MATLAB and Simulink/ Simulink Simscape. Basic knowledge of programming, mechatronics, robotics and control systems are also recommended.

Additionally required:

Subject:	Required (R)/ Optional (O):	Note:
Personal Computer (PC)	R	Minimums: Processor: Intel <sup>®4</sup> or AMD <sup>®5</sup> x86-64, RAM: 4GB, HDD: 4-6 GB free disk space, Graphics: OpenGL <sup>®6</sup> 3.3 with 1GB GPU
Operating System (OS)	R	Microsoft <sup>®7</sup> Windows <sup>®7</sup> 7 Service Pack 1, Apple <sup>®8</sup> macOS <sup>®8</sup> 10.12, Linux <sup>®9</sup> : see <sup>10</sup> or higher
Simulation Software	R	MathWorks MATLAB R2018b or higher (In accordance with the used OS)
CAD Software	O	SolidWorks <sup>™11</sup> 2001Plus, OR WildFire <sup>®12</sup> 2.0, OR Creo <sup>®12</sup> 1.0, OR Autodesk Inventor <sup>®13</sup> 2009 or higher
CAD Software Plug-in	O	MathWorks Simscape Multibody Link Version 6, R2018b (In accordance with the used CAD software) or higher
Spreadsheet Software	O	Microsoft Excel <sup>®7</sup> 2010 and higher
Robot Manufacturers Software	O	ABB RobotStudio 6.08 and higher

Simulation model data set(s)/folder(s):

Folder Name:	Est. File Size [MB]:		Password:
	Zipped:	Unzipped:	
<a href="#">BT_ABB_IRB_2600_Robot_Sim._v_A.zip</a>	18	130	#20RbT19Sim!
<a href="#">BT_ABB_IRB_2600_Robot_Sim._v_B.zip</a>			

<sup>4</sup> Intel<sup>®</sup> is a registered trademark of Intel Corporation

<sup>5</sup> AMD<sup>®</sup> is a registered trademark of Advanced Micro Devices, Inc.

<sup>6</sup> OpenGL<sup>®</sup> is a registered trademark of Hewlett Packard Enterprise

<sup>7</sup> Microsoft<sup>®</sup>, Windows<sup>®</sup> and Excel<sup>®</sup> are trademarks or registered trademarks of Microsoft Corporation

<sup>8</sup> Apple<sup>®</sup> and macOS<sup>®</sup> are registered trademarks of Apple Inc.

<sup>9</sup> Linux<sup>®</sup> is a registered trademark of The Linux Foundation<sup>®</sup>.

<sup>10</sup> <https://www.mathworks.com/support/requirements/matlab-system-requirements.html>

<sup>11</sup> SOLIDWORKS<sup>™</sup> is a trademark of Dassault Systèmes<sup>®</sup>

<sup>12</sup> WildFire<sup>®</sup> and Creo<sup>®</sup> are registered trademarks of PTC Inc.

<sup>13</sup> Autodesk Inventor<sup>®</sup> is a registered trademark of Autodesk Inc.

## 1. INTRODUCTION

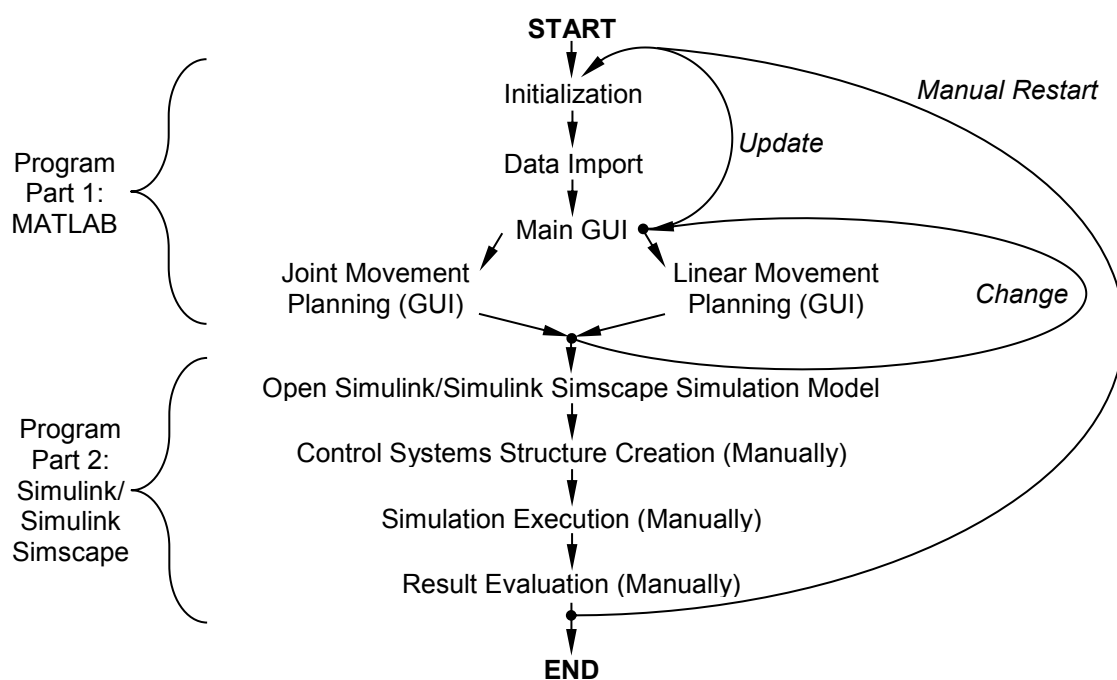
6 (36)

The simulation program and its flow can be divided into two consecutive main program parts – an initial and looped MATLAB part and a following non-looped Simulink/ Simulink Simscape part.

The MATLAB program part acquires and provides all required data for the Simulink/ Simulink Simscape simulation, represented by five main simulation variables (check section EXTENSIONS/ MODIFICATIONS (page 182) for more detailed information), saved to the MATLAB “Workspace” (base) after execution. The contents/ values of the main simulation variables are determined with the help of the import of external data, user inputs and commands received from three graphical user interfaces (GUI) and a number of evaluation algorithms. Two different types of motion planning are also covered by the MATLAB program part.

The Simulink/ Simulink Simscape program part virtually represents the real robotic system as a block diagram structure and uses the values of the formerly mentioned main simulation variables for the model parameterization. In contrast to the flow of the MATLAB program part, it is mostly ran manually (creation of a control system structure, execution of the simulation, evaluation of the results, etc.)

The subsequently shown simplified overall program flow chart may supports understanding the general flow of the simulation procedure.

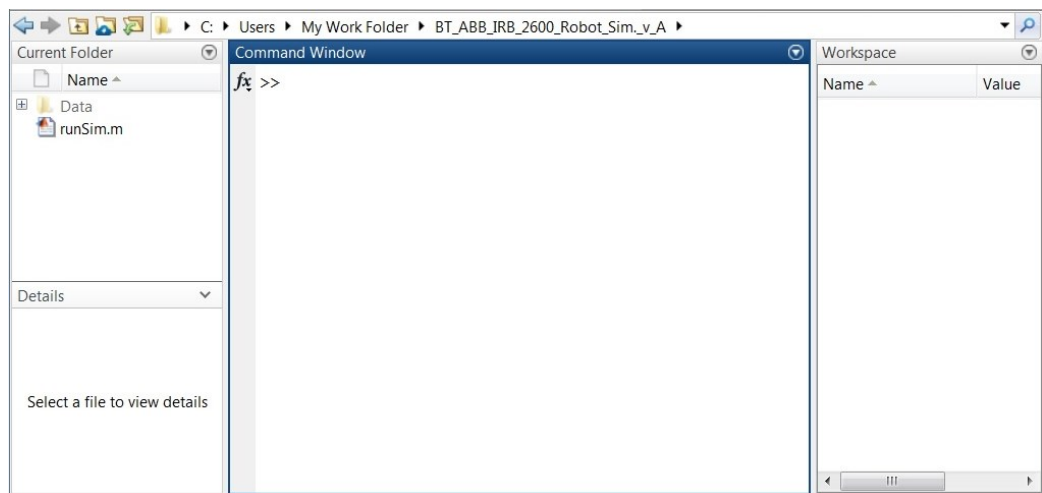


## 2. INSTALLATION

7 (36)

The instructions of step one and step two given in this section, INSTALLATION, only need to be executed once after the initial acquisition of the simulation program data set. Step three needs to be repeated whenever starting or restarting the MATLAB software or for any other necessary reason (e.g. after changing the work directory/ folder).

1. Extract the [BT\\_ABB\\_IRB\\_2600\\_Robot\\_Sim.\\_v\\_A](#) folder from the [BT\\_ABB\\_IRB\\_2600\\_Robot\\_Sim.\\_v\\_A.zip](#) file. While/ before the extraction procedure you will be asked for a password – use the password listed in the second table of the section PREREQUISITES (page 163).
2. Save the extracted [BT\\_ABB\\_IRB\\_2600\\_Robot\\_Sim.\\_v\\_A](#) folder to a proper work directory and folder.
3. Run MATLAB and browse to the work folder prepared in step two. The MATLAB “Address Field” and “Current Folder” sub windows should now look like this:



The installation is now completed. Continue with the section OPERATION (page 166).

### 3. OPERATION

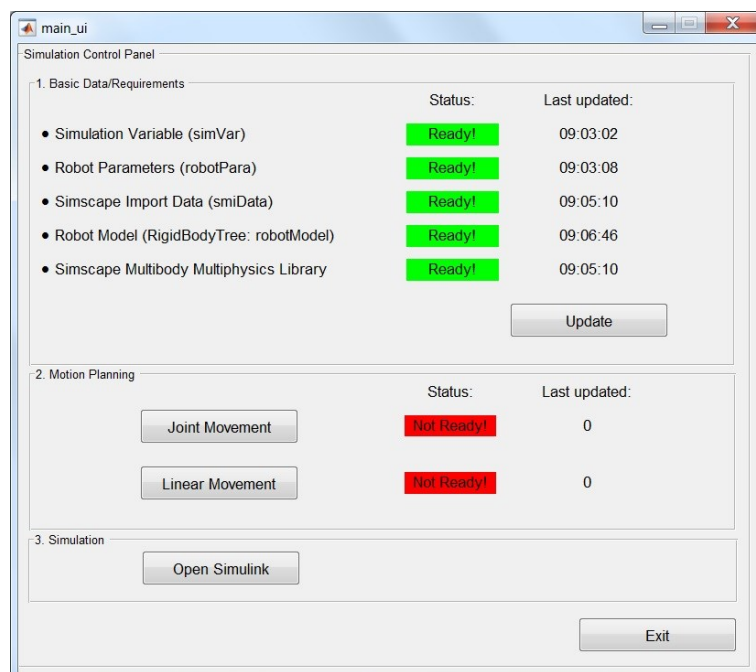
8 (36)

The section OPERATION covers instructions for the operation of the unedited, original simulation program. The subsequently shown instructions and/ or sequences may not be applicable for extended/ modified versions of the simulation program.

As mentioned in the INTRODUCTION (page 164), the simulation program can be divided in two parts, therefore, the instructions for the operation are also divided into two consecutive parts: MATLAB and Simulink/ Simulink Simscape. Refer to the section INSTALLATION (page 165) before continuing.

#### MATLAB:

1. Type `runSim;` to the MATLAB “Command Window” and press “Enter”.
2. If not already existent, MATLAB now automatically installs the required Simscape Multibody Multiphysics Library R2018b Version 2.7.0.0. Press “OK” to close the “Installation Successful” window and to continue.
3. The “Robot System Import/Update” window now appears. Press “Import/Update Robot System now” (The import procedure may take several seconds).
4. If executed successfully, the “main\_ui” window containing the “Simulation Control Panel” should open and appear similar to:

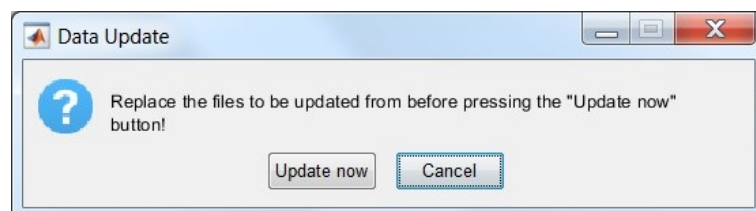




<b>i</b>	The “main_ui” window is always existent (loop) while running the MATLAB program part and allows updating or changing the variable contents at any time (except during the ”joint_move_ui” or “lin_path_ui” windows are open). If brought to the background, restore the window(s) from the task bar of your operating system.
----------	---

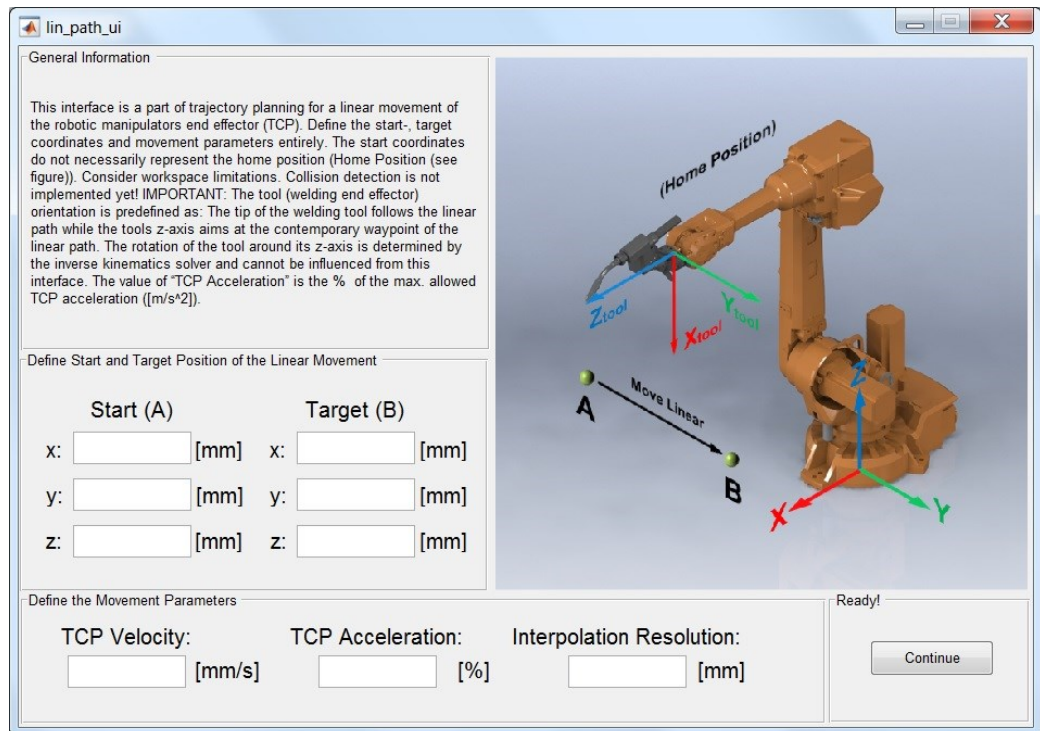
<b>!</b>	Always use the “Exit” button of the “main_ui” window to terminate the simulation program properly without any errors.
----------	---

5. To update any data listed in Panel 1, “Basic Data/Requirements”, press the “Update” button. **Make sure to replace/ update the files to be updated from before pressing the “Update” button!** A “Data Update” window will appear - press the "Update now" button to continue, press the "Cancel" button to terminate the update procedure.



Following this, two other windows will appear in a sequence. Press the "Install" button of the first "Library Installation" window if you wish to update the Simscape Multibody Multiphysics Library. Press "Cancel" to skip this step. Press the "Import/Update Robot System now" button of the second "Robot System Import/Update" window if you wish to update the `robotModel` and `importInfo` variables. Press "Cancel" to skip this step. The `simVar`, `smiData` and `robotPara` variables are updated automatically.

6. The simulation of the Simulink/ Simulink Simscape simulation model requires a motion planning in order to provide the required set values for the (revolute) joints of the robotic manipulator's model. Press the "Joint Movement" or "Linear Movement" button of Panel 2 "Motion Planning" to start the procedure of planning the desired motion (see the figure of the "main\_ui" window below step four). (In this manual, only the "Linear Movement" procedure is presented. The "Joint Movement" procedure is quite similar; therefore, the subsequent instructions are also valid.)
7. A separate GUI ("lin\_path\_ui" or "joint\_move\_ui" window(s)) will be opened. Read the information given in the Panel "General Information" of the "lin\_path\_ui" window carefully!



!

All input fields need to be defined by appropriate input values. Motion planning cannot be executed successfully without completely and correctly filled input fields. Inputs are filtered and checked for being a number and being within the allowed boundaries. You may obtain the joint/ axis angle limitations, the movement parameters limits and the workspace limitations from the corresponding thesis document (section 3.3, TABLE 3.1) and/ or the [ABB\\_IRB\\_2600-12-1.85\\_Parameters.xlsx](#) spreadsheet and/ or the robotic manipulator manufacturer's documents.

8. Define all input fields by applying appropriate values. Press the “Continue” button of the Panel “Ready!” to calculate the trajectory and to return to the “main\_ui” window.

<b>!</b>	<b>Motion planning of linear movements requires solving inverse kinematics in order to calculate a trajectory in joint space (from the workspace trajectory). Joint movements in contrast are directly planned in joint space. Therefore, motion planning of linear movements can take seconds up to several minutes, depending on the length and orientation of the linear path and the performance of the used computer!</b>
----------	--

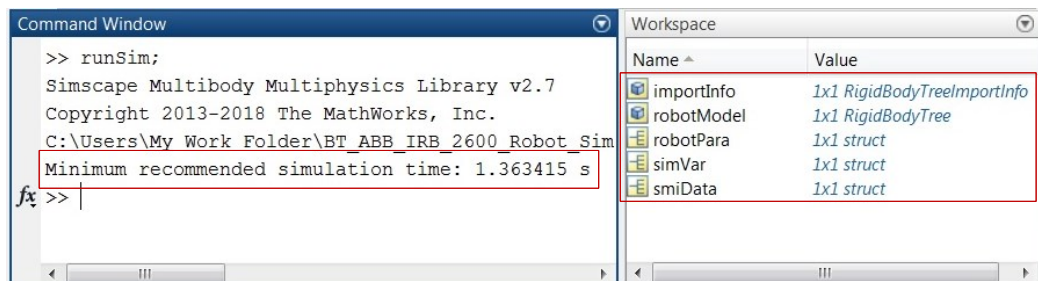
9. Make sure that the status of the corresponding entry of Panel 2, “Motion Planning” (see the figure of the “main\_ui” window below step four) is “Ready” and the “Last updated” timestamp is within a comprehensible range.
10. You may restart the procedure of motion planning to change or update the desired movement type from the “main\_ui” window as often as required.

<b>!</b>	Consider that only one movement type can be finally defined as input for the Simulink/ Simulink Simscape simulation mode (either “Joint Movement” OR “Linear Movement”). When repeating motion planning for updating or changing the movement type, former results are overwritten or deleted!
----------	--

11. To terminate the MATLAB program part and to start the Simulink/ Simulink Simscape program part, press the “Open Simulink” button of Panel 3 “Simulation” of the “main\_ui” window. Pressing the “Open Simulink” button will also cause printing the recommended minimum simulation time to the MATLAB “Command Window”. Furthermore, the main simulation variables are made visible in the MATLAB “Workspace”. The Simulink/ Simulink Simscape environment containing the simulation model will be opened and all other windows will be closed. Before pressing the “Open Simulink” button, make sure that the status of all entries of Panel 1, “Basic Data/Requirements” and one of the entries of Panel 2 “Motion Planning” are “Ready” and the “Last updated” timestamps are within a comprehensible range.

12 (36)

If not, check the MATLAB “Current Folder”, the content of the folder (see EXTENSIONS/ MODIFICATIONS (page 182)) and try to update the data using the “Update” button and repeat the procedure of motion planning.

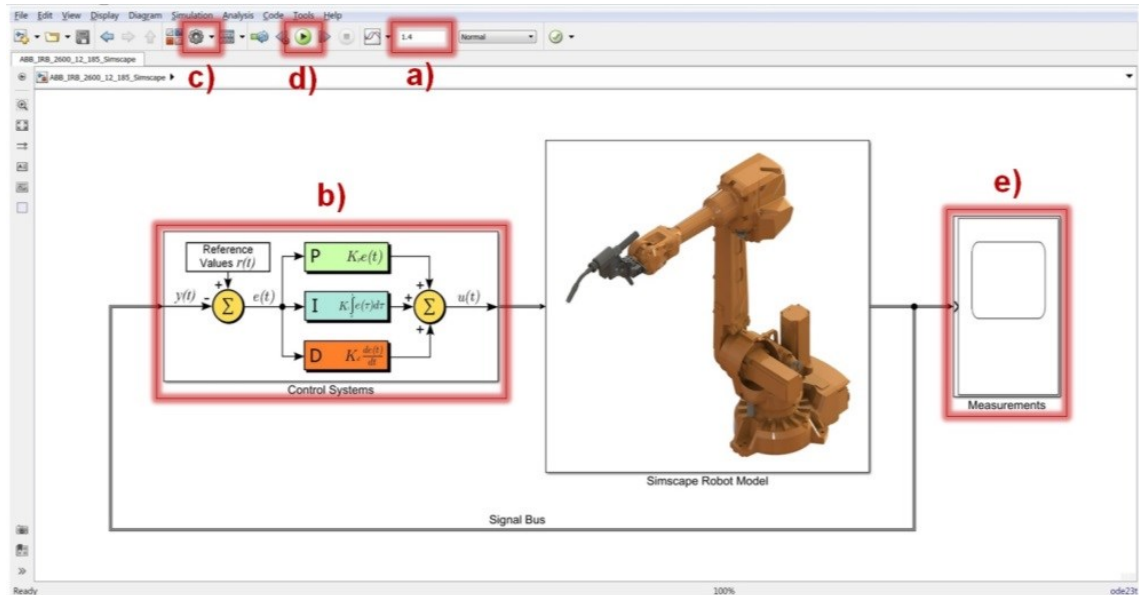


12. Continue with the instructions for SIMULINK/ SIMULINK SIMSCAPE (page 171).

## SIMULINK/ SIMULINK SIMSCAPE

13 (36)

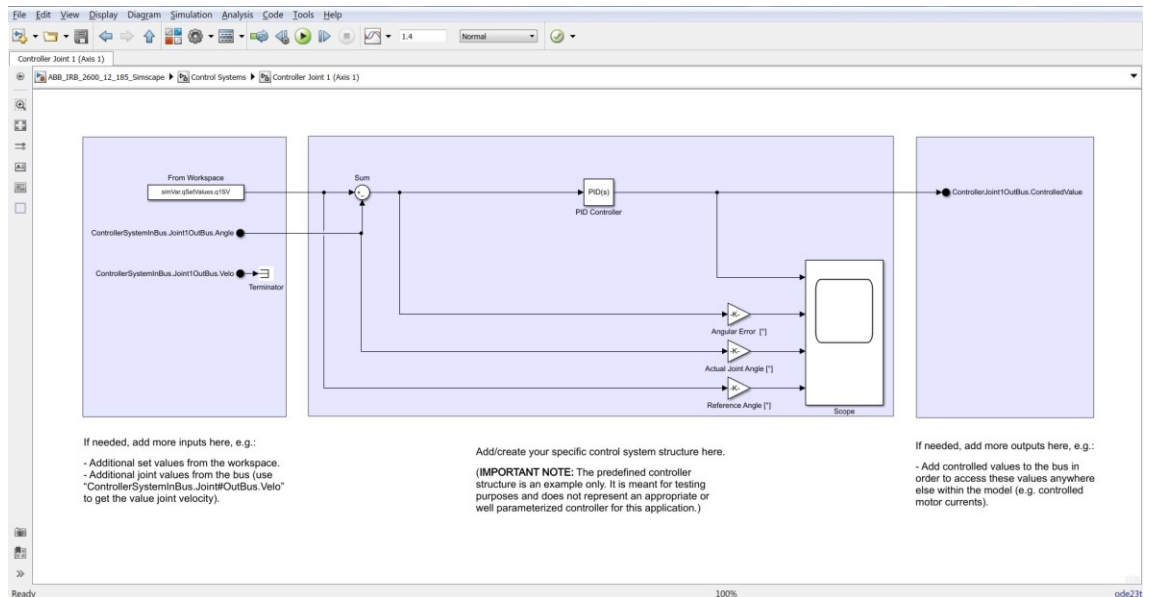
- Continuing from step eleven of the MATLAB sub section (page 166) of this section, the Simulink/ Simulink Simscape environment (window) should be visible and appear similar to:



- Type at minimum the minimum recommended simulation time, printed to the MATLAB “Command Window”, into the “Simulation stop time” field of the Simulink/ Simulink Simscape environment (see mark a) in the figure above) (you may also add a small margin).
- Enter the “Control Systems” (here exemplarily “Controller Joint 1 (Axis 1)”) subsystem of the simulation model (see mark b) in the figure above).




The “Control Systems” subsystem consists of six second level (sub subsystem) control structure subsystems – one for each joint/ axis of the robotic manipulator’s simulation model. Neglecting the individual input and output signals of the single control systems, all individual systems do have same structure. Therefore, the subsequent instructions are also valid for the joint controllers of the other axes.





4. Add/ create the desired control system design/ structure block diagram(s). Before the creation, read the information written below each of the three blue shaded (background) areas. (The control system structure must not be necessarily kept inside the areas (left: input, middle: controller, right: output). The separation is only meant as suggestion for keeping a clear structure of the block diagram(s).)

<b>!</b>	<p><b>When creating control system structures, consider that the drivers of the joint motors of the unedited Simulink/ Simulink Simscape simulation model were designed in a way to expect scalar values within the range from -1 to +1 as the controlled value (input). Thus: “+1” = “100% power in the positive direction” and “-1” = “100% power in the negative direction”.</b></p>
----------	---

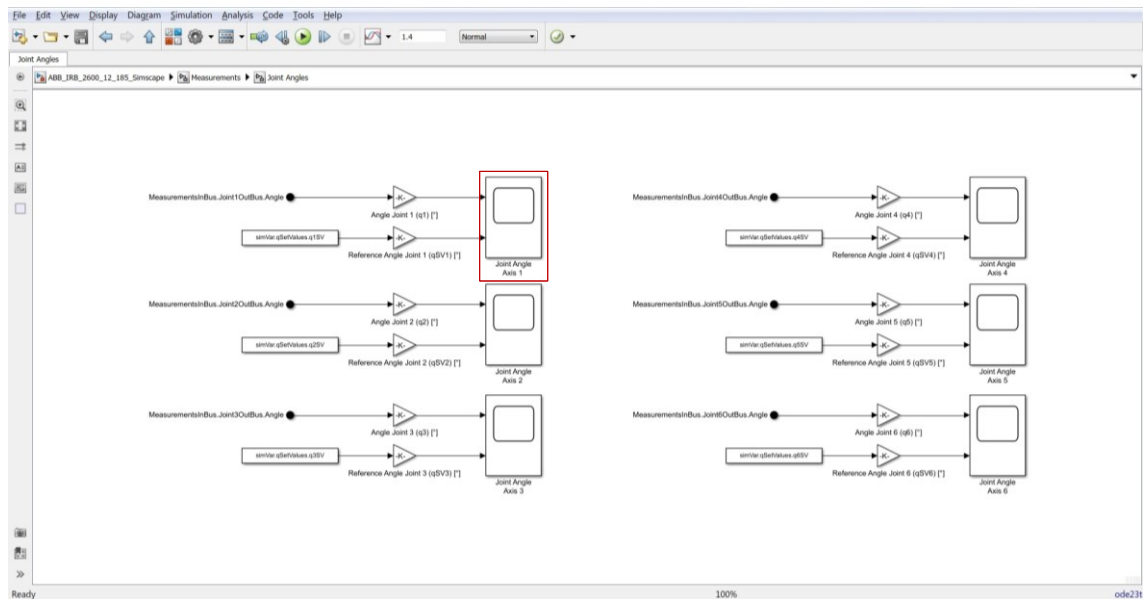
	If you wish to apply any other changes than changes of the control system structures to the Simulink/ Simulink Simscape simulation model, it is highly recommended to refer to the EXTENSIONS/ MODIFICATIONS section (page 182) in advance.
---	---

5. Repeat the procedures of step three and step four for each of the other control structure sub subsystems.
6. Navigate back to the main structure of the Simulink/ Simulink Simscape simulation model (see figure below step 1). Adjust the “Configuration Parameters” of the simulation model if necessary (e.g. solver settings) (see mark c) in the figure below step 1).
7. Run the simulation (see mark d) in the figure below step 1) and wait until the solving was completed. Alternatively, you are also able to view simulation results live while model solving is executed as explained subsequently (step eight). You may abort the model solving before finishing, e.g. when simulation results seem to be obviously faulty, in order to save time.

	Computation time may vary significantly, depending on the used simulation settings (e.g. trajectory length, solver type, solver step size, operating system and soft- and hardware).
---	--

	Always take into consideration that simulation results can be faulty and do not necessarily represent real systems behaviour.
---	---

8. Enter the “Measurements” subsystem (see mark e) in the figure below step 1). In contrast to the “Control System” subsystem, the second level subsystems of the “Measurements” subsystem are organized by the type of the measured values (joint angles, velocities, accelerations and torques) and not by the origin of the value (revolute joint blocks 1-6 of the Simulink/ Simulink Simscape simulation model). The structures of the sub subsystems are quite similar. Therefore, the subsequent exemplary instructions related to the “Joint Angles” sub subsystem are also valid for the other sub subsystems.
9. Enter the sub subsystem covering the measurements wished to be viewed (as mentioned above, here exemplarily “Joint Angles”).

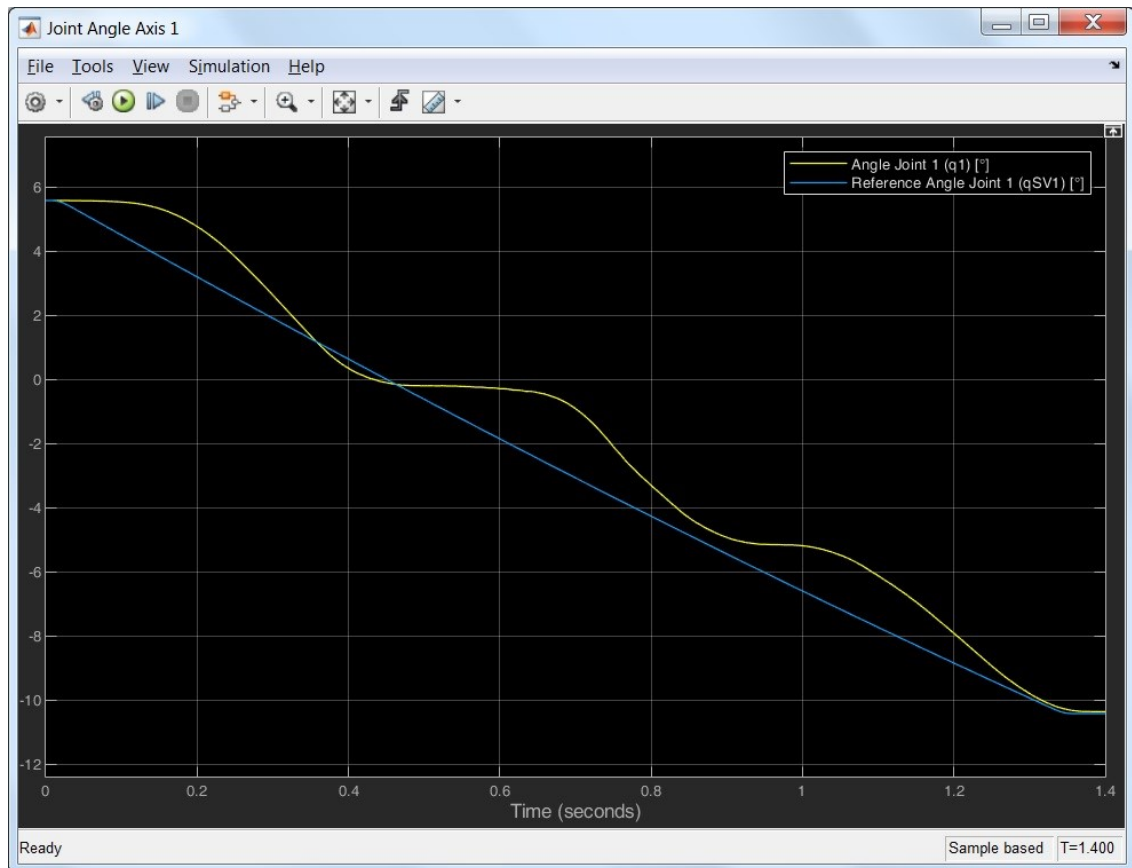


10. Measurements are taken and recorded with Simulink “Scope” blocks. As shown in the figure above, signals to be measured (actual values) are taken from the signal bus and routed to the according “Scope” block(s). In the case of the joint angles measurement, the set values of the joint angles, calculated in the MATLAB program part, are also fed to the corresponding “Scope” block(s) for comparison purposes. Both, actual and set value signals are also led through “Gain” blocks for the purpose of unit conversion.

<b>!</b>	<p><b>Consider that all bus signal values have SI units or derived SI units. For increasing the comprehensibility of the measured values, signals of the “Joint Angles”, “Joint Velocities” and “Joint Accelerations” sub subsystems are converted from the unit radian [rad] to the unit degree [°], whereas in the “Joint Torques” sub subsystem the unit [Nm] is measured. You may undo the predefined unit conversion by changing the gain values of the preceding “Gain” blocks of each scope input signal to one (1).</b></p>
----------	---

11. Double-click the “Scope” block to view the graphs of the measured input signals (see figure above, here exemplarily “Joint Angle Axis 1”).



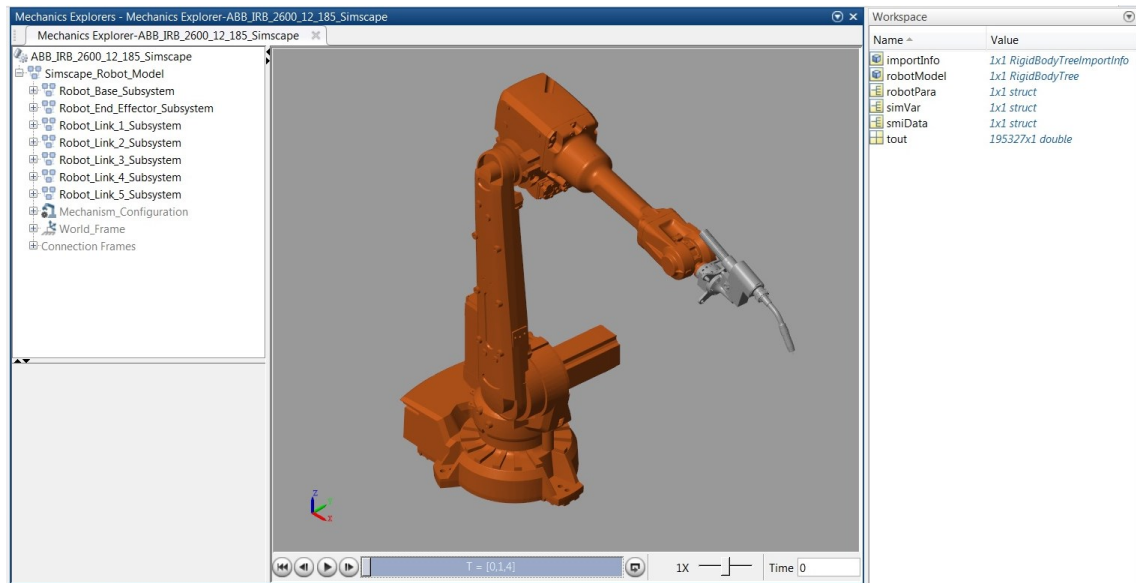


12. Evaluate the measurement result(s). Print and/ or save the graph(s)/ result(s) if necessary. Also check all other measurements of interest.

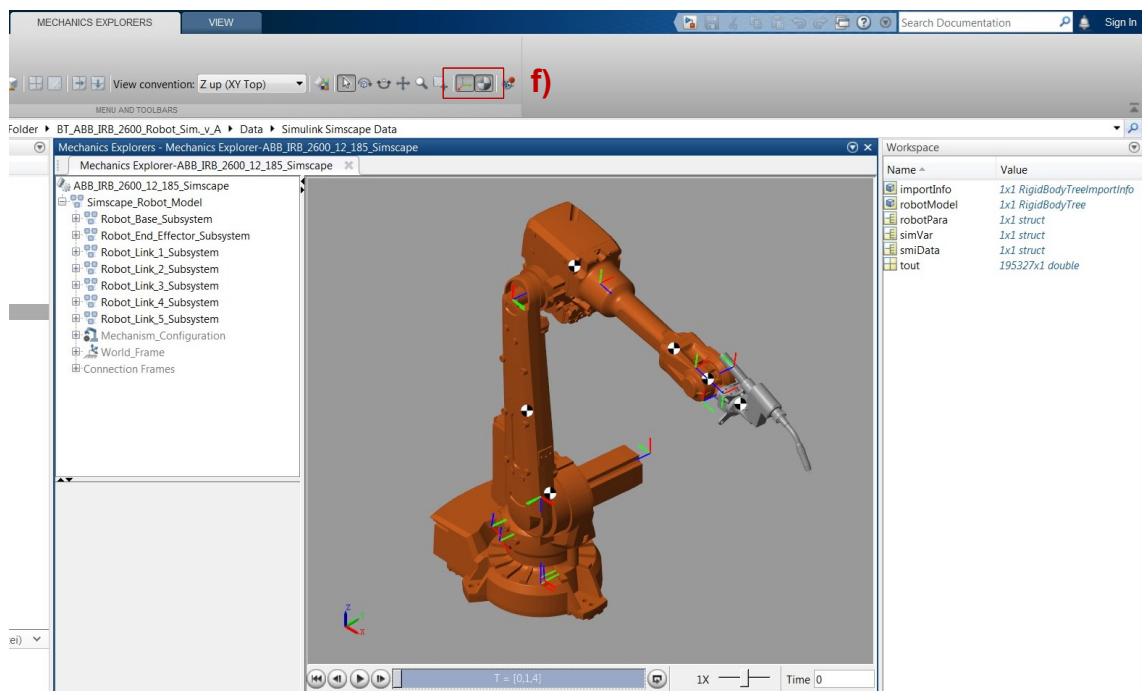


Change the predefined appearance of the scope window to align it with your own requirements. E.g. go to “View”, “Layout” and “2x1” to create two separated signal graphs with the same time axis inside the graph window.

13. To view the 3D animation/ simulation of the Simulink Simscape simulation model, represented by the .stl geometry files gained from the CAD model and the calculated kinematics and dynamics, change from the Simulink/ Simulink Simscape environment window to the MATLAB environment window. The opened window should appear like shown below, if not, change the tab of the MATLAB window to “MECHANICS EXPLORERS” manually.



You may enable the visibility of the frames and the centers of masses of the links (rigid bodies) of the Simulink Simscape simulation model (see figure below, use the marked buttons of the “MECHANICS EXPLORERS” (mark f)).



14. Evaluate the results and save them if needed. To restart the MATLAB program part in order to change parameters or motion planning, repeat the procedure of the MATLAB subsection of this section (page 166) starting from step one. If you wish to terminate the simulation program, just close MATLAB and MATLAB Simulink as usual.

#### 4. CHANGE OF PARAMETERS

19 (36)

All parameters of the simulation model are stored in an external Microsoft Excel spreadsheet in order to provide a centralized and clearly arranged parameter compilation. All required parameters are imported from the spreadsheet during the initialization of the simulation program and are provided to the Simulink/ Simulink Simscape simulation model via the MATLAB “Workspace” (base). Therefore, do not change any variable entries within the Simulink/ Simulink Simscape block(s) (diagrams) settings to change parameters.

The full name of the Microsoft Excel spreadsheet is:

[ABB\\_IRB\\_2600-12-1.85\\_Parameters.xlsx](#)

The spreadsheet can be found from the relative file path:

[../BT\\_ABB\\_IRB\\_2600\\_Robot\\_Sim.\\_v\\_A\Data\Robot Parameters\](#)

[ABB\\_IRB\\_2600-12-1.85\\_Parameters.xlsx](#)

The general structure and contents of the spreadsheet are listed in the subsequent table:

Sheet No.:	Sheet Name:	Content(s)/ Purpose(s):
1	(1) General Robot Information	Handling capacity, reach, weight
2	(2) Axis Range Limits	General (angular) axis limitations (A1-A6)
3	(3) Axis Speed Limits	General axis angular velocity limitations (A1-A6)
4	(4) Axis Acceleration Limits	General axis angular acceleration limitations (A1-A6)
5	(5) TCP Limits	General TCP velocity and acceleration limitations
6	(6) Joint Parameters	(Revolute) joint(s) parameters: State targets (position, velocity), internal mechanics (equilibrium pos., spring stiffn., damping coeff.), bearings (friction torques, damping coeff.)
7	(7) Motor Parameters	Electrical motor(s) (asynchronous machine (ASM) with squirrel cage rotor (three-phase)) parameters: El. ratings (power, voltage etc.), el. parameters (stator resistance, reactance, etc.) and mechanical parameters (rotor inertia, etc.)
8	(8) Transmission Parameters	Cycloidal transmission (gear box) parameters: teeth numbers (gear ratio), efficiency, inertia, etc.
9	(9) Motor Drivers Parameters	Six-pulse three phase converter parameters: DC link voltage, switching freq., sample time, etc.

<b>!</b>	When simplifying parameters e.g. by adding several viscous damping and/ or friction coefficients to one overall value, always check the corresponding application(s)/ block(s) from the Simulink/ Simulink Simscape simulation model in advance (e.g. different angular velocities due to transmission gears)! This is also valid for inertias where gear ratios typically cause squared impacts (e.g. $r = 130, \rightarrow r^2 = 16900!$ ).
----------	---

<b>!</b>	It is strongly recommended not to change the structure (this includes the exact names and sequences of the sheets, cell locations, etc.) of the spreadsheet. Own sheets can be added after the last original sheet "(9) Motor Drivers Parameters".
----------	--

1. Navigate to the (relative) file path of the spreadsheet mentioned above. Save a copy of the unedited spreadsheet to any proper location before continuing.
2. Open the spreadsheet and navigate to the sheet(s) containing the parameter(s) to be changed (check the structure and contents table above).

ABB IRB 2600-12/1.85													
(Revolute) Joint Parameters				State Targets			Internal Mechanics			Bearing			
n:	Description:	Corresponding Axis:	Base (B):	Follower (F):	Position Target Value [rad]:	Velocity Target Value [rad/s]:	Equilibrium Position [rad]:	Spring Stiffness [N*m/rad]:	Damping Coefficient [N*m/(rad*s)]:	Breakaway Friction Torque [N*m]:	Breakaway Friction Velocity [rad/s]:	Coulomb Friction Torque [N*m]:	Viscous Friction Coefficient [N*m/(rad*s)]:
1	Joint 1	Axis 1	Base	Link 1	0,0000	0,0000	0,0000	0,0000	0,0000	0,000001	0,000001	0,000001	0,000001
2	Joint 2	Axis 2	Link 1	Link 2	0,0000	0,0000	0,0000	0,0000	0,0000	0,000001	0,000001	0,000001	0,000001
3	Joint 3	Axis 3	Link 2	Link 3	0,0000	0,0000	0,0000	0,0000	0,0000	0,000001	0,000001	0,000001	0,000001
4	Joint 4	Axis 4	Link 3	Link 4	0,0000	0,0000	0,0000	0,0000	0,0000	0,000001	0,000001	0,000001	0,000001
5	Joint 5	Axis 5	Link 4	Link 5	0,0000	0,0000	0,0000	0,0000	0,0000	0,000001	0,000001	0,000001	0,000001
6	Joint 6	Axis 6	Axis 6	Link 6 (End effector)	0,0000	0,0000	0,0000	0,0000	0,0000	0,000001	0,000001	0,000001	0,000001

3. Apply the desired changes - always consider the corresponding unit(s) and cell format (number, text, etc.)!
4. Save all changes (overwrite; do not change the file path or file name) and close the spreadsheet.
5. Run the simulation (see section OPERATION (page 166)) and check if the values of the `robotPara` variable (see section EXTENSIONS/ MODIFICATIONS (page 182)) are in accordance with the applied changes.
6. If any error(s) occur(s), firstly check the section TROUBLESHOOTING (page 193). Then repeat the procedure of this section starting from step one.

## 5. CAD MODEL UPDATE

21 (36)

This section covers instructions for updating the MATLAB and Simulink/ Simulink Simscape data of the simulation model based on the CAD model. The update procedure needs to be applied after each change of the CAD model (assembly) of the robotic manipulator.

<b>i</b>	Always check if desired changes could possibly be implemented without changing and updating the CAD model in advance. E.g. adding simple geometries or changing the mass of a robotic manipulator's link and/ or the end effector/ tool can be accomplished within the MATLAB and/ or Simulink/ Simulink Simscape environment. For further information refer to the section EXTENSIONS/ MODIFICATIONS (page 182).
----------	---

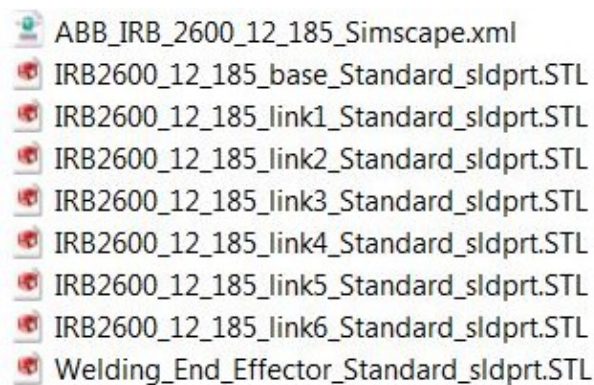
<b>!</b>	When applying changes to the CAD model (assembly), always consider that the Simulink Simscape block diagrams general structure is derived from the CAD model structure but cannot be adapted automatically. Therefore, do not change the existing CAD model constraints between in the links (revolute), or link 6 and the end effector/ tool (rigid). Furthermore it is strongly recommended not to change the overall number of individual parts (seven) of the CAD model assembly (e.g. when adding a secondary payload to one of the robotic manipulator's links, use "unite" or similar functions in order to obtain one rigid body). For a complete listing of the structure (including constraints, naming and data types) of the CAD model assembly refer to the corresponding thesis document (section 6.1, TABLE 6.1 and TABLE 6.2).
----------	--

1. Make sure that you are using one of the suitable CAD software listed in the first table of PREREQUISITES (page 163, table of requirements).
2. If the Simscape Multibody Link Plug-In is not already installed to your CAD software, go to <https://www.mathworks.com/help/phymod/smlink/ug/installing-and-linking-simmechanics-link-software.html><sup>14</sup> and follow the instructions carefully.
3. Run the CAD software and apply the desired changes to the robotic manipulator's individual parts and/ or assembly.

---

<sup>14</sup> The MathWorks Inc. 2019. Support. Documentation: Simscape Multibody Link Plug-In. Install the Simscape Multibody Link Plug-In. Release: R2019a. Read on 23.03.2019

4. Create an empty folder: *Simulink Simscape Data* (exact name) and save it to any proper directory.
5. Create and export the Simscape Multibody model (*.xml* file) from your CAD software to the *Simulink Simscape Data* folder created in step four, using the Simscape Multibody Link Plug-In. (Use *.stl* geometry file format). For assistance, explore <https://www.mathworks.com/help/physmod/smlink/index.html><sup>15</sup>. The content of the *Simulink Simscape Data* folder should now look like shown below:



6. In MATLAB, navigate to the (relative) file path *../BT\_ABB\_IRB\_2600\_Robot\_Sim.\_v\_A\Data\Simulink Simscape Data*
7. Save all files shown in the MATLAB “Current Folder” sub window to any proper folder and directory in order to create backup data.
8. Rename the *ABB\_IRB\_2600\_12\_185\_Simscape\_DataFile.m* file to *oldDataFile.m*.
9. Delete all files shown in the MATLAB “Current Folder” sub window except *oldDataFile.m* and *ABB\_IRB\_2600\_12\_185\_Simscape.slx*.
10. Copy all newly created files from the *Simulink Simscape Data* folder (step five) to the MATLAB “Current Folder”.

---

<sup>15</sup> The MathWorks Inc. 2019. Support. Documentation: Simscape Multibody Link Plug-In. Release: R2019a. Read on 23.03.2019

11. Type `smimport('ABB_IRB_2600_12_185_Simscape.xml',...  
'ImportMode','dataFile','DataFileName',...  
'ABB_IRB_2600_12_185_Simscape_DataFile',...  
'PriorDataFile','oldDataFile.m');` to the MATLAB “Command Window” and press “Enter”.

<b>!</b>	<b>Never use 'modelAndDataFile' instead of 'dataFile' as the value for the 'ImportMode' input argument to call <code>smimport()</code>! This would cause an update of the complete Simulink/ Simulink Simscape block diagram and not only its data set!</b>
----------	---

12. Wait until MATLAB finished the procedure. Read the procedure report printed to the MATLAB “Command Window” carefully and follow the instructions if necessary.
13. Check if the new [ABB\\_IRB\\_2600\\_12\\_185\\_Simscape\\_DataFile.m](#) file exists in the MATLAB “Current Folder” sub window.
14. Delete the [oldDataFile.m](#) file via the MATLAB “Current Folder” sub window.
15. Open the [ABB\\_IRB\\_2600\\_12\\_185\\_Simscape.slx](#) Simulink/ Simulink Simscape simulation model and check for any errors.
16. If any error(s) occur(s), firstly check the TROUBLESHOOTING section (page 193). Then repeat the procedure of this section starting from step one. You may restore the original [Simulink Simscape Data](#) folder from the backup data created in step seven if the error(s) cannot be solved.

## 6. EXTENSIONS/ MODIFICATIONS

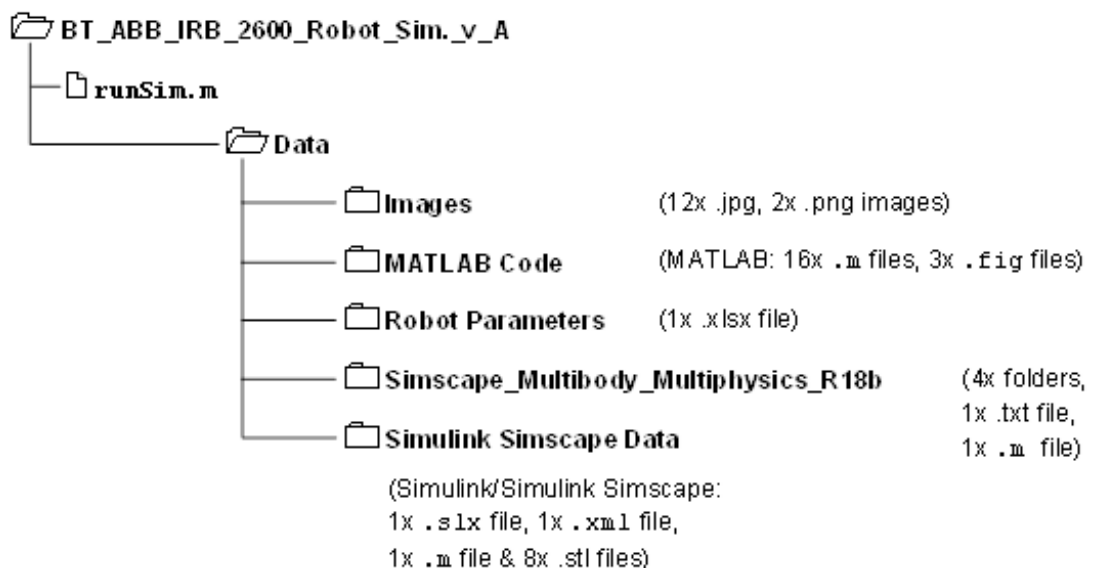
24 (36)

The section EXTENSIONS/ MODIFICATIONS provides useful hints and general information concerning the structure and flow of the complete simulation program (covering the MATLAB and the Simulink/ Simulink Simscape parts). Because of the wide variety of applicable extensions/ modifications, this section does not contain any specific instruction sequences. Several example extensions/ modifications are shown at the end of this section.

<b>!</b>	Adding, removing or modifying any data of the original data set can cause fatal errors. Always save a copy of the original data separately before applying any changes. Save your work periodically.
----------	--

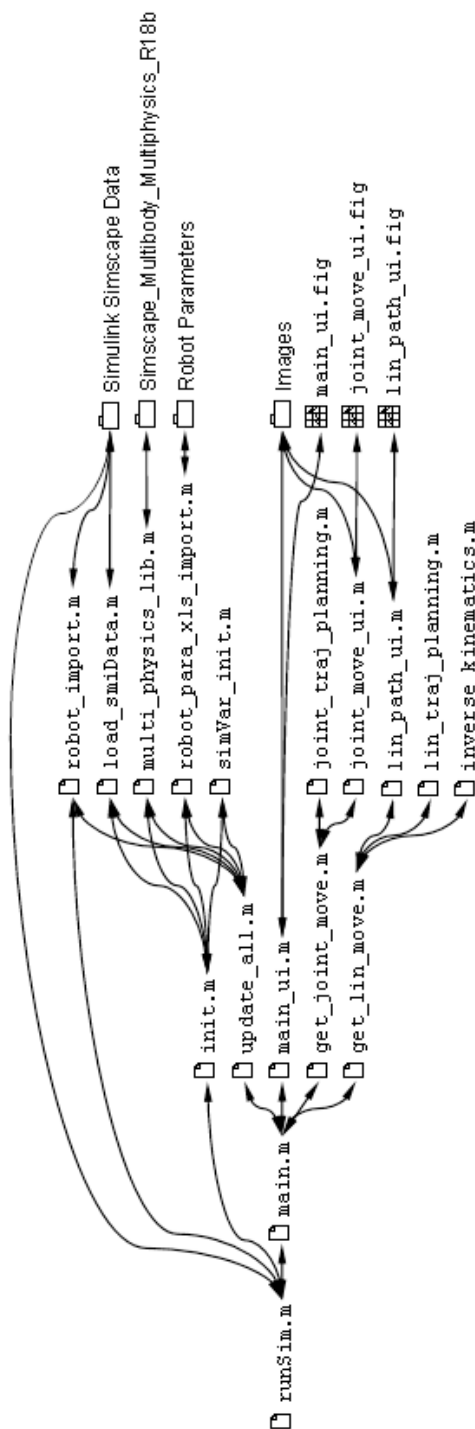
Before applying any changes to the simulation program, it is recommend to firstly get familiarized with its overall structure in general and its single components in particular. Therefore, read the sections INTRODUCTION (page 164) and CHANGE OF PARAMETERS (page 177) in advance. You may also check the program flow charts of the MATLAB .m files from the appendices of the corresponding thesis document (Appendix 4. Program Flow Charts).

The general structure of the simulation programs data set folder is shown in the subsequent figure. (In the context of TROUBLESHOOTING (page 193), you may also ensure the completeness of the data set.)






The structure, dependencies and interactions of the MATLAB .m files and external data (within the data set) are depicted in the figure below. The directions of connections refer to the real data flow (from the left to the right: calls, from the right to the left: returns). Check the contents of the MATLAB .m files or the program flow charts from the appendices of the corresponding thesis document (Appendix 4. Program Flow Charts) to learn more about the input and output variables/ arguments of each function/ .m file.



As mentioned in the INTRODUCTION (page 164), all required data of/ for the simulation (program) are stored to five main variables which are made visible/ accessible in the MATLAB “Workspace” (base) with the transition from the MATLAB to the Simulink/ Simulink Simscape program part. Consider the subsequent tables and explanations in order to obtain more detailed information concerning the main simulation variables.

Name:	Type:	Description/Purpose:	Initialized/ Changed by:
simVar	1x1 struct (9 fields)	Contains all required data for the execution of the MATLAB program part. Provides the results of the MATLAB program part to the Simulink/ Simulink Simscape program part.	Initialized by <code>simVar_init()</code> , changed by all other functions of the MATLAB program part
smiData	1x1 struct (3 fields)	Contains the block parameter values of the imported Simscape Multibody simulation model automatically created during the procedure of the execution of the <code>smimport()</code> function.	Created by <code>smimport()</code> , initialized by <code>load_smiData()</code>
robotPara	1x1 struct (7 fields)	Contains mainly values for the parameterization of the block(s) (diagram(s)) of the Simulink/ Simulink Simscape simulation model.	<code>robot_para_xls_import()</code>
robotModel	1x1 Rigid- BodyTree	Contains the robotic manipulator's simulation model kinematic structure (represented by rigid bodies connected by joints) and corresponding parameters.	<code>importrobot()</code>
importInfo	1x1 Rigid- BodyTree- ImportInfo	Contains information concerning the import procedure of the <code>importrobot()</code> function.	

The variables `robotModel` and `importInfo` are the returning values/ variables/ objects of the MATLAB `importrobot()` function. If required, further information concerning the `importrobot()` function can be obtained from <https://www.mathworks.com/help/robotics/ref/importrobot.html><sup>16</sup>. General information concerning the `robotModel` and `importInfo` variables can also be found there. To get more detailed information about the `robotModel` (`RigidBodyTree`) variable, go to <https://www.mathworks.com/help/robotics/ref/robotics.rigidbodytree-class.html><sup>17</sup>. It is strongly recommended to not to apply any changes to these variables manually.

	Type <code>show(robotModel);</code> to the MATLAB “Command Window” and press “Enter” to view a 3D plot of the robot’s general structure. Use <code>showdetails(importInfo);</code> in the same manner to print the contents of the <code>importInfo</code> variable to the MATLAB “Command Window” in a readable format.
---	--

The `smiData` variable is loaded from the [ABB\\_IRB\\_2600\\_12\\_185\\_Simscape\\_DataFile.m](#) file using the `load_smiData()` function. The [ABB\\_IRB\\_2600\\_12\\_185\\_Simscape\\_DataFile.m](#) in turn is a model data file derived from the Simulink Simscape Multibody Import .xml file ([ABB\\_IRB\\_2600\\_12\\_185\\_Simscape.xml](#)) using the `smimport()` function (for more information refer to section CAD MODEL UPDATE (page 179)).

The `simVar` and `robotPara` variables were designed from the author in the context of the accomplishment of the corresponding thesis work/ document. The `robotPara` variable is created by the `robot_para_xls_import()` function which in turn reads values from the [ABB\\_IRB\\_2600-12-1.85\\_Parameters.xlsx](#) spreadsheet file.

<sup>16</sup> The MathWorks Inc. 2019. Support. Documentation: Robotics System Toolbox. Manipulator Algorithms. Functions. `Importrobot`. Release: R2019a. Read on 24.03.2019

<sup>17</sup> The MathWorks Inc. 2019. Support. Documentation: Robotics System Toolbox. Manipulator Algorithms. Classes. `robotics.RigidBodyTree` class. Release: R2019a. Read on 24.03.2019

The `simVar` variable is initialized by the `simVar_init()` function and received and returned from all functions of the MATLAB program part in order to allow all functions to read/ write information from/ to one centralized variable. See the tables below to learn more about the structures and contents of the `simVar` and `robotPara` variables.

Variable:	Fields (First Level):	Description/ Purpose:
simVar	uiInput	Contains further subfields and sub subfields; contains inputs of the graphical user interfaces “joint_move_ui()” and “lin_path_ui()”.
	uiControl	Contains further subfields (e.g. <code>exeUpdate</code> ); for control functionalities of the main graphical user interface “main_ui”.
	statusFlags	Contains (flag-) values (either “1” = “true” or “0” = “false”); for the interaction/ control functionalities between the different graphical user interfaces.
	updateTime	Contains the update times of updated/ loaded / created/ executed data/ libraries/ programs (e.g. Simscape Multibody Multiphysics Library) for the “Last updated:” labels in the “main_ui” GUI window.
	linPathPlan	Contains further subfields (e.g. <code>pRes</code> ); contains the results of the linear trajectory planning <code>lin_traj_planning()</code> (for <code>get lin move()</code> internal use).
	initVal	Contains further subfields (e.g. <code>qStartA</code> ); contains the initial pose (and velocities) of the Simulink/ Simulink Simscape simulation model.
	targetVal	Contains further subfields (e.g. <code>qTargetB</code> ); contains the target pose (and velocities) of the Simulink/ Simulink Simscape simulation model.
	gik	Contains a further subfield ( <code>qRes</code> ); stores the (unformatted) results of the inverse kinematics ( <code>inverse_kinematics()</code> ).
	qSetValues	Contains further subfields ( <code>q1SV...q6SV</code> ); contains the (formatted) set values of the joint angles for the Simulink/ Simulink Simscape simulation model.

Variable:	Fields (First Level):	Description/ Purpose:
robotPara	generalRo- eralRo- botInfo	Contains further subfields (e.g. capacity); contains general information of the real robotic manipulator.
	axisLim- its	Contains further subfields (e.g. range); contains axis/ joint limitations of the Simulink/ Simulink Simscape simulation model equal to the axis/ joint limitations of the real robotic manipulator (e.g. for input filtering in “joint_move_ui()” and “lin_path_ui()”).
	tcpLimits	Contains values (e.g. velocity); contains TCP limitations of the Simulink/ Simulink Simscape simulation model equal to the TCP limitations of the real robotic manipulator(e.g. for input filtering “joint move ui()” and “lin path ui()”).
	jointPara	Contains further subfields and sub subfields (e.g. stateTar); contains values for the parameterization of the revolute joint block(s) (diagram(s)) of the Simulink/ Simulink Simscape simulation model.
	motorPara	Contains further subfields and sub subfields (e.g. ratPow); contains values for the parameterization of the joint motor/ driver block(s) (diagram(s)) of the Simulink/ Simulink Simscape simulation model.
	transmPa- ra	Contains further subfields and sub subfields (e.g. nCdt); contains values for the parameterization of the joint transmission block(s) (diagram(s)) of the Simulink/ Simulink Simscape simulation model.
	motDrivPa ra	Contains further subfields and sub subfields (e.g. vdc); contains values for the parameterization of the joint motor driver block(s) (diagram(s)) of the Simulink/ Simulink Simscape simulation model.

It is recommended to only read the original variables structures and not to apply any changes to them. If required, add your own structures/ entries to the variables without overwriting the existing contents.



Use the MATLAB “Workspace” sub window to view the variables contents by double-clicking them.

### Example 1: Modification of Robot Model (Links (Bodies)/ Joints) Properties

As mentioned in section CAD MODEL UPDATE (page 179), some simple changes of the properties (links (solids/ rigid bodies) masses, centers of masses, inertia properties, joint properties, rigid transformations, etc.) of the robotic manipulator's simulation model can be applied without changing and updating the CAD model. In this example, the change of the mass of the end effector/ tool of the simulation model is shown exemplarily.


- Use the MATLAB “Current Folder” sub window or the “Address Field” to navigate to the relative path: [../BT\\_ABB\\_IRB\\_2600\\_Robot\\_Sim.\\_v\\_A\Data\Simulink Simscape Data](#).
- Open the ABB\_IRB\_2600\_12\_185\_Simscape\_DataFile.m file from the MATLAB “Current Folder” sub window by double-clicking.
- Change to the MATLAB “Editor” sub window to view the code of the ABB\_IRB\_2600\_12\_185\_Simscape\_DataFile.m file. Search for the code line: `smiData.Solid(6).ID = 'Welding_End_Effector*:*Standard';` (line 196 in this case, see mark a) in the figure below). (Consider that the end effector body is represented by the solid body with the index six (6).)

```


187
188 %Inertia Type - Custom
189 %Visual Properties - Simple
190 - smiData.Solid(6).mass = 5; % kg b)
191 - smiData.Solid(6).CoM = [-6.598118590094697 89.202208174451954 112.75924894509187]; % mm
192 - smiData.Solid(6).MoI = [11039.93918588447 1894.5671108729034 11790.502464329467]; % kg*mm^2
193 - smiData.Solid(6).PoI = [414.19741541277642 89.323461907090248 -2574.7165246762947]; % kg*mm^2
194 - smiData.Solid(6).color = [0.75294117647058822 0.75294117647058822 0.75294117647058822];
195 - smiData.Solid(6).opacity = 1;
196 - smiData.Solid(6).ID = 'Welding_End_Effector*:*Standard'; a)
197


```

- Go to `smiData.Solid(6).mass` (line 190) and apply the desired changes (see mark b) in the figure above) (consider the corresponding unit written as comment behind the value).
- Save (overwrite) the applied changes.

	When changing e.g. the center of mass of a body (e.g. smiData.Solid(6).CoM, line 191 in the figure above) always check the corresponding origin of this value from the Simulink Simscape simulation model block diagram in advance.
---	---

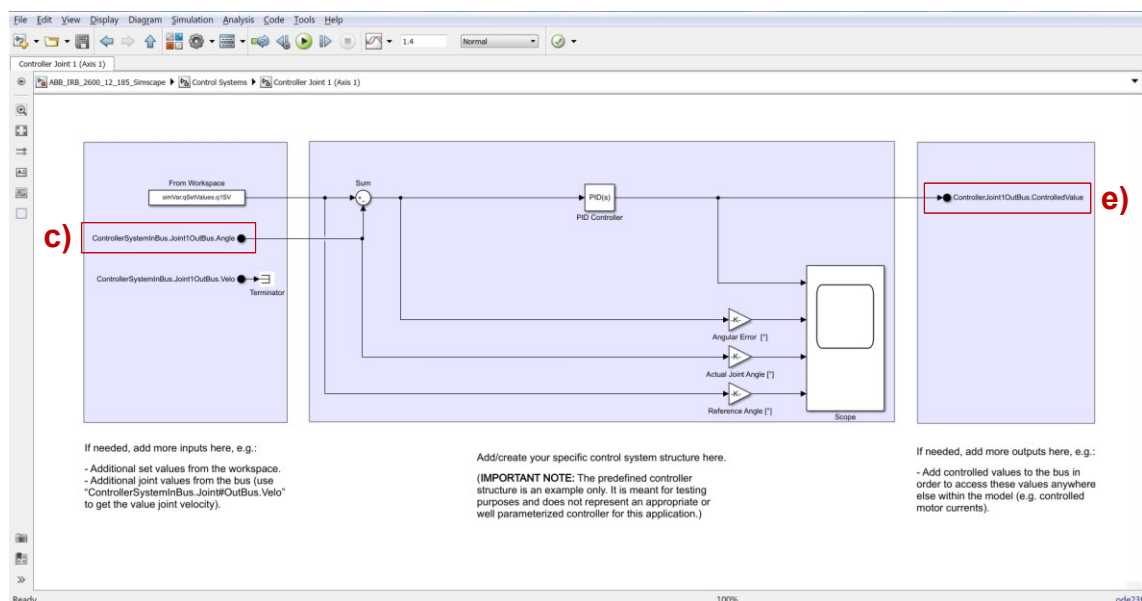
## Example 2: Adding/ branching off Signal Bus Signals

	Within the Simulink/ Simulink Simscape simulation model, all signals are routed with the help of a signal bus system (exception: values/ parameters directly or indirectly (“From Workspace” block) obtained from the MATLAB “Workspace” (base)).
---	---

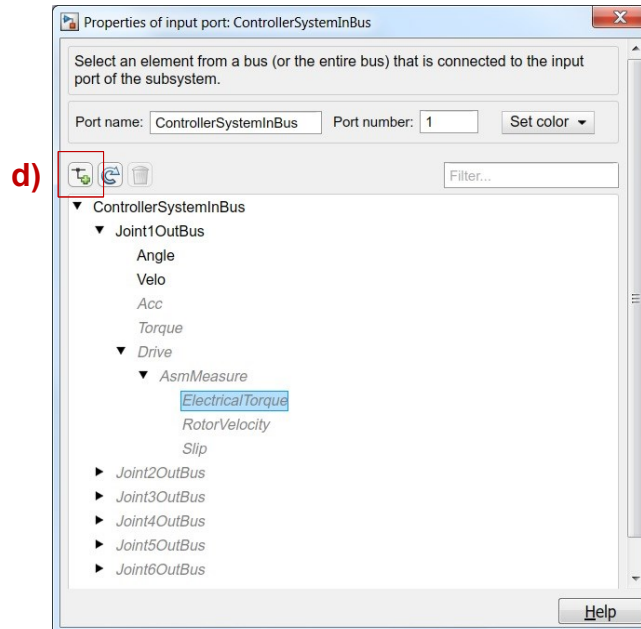
	Simulink and Simulink Simscape use different signal domains. Always use “PS-Simulink Converter” or “Simulink-PS Converter” blocks for interfacing when adding or branching off bus signals (bus signals are in the Simulink signal domain and use SI units or derived SI units only).
---	---

The example adding/ branching off signal bus signals is exemplarily shown for adding/ branching off signal bus signals within the “Controller Joint 1 (Axis 1)” subsystem.

- Within the Simulink Simscape simulation model environment/ window, navigate to the “Control Systems” subsystem. Then enter the sub subsystem “Controller Joint 1 (Axis 1)”:



- For branching off signals from the signal bus, double-click an existing “Bus Element In” block (here “ControllerSystemInBus.Joint1OutBus.Angle” exemplarily, see mark c) in the figure above).



- Search for the desired signal in the bus structure depicted in the opened window. Mark the signal(s) to be added (here “ElectricalTorque” exemplarily) and press the “Add blocks for selected signals” button, see mark d) in the figure above). The new “Bus Element In” block(s) will appear in the block diagram underneath the existing one(s). Alternatively, you can add “Bus Element In” blocks from the Simulink “Library”.
- For adding any signal(s) to the signal bus, double-click an existing “Bus Element Out” block (here “ControllerJoint1OutBus.ControlledValue” exemplarily, (see mark e) in the second last figure). A window will appear, similar to the one shown in the figure above. Instead of the “Add blocks for selected signals” button, now press the “Add a new signal” button (same appearance). A new signal will be added to the bus structure depicted in the opened window. Furthermore, a new “Bus Element Out” block will be added in the block diagram underneath the existing one. Apply appropriate naming to the added bus signal. Alternatively, you can add “Bus Element Out” blocks from the Simulink “Library”.

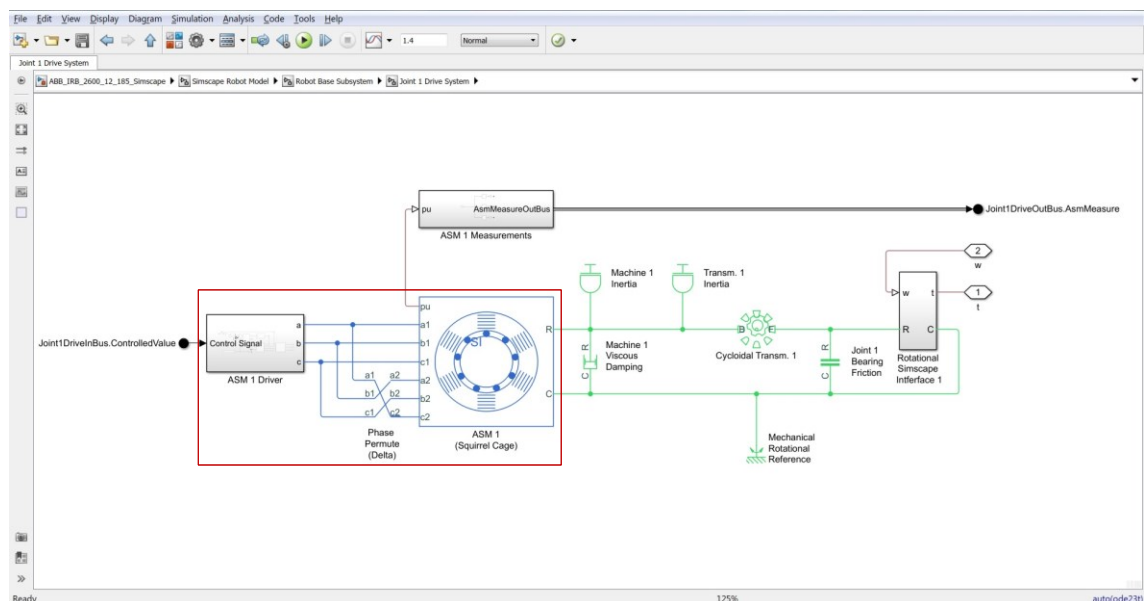


**Example 3:** Change of the Joint Motor and Motor Driver


33 (36)

Subsequently, the change of the motor and the motor driver of Joint 1 of the Simulink Simscape simulation model is shown exemplarily.

- Within the Simulink Simscape simulation model environment/ window, firstly navigate to the “Simscape Robot Model” subsystem. Secondly, enter the sub subsystem “Robot Base Subsystem”, following that, enter the third level subsystem “Joint 1 Drive System”.



- Delete the “ASM 1 Driver”, “Phase Permute (Delta)” and “ASM 1 (Squirrel Cage)” blocks (see the marked area in the figure above). If not required for own signal routing purposes, also delete the “ASM 1 Measurements” and “Joint1DriveOutBus.AsmMeasure” blocks.
- Create/ Insert the new motor model (and driver if required) and connect it to the existing mechanical rotational conserving lines “R” (rod) and “C” (case).

	<p>Only signal(s)/ block(s) from the same Simulink Simscape domain can be connected directly (e.g. “Driveline”, “Electrical”, etc.) (consider the different colours of the signals of the different domains). For interfacing, use “Interface” blocks (e.g. “Rotational Simscape Intftt” block) to be found from the Simulink “Library” -&gt; Simscape Multibody Multiphysics Library.</p>
---	--

- Use the existing “Machine 1 Inertia” and “Machine 1 Viscous Damping” blocks for representing the mechanical behaviour/ properties of the motor. Therefore, change the corresponding parameters in the [ABB\\_IRB\\_2600-12-1.85\\_Parameters.xlsx](#) spreadsheet (see section CHANGE OF PARAMETERS (page 177)). To disable the mentioned predefined blocks, e.g. when neglecting mechanical influences of the motors on the simulation model, set the specific parameters to zero. (Setting the specific parameters to zero can possibly cause various errors – try to use small values near zero (e.g. 1E-12) to avoid these errors whenever occurred.)


Alternatively, you can delete the variable entries of the blocks settings and type in the desired values directly.

Consider adapting the control system structures, to be found from the “Control Systems” subsystem, in order to align the type(s) of the controlled value(s) with the expected input(s) of the new motor and its driver (in this context, also refer to step ten of the sub section SIMULINK/ SIMULINK SIMSCAPE of the section OPERATION (page 171)).

## 7. TROUBLESHOOTING

35 (36)

The TROUBLESHOOTING section is meant for identifying and fixing errors occurred in the context of using an unedited or an edited version of the simulation program. Because of the high number of possible mistakes/ errors/ faults, only a few typical, common and very specific, matters can be handled within this section. Check the entries listed in the subsequent table whenever unknown errors occurred and apply all applicable solutions/ corrections.

	Also explore <a href="https://www.mathworks.com/help/">https://www.mathworks.com/help/</a> for further help. (The MathWorks Inc. 2019. Support. Documentation. Release: R2019a. Read on 23.03.2019)
---	---

Description(s)/ Error(s)/ Fault(s):	Explanation(s)/ Solution(s):
For an unknown reason, I cannot use the GUI and/ or close any window(s) and/ or exit the simulation program.	Go to the MATLAB “Command Window”, place the cursor at any position inside and press “CTRL+D”. This should terminate all active MATLAB tasks.
I changed the robot parameters in the robot parameters spreadsheet. The changes are not applied to the MATLAB or/ and Simulink/ Simulink Simscape environment(s).	Consider that the MATLAB program needs to be run again after applying changes to the robot parameters. Use the “Update” button in the “main_ui” or restart the program from the MATLAB “Command Window” using <code>runSim;</code> . Make sure that you saved the applied changes by overwriting the existing spreadsheet. Do not use “Save as”.
I changed the robot parameters in the robot parameters spreadsheet. MATLAB or/ and Simulink/ Simulink Simscape now report various errors.	Check the applied changes for typos, incorrect separators, incorrect units and incorrect cell formats.
I changed values in MATLAB or/ and Simulink/ Simulink Simscape environment(s) manually. Now various errors are reported (e.g. exceeded variable boundaries).	Ensure that the correct separators were used. Example: one point zero five: Incorrect: 1,05 Correct: 1.05
Motion planning of linear movements takes a lot more time and computational efforts than the motion planning of joint movements.	Motion planning of linear movements requires solving inverse kinematics in order to calculate a trajectory in joint space (from the workspace trajectory). Joint movements in contrast are directly planned in joint space. Therefore, motion planning of linear movements requires more computational time and efforts.

Description(s)/ Error(s)/ Fault(s):	Explanation(s)/ Solution(s):
<p>Compiling and solving the Simulink/ Simulink Simscape simulation model is very slow and causes high CPU, RAM and disk usage.</p>	<p>The simulation model at hand is comprehensive and detailed – a high demand of computational performance is considered as normal. You may close all other running applications on your computer in order to provide MATLAB the maximum of available CPU, RAM and disk capacities.</p>
<p>I opened the Simulink/ Simulink Simscape simulation model manually from its folder. Several blocks are marked in a red colour and I cannot compile/ run the simulation without errors.</p>	<p>Always run the GUI in advance in order to provide the required data and variables via the MATLAB “Workspace” (base) to the Simulink/ Simulink Simscape simulation model.</p>
<p>I executed the simulation and tried to run the GUI again using <code>runSim;</code> in the MATLAB “Command Window”. MATLAB reports ‘...’ is not found the current folder.</p>	<p>Check the MATLAB “Current Folder”, must be:  <a href="#">../BT_ABB_IRB_2600_Robot_Sim._v_A</a>  or:  <a href="#">../BT_ABB_IRB_2600_Robot_Sim._v_B</a></p>
<p>I executed the Simulink/ Simulink Simscape simulation. Result evaluation revealed that the set values did not reached steady states until the end of the simulation time.</p>	<p>Make sure that the value of the Simulink/ Simulink Simscape simulation time is equal to or higher than the minimum recommend simulation time printed to the MATLAB “Command Window” after pressing the “Open” button in the “main_ui”.</p>
<p>I tried to extend the Simulink Simscape simulation model block diagram. I was not able to connect a Simulink block to a Simulink Simscape block or signal (or vice versa) (e.g. “Scope” block).</p>	<p>Simulink and Simulink Simscape use different signal domains. Always use “PS-Simulink Converter“ or “Simulink-PS Converter“ blocks for interfacing. Also ensure to apply appropriate settings of the converter blocks (units, input handling).</p>
<p>I disconnected a block from the Simulink Simscape simulation model block diagram for testing purposes. The Simulink “Diagnostic viewer” now reports various errors when trying to compile/ run the simulation.</p>	<p>Simulink Simscape block diagrams do always need exactly one “Solver Configuration” block. All blocks of the block diagram need to be directly or indirectly (via other blocks) connected the “Solver Configuration” block.</p>
<p>The screen of my computer does not display anything. I cannot open the manual and check the troubleshooting section.</p>	<p>Try to reduce your personal environmental impact by avoiding printing useless manual documents next time.</p>