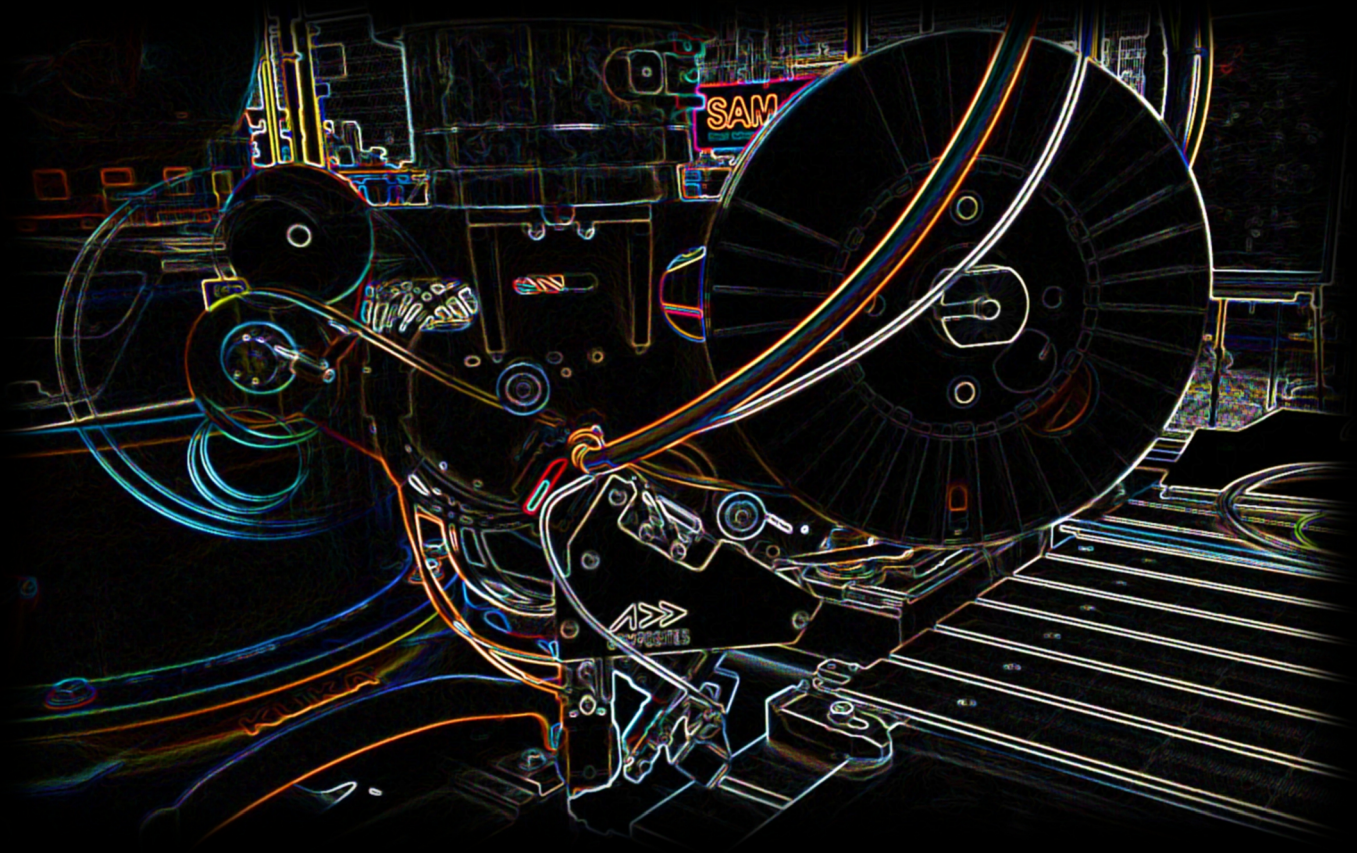


LayLa: An open-source offline programming framework for composite deposition

Implementation of a constant laydown speed deposition for laser assisted fiber placement of variable stiffness laminates

A. M. Mendes Florindo



LayLa: An open-source offline programming framework for composite deposition

Implementation of a constant laydown speed
deposition for laser assisted fiber placement of
variable stiffness laminates

by

A. M. Mendes Florindo

in partial fulfillment of the requirements for the degree of

Master of Science in Aerospace Engineering

at the Delft University of Technology, to be defended publicly on
Wednesday July 22, 2020 at 2:00 PM.

Student number:	4742125	
Thesis committee:	Dr. C. A. Dransfeld,	TU Delft, Chairman
	Dr. S. G. P. Castro,	TU Delft, Examiner
	Dr. D. M. J. Peeters,	TU Delft, Supervisor
	Ir. R. Tonnaer,	SAM XL, Co-supervisor

An electronic version of this thesis is available at <http://repository.tudelft.nl/>.

Acknowledgments

I would like to take this opportunity to express my deepest gratitude to everyone who directly or indirectly contributed to this thesis.

Daniël, thank you for your guidance and for keeping me on track through constant update meetings, forcing me to prioritize on the proposed objectives and preventing me from derailing with other inopportune tasks. Your practical wisdom on variable stiffness laminate and automated composite technologies brought realism into the simulations and elevated this work to another level. Also thank you for reviewing the thesis draft with many, many critical and insightful comments. Finally, I am grateful for your patience, it definitely took longer than we both expected, but you always made yourself available to assist me.

Rik, thank you for bringing up a topic that fruitfully sparked an old interest in robotics. I am grateful for all the time and effort you put into giving me introductory lessons on both ROS and KUKA manipulators. Furthermore, your presence at the update meetings helped to envision either problems or solutions that I was unaware of. At last, your initiative to get me working at SAM|XL during this thesis led me a huge learning experience and to feel unexpectedly part of an awesome team.

Special mention for the SAM|XL team. Kjelt and Claudia, thank you for giving such warm hospitality to our facilities. Dave, Raoul, Jonas, Julie and Yuri thank you for all the daily chit-chats, precious laughs and insights into dutch culture. Gijs, thank you for the invaluable discussions that helped steer this thesis. Berthil, thank you for attaching the end-effector to the manipulator during the experiments. I am excited about what the future holds for us in our upcoming endeavors and to share new memories with you all again.

Colleagues, friends, thank you for being there to make this journey easier and more enjoyable. These past three years were filled with plenty of memorable events that I will cherish for a long time. Thanks to all of you, I will take from Delft a great personal growth and self-improvement. Also, huge shout-out to our small but close portuguese community here in Delft.

Finally, to my parents and sister, thank you for our unconditional love and support.

*André
Delft, July 2020*

Abstract

The versatility of automated composite technologies granted the possibility of manufacturing laminates with a continuous variation of fiber orientation, also known as Variable Stiffness Laminates (VSL). Coupled with in-situ consolidation, such technologies have an important role in driving composite production for out-of-autoclave processes. Despite offering enough dexterity to be adapted for composite deposition, industrial manipulators are nontrivial to integrate into other applications than those originally envisioned (i.e, welding, material handling and coating). The complexity of programming a deposition task is exacerbated by the lack of control that these robots' proprietary software provide over the system capabilities and the exact trajectory executed. For several robot manufacturers, these complex curvilinear tow courses cannot be exactly specified in their software, hence an approximated solution is constructed by combining linear and circular motions. To aggravate the task, variations in the laydown speed are found when changing from linear to circular motion and vice versa, leading to changes in the heating temperature, possibly resulting in an uneven consolidation. Therefore, the inherent inaccuracy with manipulator systems demands performing a series of trials before accurately executing the requested tow course, making the programming of individual courses a cumbersome and time consuming task. Further improvements in layup control and performance are necessary to induce an accurate tracking of fiber tow courses and encourage new experimental validation of optimized VSL designs.

Contrary to most research in this field, the current thesis project focused on enhancing the deposition process quality rather than increase its productivity. The goals were to a) reduce variability of layup control with industrial manipulator systems, b) eliminate experimental iterative steps associated with programming tow courses and c) enforce a constant laydown speed that aids consolidation quality. To achieve such objectives, a framework entitled LayLa (Laying Laminates) was developed as an offline programming tool that automatically computes the series of continuous robot motions to perform tow courses at a desired laydown speed. LayLa exploits the potential of the Robot Operating System (ROS) and ROS libraries to communicate with the robot controller at a low-level and plan the deposition motion, while avoiding the use of native robot languages. After successfully generating the robot trajectory, the achieved deposition stage can be simulated in a graphical interface and later forwarded to the manipulator as an external command.

Experiments with LayLa were conducted for Automated Fiber Placement (AFP) using a six degrees-of-freedom KUKA manipulator, without actual deposition. From a VSL design algorithm, two fiber tow courses with a reduced turning radius were selected to be performed at a laydown speed of 0.1 m/s. The joint responsible for rotating the end-effector about its symmetry axis was identified as the primal source of issues, requiring higher accelerations to keep on track with the variable tangent vector while maintaining a constant laydown speed. Tow courses with constant or linear turning radius profiles were shown to be perfectly feasible from the robot's perspective, requesting almost constant accelerations at the previous joint. On the other hand, tow courses with a localized major shift in the turning radius, triggered by inflection points, required greater caution as the end-effector needed to completely change its rotating direction in a short period, only possible with an acceleration step that induced larger inaccuracies. Nevertheless, the overall results revealed reasonably accurate trajectory tracking of both joint and operating space variables, with errors at the end-effector position around 0.05 mm and at the laydown speed of 3.2 mm/s, corroborating the use of LayLa to create external commands for composite deposition.

This project was intended to serve as a starting point for the development of an independent and transparent composite deposition framework and much work remains to be done before it can be conceptualized as a competitive solution against industrial commercial software. Future efforts should focus on embedding LayLa with new functionalities, including creating an intuitive interface, incorporating fiber distribution methods, limiting joints' jerk and adding a seventh and eighth axis to the system, such as a linear track and a mandrel, commonly employed in Filament Winding (FW).

Keywords: variable stiffness laminates, in-situ consolidation, offline programming, industrial manipulators, robot operating system.

Contents

List of Figures	ix
List of Tables	xiii
List of Acronyms	xv
I Introduction and Motivation	1
1 Introduction	3
1.1 Background	3
1.1.1 The variable stiffness concept	3
1.1.2 On the road to automate composite deposition	4
1.2 Motivation	5
1.3 Report outline.	7
II Literature Review and Research Project	9
2 Literature Review	11
2.1 Design for fabrication of variable stiffness laminates	11
2.1.1 Manufacturing limitations	11
2.1.2 Steering with in-situ consolidation.	13
2.2 Industrial manipulator system	13
2.2.1 Robot manipulator.	13
2.2.2 Robot controller	14
2.2.3 Robot system software	15
2.3 Programming robot motions for deposition processes	16
2.3.1 Commercial software packages to program AFP technology	17
2.3.2 Open-source software packages to program robot motion	18
2.3.3 Enhance industrial manipulator performance	18
2.4 Motion planning for industrial manipulators	19
2.4.1 Path planning	20
2.4.2 Trajectory planning	21
3 Research Project	25
3.1 Research question and objectives	25
3.2 Hypotheses	26
3.3 Limitations	26
III Framework Developed	27
4 Experimental Setup	29
4.1 Industrial manipulator components	29
4.2 ROS components	31
4.3 Deposition specifications	34
5 The LayLa Framework	35
5.1 LayLa capabilities	35
5.2 LayLa workflow	36
5.2.1 Pose Builder	38
5.2.2 Path Planning	38
5.2.3 Time Parameterization.	38

IV	Data Analysis	39
6	Tow Course Fitting	41
6.1	Waypoint frame assignment for a space curve	41
6.2	Smoother curve approximation	44
6.3	The degree of the curve	46
6.4	Number of interpolated waypoints	48
7	Path Planning Benchmark	51
7.1	Motion problem construction.	51
7.2	Benchmark	52
8	Constant Laydown Speed	55
8.1	Available time parameterization in MoveIt	55
8.2	Impose a constant laydown speed	58
8.3	Soft approach to the tow course.	60
9	Accuracy of External Control	63
9.1	Experimental joint trajectory	63
9.2	Experimental end-effector trajectory	66
V	Conclusion and Recommendations	69
10	Conclusions	71
11	Recommendations	73
11.1	Extend LayLa capabilities	73
11.2	Manufacturing with LayLa	74
11.3	ROS libraries developments.	74
VI	Appendix	75
A	Appendix A: Industrial Manipulator Kinematics	77
A.1	Manipulator geometric model	77
A.2	Robotic kinematics models	79
A.2.1	Forward Kinematics	79
A.2.2	Inverse Kinematics.	80
A.2.3	Forward Differential Kinematics	83
A.2.4	Inverse Differential Kinematics	84
A.3	Kinematic singularities	85
A.4	Kinematic redundancy	85
B	Appendix B: Experimental Robot Motion	87
B.1	Tow Course 1	88
B.2	Tow Course 2	92
	Bibliography	97

List of Figures

1.1	Pictorial representation of conventional and variable stiffness laminates	4
1.2	Schematic representation of Laser Assisted Fiber Placement (LAFFP)	5
1.3	Example of an industrial ATL/AFP system	6
1.4	Example of a research ATL/AFP system	6
2.1	Schematics of fiber deposition strategies	12
2.2	Manufacturing technologies for VSL production	12
2.3	Types of robot arm and wrist adopted in the industry	13
2.4	Architecture of a standard 6 revolute degrees-of-freedom robot manipulator	14
2.5	Robot configuration vectors in the joint and operational space	14
2.6	Basic motions types provided by robot controllers	15
2.7	Complex cartesian paths obtained through multiple approximated linear motions or a spline motion	16
2.8	Initial path computed as the intersection between the surface and the projection of a major axis onto the surface	17
2.9	Illustration of an industrial AFP process obtained through VERICUT Composite Simulation . . .	17
2.10	Representation of RRT graph trees after 45 iterations and 2345 iterations	20
2.11	Schematic representation of the Descartes Dense algorithm	20
2.12	Schematic representation of a feedrate scheduling algorithm for a revolute manipulator	22
2.13	Cartesian trapezoidal and bell velocity profiles and their corresponding accelerations profiles .	22
2.14	Representation of TOPP limit curves in a velocity plot	23
2.15	Schematic representation of a TOPP algorithm	23
4.1	KUKA KR extra: a 6 revolute degrees-of-freedom robot manipulator	30
4.2	Robot cell adopted for the experiments in this thesis	30
4.3	Schematic representation of ROS mechanics for message passing	31
4.4	Replica of the experimental robot cell in Rviz	32
4.5	Schematic representation of ROS control	33
4.6	Example of a ROS message: <i>PoseArray</i>	34
5.1	Format of the <i>.txt</i> file containing the laminate description	36
5.2	LayLa simplified workflow diagram	37
6.1	Tow Course 1	41
6.2	Tow Course 2	41
6.3	Representation of a Frenet frame at a curve $C(s)$	42
6.4	Convention adopted for the TCP coordinate frame	42
6.5	Using the Frenet frames method to compute the orientation along the Tow Course 2	43
6.6	Using the Parallel Transport method to compute the orientation along the Tow Course 2	44
6.7	Unit tangent and normal x -axis components in function of the x coordinate if adopted the way-points provided in the laminate description file	44
6.8	Unit tangent and normal x -axis components in function of the x coordinate for curves with different smoothing condition	45
6.9	Close up of the region prior to the inflection point at the Tow Course 2, using different smoothing conditions	45
6.10	Discrepancy induced when approximating the Tow Course 2 by cubic B-splines with different smoothing conditions	45
6.11	Tow Course 2 interpolated using curves with different degrees	46
6.12	Discrepancy induced when approximating the Tow Course 2 using curves with different degrees	46

6.13 Unit tangent and normal x -axis components in function of the x coordinate for curves with different polynomial degrees	47
6.14 Acceleration requested at the joint A6 when approximating the Tow Course 2 by curves with different polynomial degrees	47
6.15 Discrepancy induced by approximating the Tow Course 2 with difference number of waypoints	49
6.16 Tow Course 1 prior and after fitted with a smoother quintic B-spline	49
6.17 Discrepancy induced by approximating the Tow Course 1 with a smoother quintic B-spline . . .	49
6.18 Tow Course 2 prior and after fitted with a smoother quintic B-spline	50
6.19 Discrepancy induced by approximating the Tow Course 2 with a smoother quintic B-spline . . .	50
7.1 Joint A1, A3 and A5 paths obtained with Descartes Dense for fitted Tow Course 2	54
7.2 Joint A2, A4 and A6 paths obtained with Descartes Dense for fitted Tow Course 2	54
8.1 Illustration of the joint position and time nomenclature used for IPTP method	56
8.2 Laydown speeds for fitted Tow Course 2 obtained using time parameterization methods available in MoveIt	57
8.3 Joint A6 trajectory for fitted Tow Course 2 using time parameterization methods available in MoveIt	57
8.4 Discrepancy induced at the fitted Tow Course 2 by time parameterization methods available in MoveIt	57
8.5 Joint A6 velocity for fitted Tow Course 2 using time parameterization methods available in MoveIt	57
8.6 Joint A6 acceleration for fitted Tow Course 2 using time parameterization methods available in MoveIt	57
8.7 Workflow of the time parameterization algorithm developed to keep a constant laydown speed	58
8.8 Laydown speed for fitted Tow Course 2 when enforced a constant end-effector speed of 0.1 m/s	59
8.9 Joint A6 velocity for fitted Tow Course 2 when enforced a constant end-effector speed of 0.1 m/s	59
8.10 Joint A6 acceleration for fitted Tow Course 2 when enforced a constant end-effector speed of 0.1 m/s	59
8.11 Illustration of the soft approach integrated at the start of a flat tow course	60
8.12 Representation of the soft approach combined with Tow Course 2, with an approach angle of 10°	61
8.13 Soft approach integrated into Tow Course 2 with linear and circular extension segment	61
8.14 Close up at the soft approach effects into the joint A6 position with a linear and a circular extension segment	62
8.15 Close up at the soft approach effects into the joint A6 acceleration with a linear and a circular extension segment	62
8.16 Laydown speed for fitted Tow Course 2 when enforced a constant speed of 0.1 m/s and incorporated a soft approach at both start and end of the tow course.	62
9.1 Joint A6 trajectory for fitted Tow Course 1, with requested waypoints distributed at a rate of 5 Hz	64
9.2 Joint A6 trajectory for fitted Tow Course 2, with requested waypoints distributed at a rate of 5 Hz	64
9.3 Joint A6 velocity for fitted Tow Course 1, with requested waypoints distributed at a rate of 5 Hz .	64
9.4 Joint A6 velocity for fitted Tow Course 2, with requested waypoints distributed at a rate of 5 Hz .	64
9.5 Joint A6 acceleration for fitted Tow Course 1, with requested waypoints distributed at a rate of 5 Hz	64
9.6 Joint A6 acceleration for fitted Tow Course 2, with requested waypoints distributed at a rate of 5 Hz	64
9.7 Joint A6 acceleration for fitted Tow Course 1, with requested waypoints distributed at a rate of 25 Hz	65
9.8 Joint A6 acceleration for fitted Tow Course 2, with requested waypoints distributed at a rate of 25 Hz	65
9.9 Illustration of the two-step correlation method to draw the commanded end-effector path closer to the executed outcome	67
9.10 End-effector path for fitted Tow Course 1	67
9.11 Discrepancy between the fitted Tow Course 1 and path performed by the robot	67
9.12 End-effector path for fitted Tow Course 2	67
9.13 Discrepancy between the fitted Tow Course 2 and path performed by the robot	67
9.14 Laydown speed for fitted Tow Course 1, with requested waypoints distributed at a rate of 5 Hz .	68

9.15 Laydown speed for the fitted Tow Course 2, with requested waypoints distributed at a rate of 5 Hz	68
A.1 Schematic representation of the Denavit-Hartenberg parameters proposed by Craig	78
A.2 KUKA KR 210 R2700 extra Denavit-Hartenberg frames	79
A.3 Visual representation of the distance between the joint A2 and wrist	81
A.4 Eight possible robot configuration for the same end-effector pose	83
A.5 The three types of singularities in a KUKA KR QUANTEC extra	85
B.1 Joint A1 trajectory for fitted Tow Course 1	88
B.2 Joint A2 trajectory for fitted Tow Course 1	88
B.3 Joint A1 velocity for fitted Tow Course 1	88
B.4 Joint A2 velocity for fitted Tow Course 1	88
B.5 Joint A1 acceleration for fitted Tow Course 1	88
B.6 Joint A2 acceleration for fitted Tow Course 1	88
B.7 Joint A3 trajectory for fitted Tow Course 1	89
B.8 Joint A4 trajectory for fitted Tow Course 1	89
B.9 Joint A3 velocity for fitted Tow Course 1	89
B.10 Joint A4 velocity for fitted Tow Course 1	89
B.11 Joint A3 acceleration for fitted Tow Course 1	89
B.12 Joint A4 acceleration for fitted Tow Course 1	89
B.13 Joint A5 trajectory for fitted Tow Course 1	90
B.14 Joint A6 trajectory for fitted Tow Course 1	90
B.15 Joint A5 velocity for fitted Tow Course 1	90
B.16 Joint A6 velocity for fitted Tow Course 1	90
B.17 Joint A5 acceleration for fitted Tow Course 1	90
B.18 Joint A6 acceleration for fitted Tow Course 1	90
B.19 X component of end-effector pose for fitted Tow Course 1	91
B.20 Y component of end-effector pose for fitted Tow Course 1	91
B.21 Z component of end-effector pose for fitted Tow Course 1	91
B.22 A component for fitted Tow Course 1	91
B.23 B component for fitted Tow Course 1	91
B.24 C component for fitted Tow Course 1	91
B.25 Joint A1 trajectory for fitted Tow Course 2	92
B.26 Joint A2 trajectory for fitted Tow Course 2	92
B.27 Joint A1 velocity for fitted Tow Course 2	92
B.28 Joint A2 velocity for fitted Tow Course 2	92
B.29 Joint A1 acceleration for fitted Tow Course 2	92
B.30 Joint A2 acceleration for fitted Tow Course 2	92
B.31 Joint A3 trajectory for fitted Tow Course 2	93
B.32 Joint A4 trajectory for fitted Tow Course 2	93
B.33 Joint A3 velocity for fitted Tow Course 2	93
B.34 Joint A4 velocity for fitted Tow Course 2	93
B.35 Joint A3 acceleration for fitted Tow Course 2	93
B.36 Joint A4 acceleration for fitted Tow Course 2	93
B.37 Joint A5 trajectory for fitted Tow Course 2	94
B.38 Joint A6 trajectory for fitted Tow Course 2	94
B.39 Joint A5 velocity for fitted Tow Course 2	94
B.40 Joint A6 velocity for fitted Tow Course 2	94
B.41 Joint A5 acceleration for fitted Tow Course 2	94
B.42 Joint A6 acceleration for fitted Tow Course 2	94
B.43 X component of end-effector pose for fitted Tow Course 2	95
B.44 Y component of end-effector pose for fitted Tow Course 2	95
B.45 Z component of end-effector pose for fitted Tow Course 2	95
B.46 A component for fitted Tow Course 2	95
B.47 B component for fitted Tow Course 2	95
B.48 C component for fitted Tow Course 2	95

List of Tables

4.1	KUKA KR 210 R2700 extra specified limits	30
6.1	Elements of discretization for the fitted Tow Course 2, assuming a constant end-effector speed of 0.1 m/s	48
7.1	Runtime and Memory consumed while planning the fitted Tow Course 2 with different algorithms	53
7.2	End-effector position discrepancies obtained while planning the fitted Tow Course 2 with different algorithms	53
A.1	Denavit-Hartenberg parameters for a KUKA KR 210 R2700 extra	78

List of Acronyms

3D	-	Three-Dimensional
AFP	-	Automated Fiber Placement
ATL	-	Automated Tape Laying
CAD	-	Computer-Aided Design
CNC	-	Computer Numerical Control
CPU	-	Central Process Unit
CTS	-	Continuous Tow Shearing
DASML	-	Delft Aerospace Structures and Materials Laboratory
FIR	-	Finite Impulse Response
FRP	-	Fiber Reinforced Polymers
FW	-	Filament Winding
I/O	-	Input/Output
IPTP	-	Iterative Parabolic Time Parameterization
ISP	-	Iterative Spline Parameterization
KRL	-	KUKA Robot Language
KSS	-	KUKA System Software
LAFP	-	Laser Assisted Fiber Placement
NURBS	-	Non-Uniform Rational B-Spline
OEM	-	Original Equipment Manufacturer
OMPL	-	Open Motion Planning Library
PI	-	Proportional and Integral
PID	-	Proportional, Integral and Differential
PLC	-	Programmable Logic Controller
RAM	-	Random-Access Memory
ROS	-	Robot Operating System
ROS-I	-	ROS-Industrial
RRT	-	Rapidly-exploring Random Tree
RSI	-	Robot Sensor Interface
SRDF	-	Semantic Robot Description Format
TCP	-	Tool Center Point
TFP	-	Tailored Fiber Placement
TOPP	-	Time-Optimal Path Parameterization
TRL	-	Technological Readiness Level
URDF	-	Unified Robotic Description Format
VSL	-	Variable Stiffness Laminates

I

Introduction and Motivation

Introduction

1.1. Background

With the rising numbers of airline passengers and airfreight shipments, aircraft manufacturers have been witnessing a recurring demand for a higher production volume and a diversified product lineup. Naturally, achieving these grand ambitions dictates an investment in resources which cannot come at the expense of undermining the structural integrity of their products. Safety is, or should be, the core driver for all aviation stakeholders. To not jeopardize this commitment, aircraft manufacturers justifiably became reluctant to adapt early-on concepts that may introduce significant long-term improvements in the overall aircraft's performance but do lack the necessary certification to implement them. The application for such certifications demands performing a series of meticulous analysis, sponsored by aircraft manufacturers and possibly other parties concerned, to demonstrate that the alternative approach complies with the airworthiness requirements set by the state authorities. These inquiries ultimately shape the production of new airplane design as financially expensive and slow, having to wait for years, if not decades, before new concepts are flying.

Even though advancements are introduced at a modest rate, retailers keep pressuring manufacturers to modernize their aircraft with increased performance. The endless ambition to lower operating cost, coupled with the soaring crude oil price and the increased environmental awareness over the public's perception (and later over the government's, imposing tighter environmental regulations), constitute the main premises that prompted an incessant search for fuel-efficient airplanes. Although not the single reason, it is not possible to address the efficient use of fuel without scrutinizing the implications of the aircraft's weight to its consumption. During both climbing and descend, the aircraft's weight directly influences the amount of thrust requested to ascend or descent to a specific height. In addition, to avoid stalling while cruising, heavy airplanes have to generate more lift by operating at higher speeds than lighter models. Indeed, minimizing weight became one of the core principles in airframe design. Considerable reductions in the structure weight are only feasible by challenging traditional practices and tackle different materials, optimize structural designs and reformulate manufacturing processes.

1.1.1. The variable stiffness concept

On the subject of materials, composites are one of the most compelling inclusions to the aerospace portfolio. In an oversimplified description, composite materials are solutions that aggregate two or more distinctive materials which, when combined, result in a material with higher properties than those of its constituents. Out of those, Fiber Reinforced Polymers (FRP) have been increasingly gaining the preference of aircraft manufacturers over its metal competitors due to their unusual combination of lightweight, high stiffness and durability. Not only do they allow the construction of lighter structures with superior physical properties, they also offer better impact, fatigue and chemical resistance, three dominant causes of aircraft structural failures. The integration of composites in the aerospace industry has already proved fruitful with both secondary and primary structures, examples being the contemporary long-range airplanes Boeing 787 and Airbus A350XWB containing more than half of their weight in composites.

Designing airplanes involves constantly making high-stakes decisions and with the desire for FRP structures comes an ever-growing range of design possibilities. These include simple choices such as the selection of its constituents, for instance glass or carbon fibers embedded in thermoset or thermoplastic matrices, in

addition to more elaborate decisions, especially related to the optimization of the layup sequence. In regards to the layup sequence, at the moment manufacturers have been only designing composite structures using a constant fiber orientation per layer, provided by unidirectional prepreg (pre-impregnated) or woven fabrics. The adoption of automatic deposition technologies has brought out the ability to tailor composite properties to the expected applied loads, producing the so-called Variable Stiffness Laminates (VSL). Their variable mechanical properties are attained either by locally changing the number of layers (creating ply drops), by continuously modifying the fiber orientation in each layer, or by combining both methods. This thesis will focus on the middle method, also denominated as fiber steering, and a composite designed by such method is displayed in Figure 1.1.

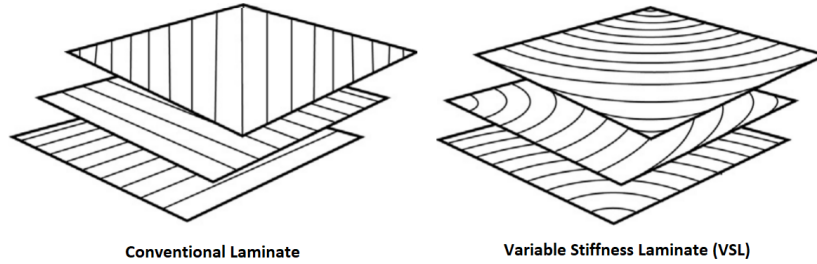


Figure 1.1: Pictorial representation of conventional and variable stiffness laminates. Cited in [1], p.2.

1.1.2. On the road to automate composite deposition

Composite raw materials are notoriously vulnerable to defect. To not compromise product quality, traditional composite deposition relies heavily on manual labor. With the staggering growth of composite materials in the aerospace and automotive industries, markets where cost-based decision making is more prevalent than in earlier composite adoption, manual production of conventional composites has failed to arouse excitement and has slowly been discarded. Numerous doubts about the financial reliability of manual production help to justify such attitude. For instance, specialized workers are required to perform these labor-intensive tasks and it is the manufacturer's responsibility to arrange training courses to teach the technique. In addition, large-scale structures, such as the lower wing shells of an A350XWB, contain various sections, each with a specific stacking sequence, implying the deposition of dozens if not hundreds of layers. Consequently, traditional manual manufacturing processes cannot simultaneously satisfy the high production rate and low cost requisites necessary for a profitable manufacturing industry. In order to overcome this workforce deficit and economic burden, manufacturers have been embracing automatic deposition technologies.

Several automatic depositions concepts have been exploited for the industrial production of conventional composite structures. Still, few solutions have received the rigorous aerospace seal of approval that met the respective high-quality standards. Automated Tape Laying (ATL) is one of the latter technologies, in which a fiber tape roll is unwound and fed into a robotic system, to be sequentially laid and compressed onto a mold by a soft silicone roller. Another similar variant widely employed is Automated Fiber Placement (AFP), steering narrower tow(s) over sharply curved surfaces. In both cases, applying a slight compression introduces enough pressure to remove most of the air trapped and insure a good adhesion bond between layers. Additionally, for thermoset materials, a better adhesion at high laydown speed is achieved by slightly pre-heating a small area of the substrate forward to the compression roll. Hence a heat source is frequently, but not compulsory, attached to the deposition tool, such as hot air or infrared radiation.

Regardless of which deposition process is selected, in the case of producing thermoset composites, a couple of secondary processing steps must follow. These steps include vacuum bagging and curing, with the latter being extremely expensive due to, in most cases, requiring a combination of high temperature and pressure only achievable in an autoclave. Alternatively, in the case of producing thermoplastic composites, an additional consolidation step may be avoided by performing in-situ consolidation. In a brief summary, in-situ consolidation consists of heating the thermoplastic tow and subtract at high temperature for a remarkably short time, so that tows are laid down in a molten state and consolidated right after being compressed. To this end, a powerful laser heater is added to an automatic system much identical to those utilized in AFP, as schematically represented in Figure 1.2. This recent deposition technology is often designated as Laser Assisted Fiber Placement (LAFFP) and promises to reduce both production time and cost, two crucial variables for the sustainable production of large-scale structures, while respecting the quality standards.

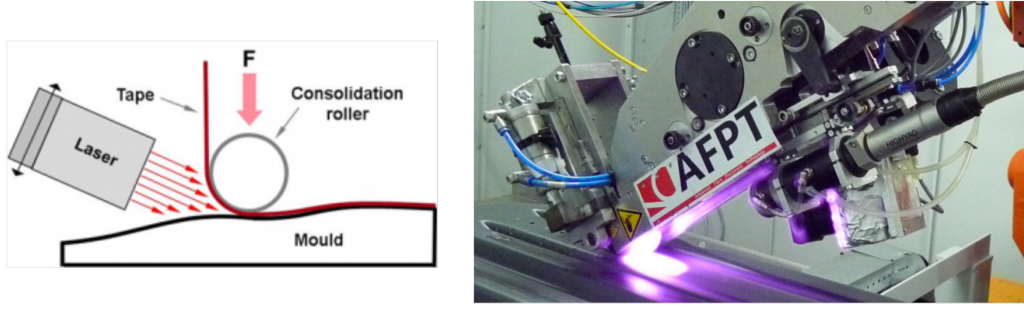


Figure 1.2: Schematic representation of Laser Assisted Fiber Placement (LAFP). Copyrights AFPT.

All manufacturing processes are susceptible to a variety of internal and external uncertainties that undermine the productivity of an actual system in a competitive and operational environment. Automated deposition technologies are no exception and examples of possible defects include puckers and bridged tows, originated by factors such as incorrect feeding rates and bad adhesion to the mold, respectively. For this reason, a thorough understanding of the technology and all its variables is crucial to assure the quality standard of the end product. Since composite industry has been extensively familiarized with ATL and AFP for mass-production of conventional composites, the state of maturity of these technologies has attracted researchers to develop VSL fiber angle distribution models based on the limitation provided by such processes.

The prospect of manufacturing VSL with LAFP could conduct the aerospace industry to a pivotal and strongly needed transformation, leaving behind the traditional portrait of a slow and expensive market for a highly fast, light and recyclable business. Still, plenty of the ground knowledge to be covered regarding either VSL and LAFP is yet to be discovered, hampering for the moment the demonstration of these design and technology's capabilities. Understandably, the research on fiber steering has predominantly been focused on the complex task of optimizing the stiffness distribution with or without taking into consideration manufacturing limitations of fiber placement technologies. Only when more research makes the transition from design to the production of these laminates, the composite community will realize several hidden hindrances with current industrial robot systems and practices of programming the robot motion. Thereby, independent studies that could streamline this transition and investigate the robot behavior for composite deposition would be much desirable. Such gap will be dealt in the present thesis.

1.2. Motivation

Aircraft manufacturers had, have and will certainly have the leading role in the development of automated composite deposition systems [2]. Despite detaining all the expertise in materials science and structural design to conceive composites structures, the fundamentals of how to construct a robotic system from the ground up exceed the core skills of aircraft companies. By partnering with engineering firms specialized in innovating industrial equipment, the missing link between composites and automation materialized and first AFP systems for a commercial environment emerged. Inspired by Computer Numeric Control (CNC) machines of the time, initial industrial AFP systems consisted of gantry systems of large proportions, where three motors provided the x , y and z translational motion and three other motors provided the rotational motion for the orientation of the tool. Examples of such systems are still commonly employed, with TORRES-FIBERLAYUP made by MTorres, as shown in Figure 1.3, used in the production of the A380 wing shells [3] and Cincinnati VIPER made by Fives Cincinnati applied in the construction of the forward fuselage section of the Boeing 787 [4].

The main strengths of AFP gantry robots rely on the fact that these machines were specifically developed to perform a unique task, the deposition of FRP tows. In this way, both the robot's hardware and software were shaped to maximize performance requested by the aircraft manufacturer. Additionally, the simplified robot structure solely utilizes translation to define the tool's position, giving the operator a clear perception of where the tool is located when analyzing robot data. In spite of their large dimensions, gantry systems are highly space-efficient in the sense that the robot can reach almost the whole area occupied by the robot cell. All in all, gantry systems are an ideal solution for the production of large composite structures and have typically attached a deposition tool containing numerous tows to increase productivity.

Industries other than aircraft are starting to conceptualize production lines containing automated composite layup. Furthermore, in order to become independent from external composite manufacturers, aca-



Figure 1.3: Example of an industrial ATL/AFP system: at the left a high-rail TORRESFIBERLAYUP of MTorres used by Airbus and at the right its large deposition tool for high productivity. Copyrights MTorres.

demetic institutes are also inclined to purchase their own AFP system to carry out internal projects. With the widespread adoption of FRP materials, new stakeholders in the composites market have emerged, which Lukaszewicz et al. [2] state will embrace a crucial and necessary role in the future of automated composite technologies. Yet, in the interest of fabricating composites in research facilities, such gantry systems are excessively large, expensive and confined to a single task, consequently representing a heavy investment with sparse return. Hence, another alternative system had to be conceived for these cases. Robot manipulators caught researchers' attention due to its precision, robustness and dexterity to adapt to a diversity of tasks at a much smaller infrastructure cost and size. Typically containing between five or six rotational degrees-of-freedom, industrial manipulators can also be linked with a linear track or a mandrel to expand their mobility and workspace. An example of AFP systems with a robot manipulator connected to an in-house developed deposition tool is represented in Figure 1.4.

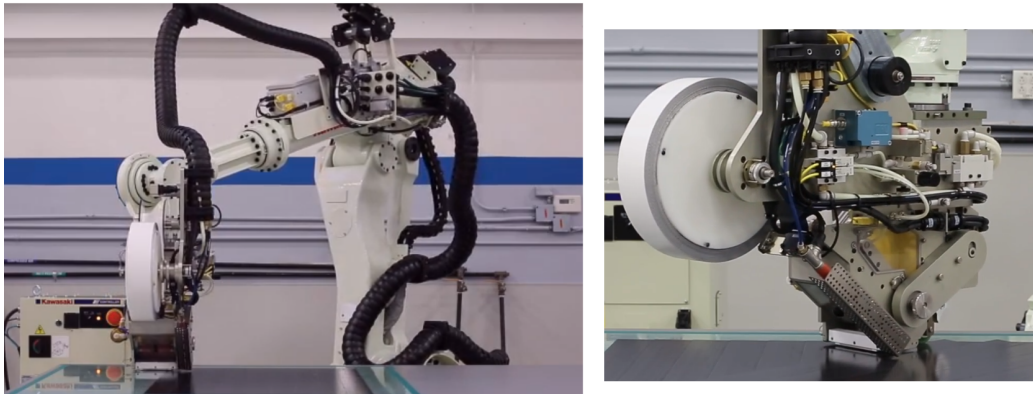


Figure 1.4: Example of a research ATL/AFP system: at the left a Kawasaki Z Robot manipulator used by Automated Dynamics and at the right its small tape dispenser. Copyrights Automated Dynamics.

Notwithstanding the flexibility to accomplish a wide variety of automation applications, for the moment, the adoption of industrial manipulators has remained utterly stagnated to a same limited set of tasks: welding, material handling and dispensing [5]. Given that manipulators are sold as a standard model, each Original Equipment Manufacturer (OEM) of these units develops its own native programming language, largely based on that set of tasks and deliberately makes it as easy as possible to use. Industrial manipulators are then governed by simplified software architectures incapable of supporting higher-level programming, which ultimately barriers researchers from adding advanced functionalities that could enhance the robot motion [6]. For this reason, a massive effort is needed to adapt their software to tasks outside the intended scope, which narrows down OEMs as the only realistic source of new capabilities. Considering that composite automation has been a rather niche market, it has not yet been commercially attractive from the OEMs point of view to invest in the development of a deposition program. Without providing a complete control over the systems' capability and the exact trajectory performed, adapting industrial manipulators for composite applications

may lead to an imprecise steering, whose final product is then propitious to misalignments or other defects such as gaps and overlaps [2, 7].

Unacquainted with possible alternatives or doubting the viability of those, composite researchers have been utilizing native robot languages to program deposition processes [8–10]. Undertaking the production of conventional composites using automated systems, the robot only has to execute straight paths or, depending on the mold's shape, paths described with a large turning radius. In practice, this implies that each tow course can be merely defined by a linear motion between two waypoints, the intended course's start and end points, or a circular motion between three waypoints, respectively. Unfortunately, the same pragmatics cannot be employed for laminates created with VSL design strategy without limiting their tow courses to solely lines or circles. As the continuous modifications in the fiber angle are achieved through the use of piecewise polynomial curves [11–13], which may give rise to extremely complicated shapes, these courses cannot be accurately defined by the usual robot motion types but rather by a combination of both linear and circular motions [9]. To obtain the requested VSL path, a naive initial motion is programmed, followed by checking its accuracy and later manually adjusted to approximate the outcome with the request, with the last two being repeated until the optimal motion is reached. Bearing in mind that a 600 mm wide composite sample with 10 layers can easily contain as many as 100 fibers tows of $\frac{1}{4}$ inches per layer, which can collectively lead to more than 1000 courses per composite structure, programming each course individually is extremely cumbersome and a different approach must be conceived.

With the implementation of in-situ consolidation to AFP, slight modifications in the state-of-the-art deposition process are to be expected, as steering will be influenced by different processing parameters [8]. To achieve a consistent part quality with LAFP, the thermoplastic laminate should be uniformly consolidated or, in other words, the temperature at the bonding interface between the substrate and incoming tow should be kept constant throughout the deposition stage. With the robot in continuous motion, the amount of thermal energy emitted by the deposition tool is a result of variables which include the source's output power and the heating time, the latter being proportional to the laydown speed. For simple motion types, modifying the maximum translation and rotation of the tool is usually sufficient to obtain a constant speed at the tool. However, when a series of motions are combined, the robot will automatically reduce the tool speed whenever it cannot perform the motion at the laydown speed specified. Therefore, in current industrial manipulator systems, variations in the laydown speed along each tow path are highly contemplated when facing curvature changes, that is, when changing from linear to circular motion and vice versa. If the heat source emits a constant output power, then different heating time are likely expected, potentially leading to local under- or over-heating, ultimately degrading the end-product properties. A possible solution could be the development of a control system that actively modifies the source's power according to the current laydown speed. Since reducing one variability present in in-situ consolidation could actually be more helpful for future research, in this thesis a more simplified idea was pursued, to enforce a constant laydown speed along each tow path.

As evident by now, several hindrances are currently undermining researchers from producing VSL. Industrial manipulators and OEM software are intrinsically the source of a portion of them, with the nondisclosure of the robot software architecture hiding some component from its operators, the arduous task for programming a complete laminate and the non-constant speed for in-situ consolidation some noticeable problems. The present project intends to investigate the motion of industrial manipulators for automated composites technologies and provide a software solution that deals with all programming steps associated with planning the robot motion planning to perform tow courses at a constant laydown speed, prior to the actual command to move the robot.

1.3. Report outline

This thesis report is organized as follows: Chapter 2 comprises relevant literature on VSL, their fabrication issues and also a small background into industrial manipulator hardware, software and motion planning strategies. Afterwards, Chapter 3 exposes the research objective of this thesis, detailed with the research question and sub-questions. In Chapter 4, the various components of the experimental setup are presented, followed by an insight into the developed framework in Chapter 5. Chapters 6, 7 and 8 describe the steps taken to transform a set of cartesian waypoints into a robot trajectory and are respectively divided into tow course fitting, planning of the robot joints' position and time parameterization for a constant laydown speed. Subsequently, in Chapter 9 those obtained robot trajectories are executed in real robot and their accuracy is analyzed. Lastly, conclusions and recommendations are made in Chapter 10 and 11, respectively.

II

Literature Review and Research Project

2

Literature Review

This chapter begins by presenting a historical review of the VSL concept, addressing its mechanical advantages and associated hindrances which are currently compromising its manufacturability. Since robotics is a relatively unexplored field for one with an aerospace background, the remaining of this chapter is focused on providing a brief take on industrial manipulators and software tools to program a composite deposition task. Henceforth, industrial manipulator and its constituents are first described and explained their working procedures. Afterwards, available software choices and improvements of robot performance for composite deposition technologies are discussed. At last, an insight into algorithms to plan the robot motion from a list of waypoints and time parametrization strategies is conducted. To avoid overcrowding this chapter with too much information, the underlying theory behind industrial manipulator kinematics can be found in Appendix A.

2.1. Design for fabrication of variable stiffness laminates

The thought of transfiguring the stiffness properties of composites through the use of curvilinear fiber paths can be traced back to the 1970s [14], yet a greater momentum was only sparked in early 1990s. In 1991, Hyer and coworkers [15, 16] employed finite elements methods to split a lamina into multiple regions, each region with its own optimized fiber orientation, creating a design model nowadays known as patch model. A couple years later, Gurdal and Olmedo [11] introduced the term Variable Stiffness Laminates (VSL) to represent composites modeled with a continuous curved fiber format, clarified the reason behind their peculiar mechanical behavior and predicted an unprecedented breakthrough in their structural efficiency over traditional laminates. In order to simulate their mechanical behavior, Gurdal and Olmedo developed a numerical model of a square panel VSL subjected to uni-axial forces. The results revealed the presence of shear stresses even when the composite had been modeled without shear-extension coupling. Their explanation for this phenomenon was that the different elastic properties along the square VSL originated transverse stresses in unexpected regions. Subsequent works corroborated the expected structural efficiency of tailoring composite stiffness properties to the specific loading condition, with significant weight reductions [17–19], attenuation of stress concentrations [20, 21] and improvements of buckling performance [22–24] extensively accomplished.

2.1.1. Manufacturing limitations

Initial experimental validations of previous analytical design models were undermined by a series of fabrication issues, including gaps, overlaps and excessive tow curvature, which jeopardized the highly anticipated superior structural performance of VSL over conventional laminates [25]. To overcome these obstacles, the limitations of state-of-the-art deposition processes must be taken into account at an early stage of the design project. Ongoing research has then been focused on optimizing fiber angle distribution while taking into consideration manufacturing constraints, with exhaustive reviews of those presented in [13, 26]. Yet, few works have provided structural analysis of real structures, and those who have presented, mainly use simple geometries and loading conditions, obfuscating how these limitations would be handled for complex VSL structures in a real-world scenario [13].

The vast majority of studies dedicated into fabricating VSL have been combining AFP process with thermoset materials, either using dry fabric [18] or prepreg tows [27, 28]. Steering fibers with such technology is

based on the premises that tows can be continuously bent [29], that is, while the inside edge is being compressed, the outside edge endures tensile loads. Specially for prepregs, where fibers are being strongly restrained from movement by the matrix, as the turning radius is reduced, the tow edges cannot tolerate such extreme forces and defects such as thickness reduction and local buckling are observed in the form of tow wrinkling [29]. As such, manufacturing VSL with AFP, the laminates are restricted to a large minimum turning radius, which is constrained by material properties, compaction pressure, laydown speed and, more notable, the tow width [27]. For tows with $\frac{1}{4}$ inches width, the minimum turning radius revolves around 0.80 m [8], with larger tow widths confining substantially larger turning radii [27].

Naturally, the strategies utilized to distribute fiber angles have also a strong influence on the occurrence of several manufacturing problems. Parallel and shifted fiber angle distribution strategies, introduced in [30] and represented in Figure 2.1, have been widely adopted for VSL design. In spite of producing VSL without gaps or overlaps, parallel methods are quickly restricted by the minimum turning radius of the material. In shifted methods, although the turning radius no longer becomes a major issue, gaps and overlaps are unavoidable [25]. On one hand, gaps decrease the composite properties, creating resin pockets which act as stress concentrations. On the other hand, overlaps behave as stiffeners, yet with the increase of the weight and thickness of the composite. With AFP, tow drops can prevent gaps and overlap regions to some extent, but not eliminate them completely. A decision has to be made between what percentage of gaps and overlaps are allowed in the final product.

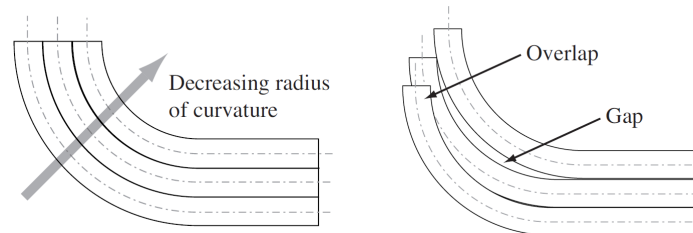


Figure 2.1: Schematics of fiber deposition strategies, on the left parallel method and on the right shifted method. The parallel method consists of increasing or decreasing the turning radius so that neighboring fiber tows share one colinear edge. In the shifted method, only a reference fiber path is computed and later repeated by shifting its location. Cited in [21], p. 11.

Towards the manufacturability of VSL, some researches have circumvented AFP limitations through proposing innovative composite manufacturing methods. Tailored Fiber Placement (TFP) [17, 20, 23] and Continuous Tow Shearing (CTS) [29, 31, 32] are the two most prevalent examples. The former is based on widely known embroidery techniques, which can easily be adapted for composites production at low cost, yet challenged by unavoidable defects in the final product, which include fiber waviness and fiber damage caused by the needle threads [17]. The latter exploits the concept of shear deformation, eliminating the coupling between the turning radius of curvature and the material width and resulting in perfectly shifted fibers without gaps or overlaps, yet uneven thickness distribution is to be expected as fiber angle and ply thickness are still coupled [31]. These tow steering manufacturing technologies are still considered to be at a lower Technological Readiness Level (TRL) and require further studies.



Figure 2.2: Manufacturing technologies for VSL production, from the left to the right: TFP [23], AFP [27] and CTS [31].

2.1.2. Steering with in-situ consolidation

Recently, thermoplastic VSL deposition with in-situ consolidation was successfully validated for LAFP processes [8]. In the same article, Clancy et al. investigated the effects that laydown speed and turning radius have on the bonding interface when laying down $\frac{1}{4}$ inches thermoplastic tows using a 3 kW diode-laser as the heat source. While for low laydown speeds of 1.5 m/min (0.025 m/s) the interface was prone to expand, even after compressed by the roller, for high laydown speeds of 10 m/min (0.16(6) m/s) poor bonding between the steered tow and substrate occurs, leading to fiber pull-up. In regards to the turning radius, a simultaneous largest decrease in width and increase in thickness was detected for a small turning radius of 0.20 m, with smallest modifications for a large turning radius of 0.80 m. The ability to fabricate large and complex VSL with LAFP technology was later authenticated by the successful fabricating and testing of a B737/A320 wingbox segment [9] and in another a 39 foot, high-aspect-ratio wingbox [33].

2.2. Industrial manipulator system

An industrial manipulator is in essence a metal carcass incorporated with various actuators, sensors and brakes. In order to operate and monitor its motion, the robot's "brain" is located in an external cabinet known as robot controller. An industrial robotic system then consists of several components, which include both manipulator and robot controller, but also a teach pendant, connecting cables, software and tool accessories. The goal of this section is to provide a basic level of understanding of how automated systems are operated from a programming perspective, with a focus on the robot manipulator, robot controller and integrated software.

2.2.1. Robot manipulator

A robot manipulator is an open serial chain of rigid links interconnected by a series of joints, with the first link anchored on a surface (ground, ceiling, pedestal or even tracks) and an end-effector attached to the last link. End-effectors, or simply tools, are devices that contain all equipment necessary to perform the assigned task. Typically, each robot's joint is made of an actuator that adds one degree-of-freedom to the structure. While prismatic joints enable translation motion in one axis, revolute joints provide rotational motion between the two connected links. As will be seen later, six joints are required to completely specify an end-effector pose in a three-dimensional (3D) space.

Using a simplified model, a manipulator can be divided into two components [34]. The three bottom joints and their respective rigid links integrate the designated robot's arm, which is the main driver for modifying the end-effector position. Changes in the tool orientation are performed by the second element, named wrist, which contains the three remaining joints. Depending on the selected coordinate systems for the end-effector, a variety of arms and wrists designs have been adopted by the industry, with some examples represented in Figure 2.3. For this study case, the manipulator will be assumed having a fixed base, containing six revolute degrees-of-freedom and a spherical wrist, as shown in Figure 2.4.

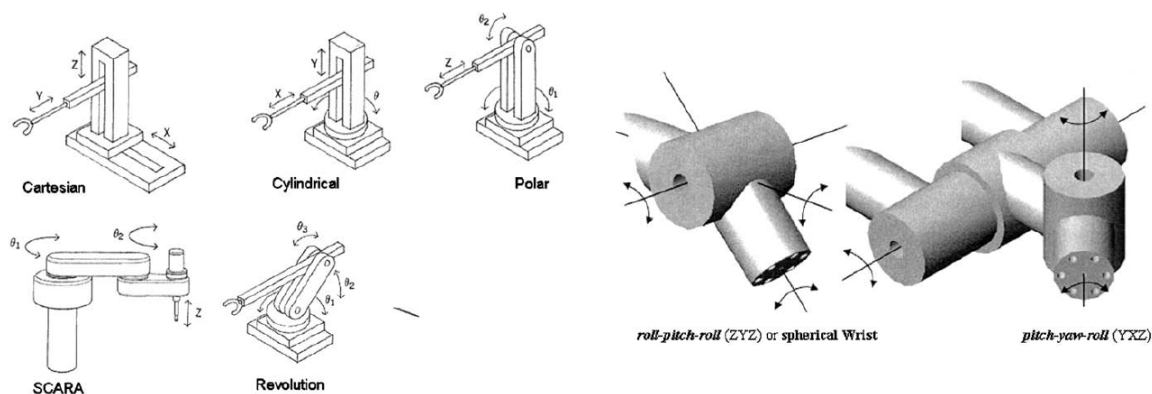


Figure 2.3: Types of robot arm and wrist adopted in the industry, from the left to the right, respectively. Cited in [34], p. 37.

A robot configuration is a set of scalar coordinates that specify the position and orientation of the robot's rigid links relative to a world fixed frame (usually located at the manipulator's base). Since each joint adds a single degree-of-freedom to the system, the robot configuration can be completely described by a scalar

vector \mathbf{q} with coordinates equal to the joints position. When the robot configuration is described using the so-called joint (or generalized) coordinates q_i , the robot is said to be operating in the joint space. For the robot manipulator represented in Figure 2.4, coordinates q_i are equivalent to the angle of rotation θ_i about the axis O_i .

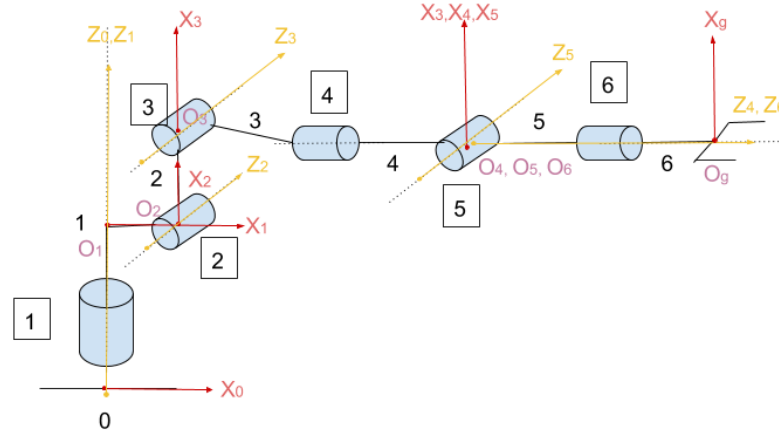


Figure 2.4: Architecture of a standard 6 revolute degrees-of-freedom robot manipulator. The Z-axis represent the axis of rotation about the respective joint.

Another approach to describe a robot configuration is performed by using the end-effector position and orientation relative to the same world frame (known simply as the end-effector pose). The manipulator can then be specified in the denominated operational space and is characterized by an operation space vector \mathbf{p} . The number of coordinates p_i depends on the coordinate system adopted to define the position and orientation of the robot tool. In many cases, the end-effector pose is expressed by the so-called minimal representation, in which its position is defined using 3D cartesian coordinates (x, y, z) , while the three Euler angles are used to describe its orientation (α, β, γ) . For the latter, quaternions are also a valid choice, yet these add an additional fourth coordinate. It should be pointed out that, in contrast to joint space, an operational vector does not explicitly represent a single robot configuration. The reason for this is how an end-effector pose is mathematically related to a joints position, typically performed through kinematics models, as described in Figure 2.5. For a 6-axis industrial manipulator, while forward kinematics returns a unique end-effector pose from a joint vector, inverse kinematics can obtain a total of eight possible robot configurations from a single operator vector.

$$\mathbf{q} = [\theta_1 \quad \theta_2 \quad \theta_3 \quad \theta_4 \quad \theta_5 \quad \theta_6]^T$$

Inverse
Kinematics

↶

Forward
Kinematics

↷

$$\mathbf{p} = \underbrace{\begin{bmatrix} x & y & z \end{bmatrix}}_{\text{position}} \underbrace{\begin{bmatrix} \alpha & \beta & \gamma \end{bmatrix}}_{\text{orientation}}^T$$

Figure 2.5: Robot configuration vectors in the joint and operational space. Forward kinematics computes the tool pose from a given joints position. On the other hand, inverse kinematics finds the joints position required to perform a given end-effector pose. The underlying theory behind these kinematics models is presented in Appendix A.

2.2.2. Robot controller

A robot controller is a device that combines hardware and software components to provide means of communication between operator and manipulator, to allow sensor reading and to grant safety features for programming and controlling the robot motions. Several components are present in one of these devices, from a computer (responsible for path planning, program management and sequence control) to drivers' power supplier and logic control unit and even communication and safety interfaces [35].

In short, a robot controller receives a command from the operator to move the machine in a concrete way, calculates a discrete robot motion that performs the prior command, conveys the requested motion into a program and finally submits the power to execute the desired program in a controlled manner. Depending on the robot OEM, the motion program is broadcast either through a download approach, where the entire trajectory is directly uploaded into the manipulator, or through a streamline approach, in which data is streamed at every few fixed microseconds. The latter approach makes real-time control feasible, even achieving extremely smooth motions if data is transmitted with a high bandwidth and a low latency.

The communication between the operator and the robot controller is primary conducted with a teach pendant, a device that displays an user-friendly interface for operating and programming industrial robots. Some robot OEMs also offer proprietary software packages to directly communicate between the robot and an external computer without the need for a teach pendant. Other licensed packages can be purchased and added to the robot controller, allowing, for instance, external control through sensor communication to even support the integration with a Programmable Logic Controller (PLC). PLCs are specially useful for automated cells with multiple machines, enabling input/output (I/O) processing and control functionality independently of the brand of the system.

Safety features embedded in robot controllers represent a crucial functionality for industrial applications, where the equipment has to follow a rigid set of regulations. If an emergency stop is pressed or sudden errors are detected, for example, when a task outside the robot's reachable area is requested or a soft performance limit is surpassed, the robot must come to a halt and execution must be manually resumed. Other functionalities include adding the so-called stopping distances, to avoid collision with the workspace boundaries. The procedure on how to stop the robot is also regulated, from a controlled stop with power available on the machine actuators to an uncontrolled stop with immediate removal of the power.

2.2.3. Robot system software

To correctly recreate a series of robot configuration, all joints have to be working in a synchronized fashion, with each joint motor needed be told where and when to move. For this purpose, a system software responsible for operating and controlling the industrial robot comes pre-installed with the robot controller. These proprietary software products are developed by the robot manufacturers and their intended use is solely restricted to OEMs system products. They provide a native programming language coupled with debugging tools and a library of functions to help operators to easily explore the machine's capabilities, reducing the efforts to program robot motions and thus incentivizing the implementation of these robot systems in manufacturing cells. A language interpreter comes within the robot controller, allowing users to write code, download it to the controller, and run it immediately without the need for any type of file conversion.

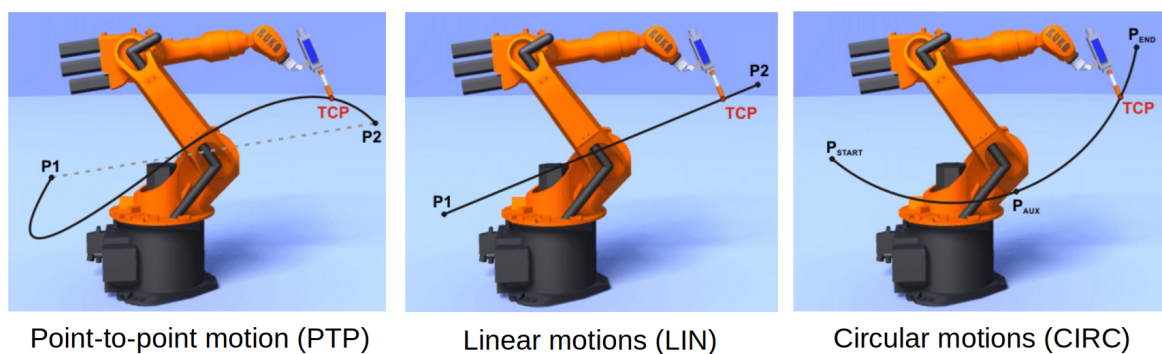


Figure 2.6: Basic motions types provided by robot controllers, with TCP meaning Tool Center Point. Cited in [36], pp.159,160

Through these software solutions, an industrial task is traditionally programmed by manually joggling the robot to the desired poses, followed by recording the obtained robot configuration and lastly specifying how the robot should move between those configurations. For the latter step, most robot OEMs incorporate in their equipment's native languages three basic motions types to automatically generate robot trajectories. Following Figure 2.6, on the left, point-to-point motions deliver the fastest trajectories from one to another waypoint. For revolute joints, curved paths are typically quicker (although not the shortest) to execute than straight lines. In the middle, linear motions execute straight trajectories from the operator's perspective between two waypoints. Despite point-to-point-motions being faster, they are unforeseeable, indulging slower

linear motions for most industrial tasks. Finally, on the right, circular motions require three points to perform circular trajectories. In the last two motions, the end-effector orientation can be programmed to be either remain constant or mutable with the movement.

In order to replicate elaborate, curved paths, defined using multiple waypoints, a combination of linear and circular motions is necessary. With such approach, even after the laborious chore of finding the best sequence of motions, only an approximation of the requested path is obtainable. Moreover, the outcome motion is extremely difficult to predict since dynamic constraints are evaluated separately for each path segment. In other words, the next path segments are constructed while the robot is already in motion. If the robot is not able to perform one of those segment, a halt state will be reached and the process step has to be manually reprogrammed. In face of these problems, some robot OEMs provide an additional motion type suitable for curved paths. For example, KUKA Robot Language (KRL) offers the denominated spline motions, which group together several motions to create a single overall motion [36]. As dynamic constraints are now considered into a single motion, the requested path is predictable, more precise and easier to achieve. Unfortunately, KUKA does not inform in their manuals which mathematical approach is behind this command, making it complicated to adapt for tasks where an exact path must be followed. Recreating the same spline geometry would be sheer luck. But again, these spline motions are not the norm for all robot software. An example of a curve obtained through a set of linear motion and using a unique spline motion is depicted in Figure 2.7.

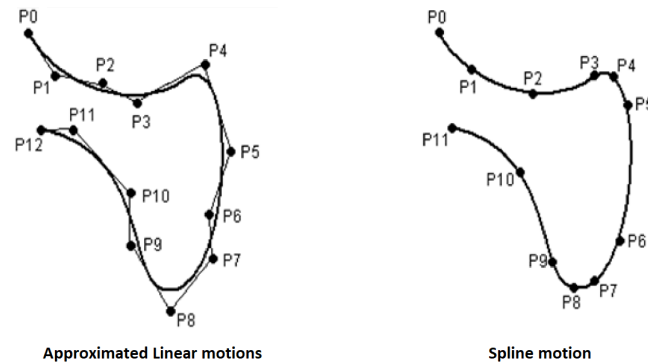


Figure 2.7: Complex cartesian paths obtained through multiple approximated linear motions or a spline motion. In the former, approximated linear motions mean that the movement of the robot is smoothed at the waypoints, thus never passing through them. Cited in [36], p.164

After selected the suitable motion for the task, the robot controller samples each joint trajectory into a series of intermediate waypoints. This process undergoes many hidden steps from the operator to make the robot motion as seamless as possible [34]. First, joints trajectories are smoothed by fitting it into a high-order polynomial, keeping the joint position and some of its time derivatives continuous throughout the motion. Generally, a 5th degree polynomial or higher is adopted so that velocities and accelerations at each joint waypoint can be specified. Furthermore, the waypoints are properly filtered to keep the joint torques in the range of the servomotor capabilities, hence obtaining very soft accelerations/decelerations. Finally, the command is sent to a servo controller, which contains a series of Proportional and Integral (PI) controllers to control the position, velocity and torque, in this order.

Most robot controllers fail to provide interfaces that permit low-level control of the robot manipulator. As mentioned earlier, manual joggling should occur to specify the robot configuration and the reason is the absence of environment recognition capabilities, possibly leading to collisions [37]. Additionally, the robot controller stealthily performs a series of approximations to achieve a smooth robot motion. The nondisclosure surrounding the robot controller architectures, the lack of judgment on the environment and the unpredictable robot trajectories make programming new tasks arduous and time consuming, alienating industrial manipulators to be efficiently adapted in many research projects.

2.3. Programming robot motions for deposition processes

As investigated by Lukaszewicz et al. [2], early patents of ATL systems date back to the very beginning of composites commercial sale, back in the 1970s. From that point forwards, ATL and AFP equipments have progressively matured and nowadays research on these technologies, also extensively surveyed in [2], have been

predominately concentrated on improving productivity, characterizing the process conditions and modeling the effects of layup. A subject that is often overlooked yet vital to bridge the breach between the designing and manufacturing phases is the development of software specialized in programming and controlling composite deposition. As will be reported throughout this section, the vast majority of research and software available is dedicated into fabricating conventional laminates, whereas specialized in VSL these are still non-existent.

Within the scope of programming AFP technologies, most research has been conducted on formulating algorithms to distribute fiber tows over open-contoured surfaces [37–39]. In generic terms, these algorithms begin by computing an initial reference line according to the requested fixed ply orientation and mold geometry, as depicted in Figure 2.8. Subsequently, the rest of the ply is built through adopting either parallel or shifted methods previously illustrated in Figure 2.1. Nevertheless, the design approach for conventional laminates is certainly very dissimilar than for VSL. With the exception for complex surfaces, the distribution of fiber tows for conventional laminates is quite intuitive, so that, when designing these composites, what comes to mind is how to select the optimal layup sequence. Whilst for VSL, the fiber tow distribution is a crucial design component and needs to be reflected with much anticipation, even for flat surfaces. In that regard, VSL design process generally involves sophisticated optimization methods that intrinsically distribute the fiber tows. As such, algorithms presented in [37–39] have to be significantly modified to cope with VSL design for complex surfaces.

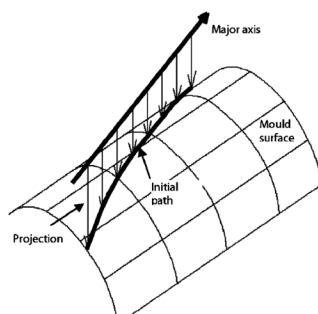


Figure 2.8: Initial path computed as the intersection between the surface and the projection of a major axis onto the surface. Cited in [38], p. 2.

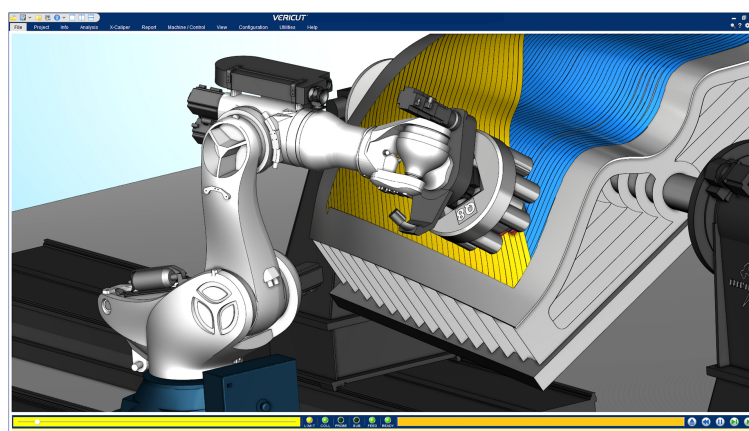


Figure 2.9: Illustration of an industrial AFP process obtained through VERICUT Composite Simulation. Copyright CGTech.

2.3.1. Commercial software packages to program AFP technology

In reality, most manufacturers adapt robot OEM software or implement third-parties commercial software to program the entire process planning for automatic layup technologies. Contemporary examples of the latter include FiberSim, CADFiber, TruComposites, VERICUT Composite (Figure 2.9) and ComposicaD (for cylindrical vessels). Available either as standalone applications or as a set of packages embedded in industrial standard Computer-Aided Design (CAD) programs (NX, CATIA, Autodesk and a few others), these software programs offer a 3D interactive graphical environment coupled with a set of programming tools to define the workspace, distribute fiber tows throughout the surface, program the robot motion and finally simulate the deposition process [40]. When the generated process is realistically acceptable, the corresponding sequence of robot motions is saved in a typical CNC program, written using a word address format commonly known as G-code, which is later downloaded to the robot controller. Instead of G-code, some include the ability to write directly robot specific languages, yet they are sparse to few robot brands.

For research purposes though, the adoption of commercial software invokes several obstacles. The majority of manipulator OEMs do not include a legacy support for G-code reading capabilities off the ground but instead offer it as a separate, expensive module, leading to an unnecessary aggravation of costs. Similarly to the native robot language, these proprietary software programs also dismiss the possibility of enhancing robot motion with independent algorithms. This is specially evident when heavy tasks are demanded, with these software struggling when employing redundant degrees-of-freedom (for instances a linear track with-/without mandrel) and checking for collision [10]. Lastly, either they are incompatible with VSL design or each VSL tow course has to be manually adjusted, ultimately making the preparation of a VSL production time-consuming.

Others software programs that combines robot graphical simulation with programming environment include KUKA.SIM for KUKA robots, RobotStudio for ABB robots, Robcad, RoboDK and Robotmaster, the latter three for multiple robot brands. Yet, not only do they suffer from the same previous hindrances, as they are also rather limited to typical industrial tasks. Therefore, current commercial robotic software packages are insufficiently integrated with deposition technologies, possibly resulting in superfluous robot motions and hidden constraints on the robotic system capabilities [2, 10].

2.3.2. Open-source software packages to program robot motion

Programming a robot requires proficiency in different and sometimes very specific fields, from control theory, environment modeling, motion planning to even sensor monitoring and many others. Thereby, the breadth of expertise for robot programming is further beyond the capabilities of a single developer. The latter premise composes one of the reasons why the development of robot computer applications is generally restricted to robot OEMs or software companies. Analogous to the publication of scientific articles, software researchers promote the sharing of knowledge by having their full source code publicly available. Anyone interested in testing the code can freely copy, inspect and even enhance it, if opportune refinements are proposed. What was formerly an application for a minor study case may be progressively improved through a collective effort and subsequently become a standard pragmatist for its and even other usages. With the growing demand for introducing robotic systems into innovative applications, open-source software could offer a cheap and versatile frameworks to compete against commercial programs.

The nondisclosure surrounding the robot controllers' architecture has compelled researchers to circumvent native robot languages and develop interfaces between those and more common programming languages. Sanfilippo et al. [41] developed a cross-platform communication interface in Java for KUKA manipulators through the use of a third-party tool (later becoming open-source) called KUKAVARPROXY. Despite supporting the execution of research code, due to the fact that KUKAVARPROXY does not have a low-level connection to the robot controller, real-time access to the robot state is unachievable, making control difficult for precision tasks.

In 2007, the Robot Operating System (ROS) was built by the sheer necessity of a lightweight framework that would offer robot-specific libraries and tools to make robot software development quicker and easier. As Quigley et al. explained in [42], they designed ROS not as an operating system per se, but to provide the functionalities of one, such as hardware abstraction, message passing and package management. Although other proprietary frameworks have dealt with the same necessity in the past [43, 44], ROS philosophies of free and open-source distribution captivated the interest of the robotics community and is currently being widely employed for research purposes. One of its main strengths is the ability to connect multiple onboard and off-board computers in a peer-to-peer topology, that is, digital hosts hand out messages directly to one another, without the need for a central server. ROS was designed to be language-neutral, in other words, to support a wide variety of programming languages, also promoting the reuse of code previously developed. Modularity is also a significant feature in ROS, with executables being individually designed and tested. Popular robot-specific libraries available in ROS include, for example, Rviz and MoveIt. While Rviz is 3D visualization tool, MoveIt provides libraries to state the robot model representation and grant environment recognition capabilities [45].

Since ROS is only specified at the messaging layer, information is then shared without modification of the original source software. A direct consequence is the possibility of integrating software and hardware developed by different people or institutions, encouraging collaboration between researchers and the industry. Hereby, ROS developers saw an opportunity in their framework to extend industrial robots capabilities, creating a consortium of ROS named ROS-Industrial (ROS-I) [5]. Currently, the ROS-I repository includes interfaces to several machine vendors, including ABB, Fanuc, KUKA and Universal Robots, and also advanced tools to plan robot motions. These last interfaces work as a communication channel between the respective robot controller and an external computer, making use of OEMs software to allow transferring commands at a much lower level. Until this work, programming composite deposition processes with ROS libraries was never addressed in the past.

2.3.3. Enhance industrial manipulator performance

The topic of improving robot performance for composite deposition processes has been rarely addressed in the literature. Debout et al. [46] developed a tool path smoothing method for redundant systems with the primary goal of reducing AFP manufacturing time without sacrificing product quality. In an attempt to alleviate robot configuration constraints, a geometrical domain was specified where changes of $\pm 1^\circ$ in the

tool rotation would be admissible. Oscillations in the speed of each robot axis were mitigated by applying a piecewise filtering method. A critical drawback of applying smoothing strategies for composite production is that the manufactured fiber path may differ from the designed fiber path. Despite Debout et al. advocating that this small difference does not compromise quality of conventional composites, this trade-off has yet to be further pursued for VSL design.

In 2018, Gao [10] developed one of the first motion planners for redundant systems specialized in composite deposition. The proposed algorithm begins by discretizing the original continuous problem, generating a multi-layer graph structure containing all robot configurations that do not violate any collision or kinematics restrictions. It is then pursued by a sequential optimization process that analyzes the obtained graph structure while searching for the path with the shortest traveling time. The results obtained from Filament Winding (FW) suggest that improvement in the deposition productivity can be achieved if this particular motion planner is adopted, as it ensured that either the velocity or acceleration of one of the robot manipulator's actuators always reached its maximum value. However, by choosing to adopt the native robot language (used a KUKA manipulator), the designed curvilinear path is expected to diverge from the one generated by the robot controller.

Both algorithms previously explained have taken different approaches to solve a common problem: improve productivity of redundant industrial systems. An insight into redundancy and its intrinsic issues can be found in Appendix A. The absence of quality assessment either through accuracy measurements or mechanical testing of real laminates undermines both Debout's and Gao's efforts. Since both researches did not assume the possibility of consolidating during the deposition stage, the laydown speed could be aimed as high as possible. For processes with in-situ consolidation, steering with high laydown speeds can decrease the peel strength of the composite produced [8]. Moreover, Lukaszewicz in [47] states that higher laydown speeds will doubtfully yield higher productivity, since nowadays deposition processes spend less than half of the processing time in actually laying down prepreg tows and the remaining in secondary operations, such as material refilling, cutting and cleaning.

While there are more published works on the topic of enhancing robot performance for composite deposition process, some of these are protected by patents or cannot be applied for VSL design at such early stages. For example, the company Electroimpact published an article [48] featuring an advanced AFP robotic system that promises high path accuracy and process flexibility. From their short explanation, a custom drive for the six axis was developed and accuracy was controlled by adding an external encoder at each joint. Furthermore, other researchers have also been developing motion algorithms for very specific composite processes, such as needle-punching [49] and filament winding [50, 51] of L and Y shape workpieces, respectively.

2.4. Motion planning for industrial manipulators

An industrial robot can be joggled from an initial to a goal configurations following countless trajectories. However, not all of these outcomes are reasonable for the requested task. Obstacles scattered through the workspace, collision between links of the robot's own structure or even constraints imposed by the operator diminish the number of viable trajectories. For these reasons, robot controllers are provided with an algorithm that evaluates possible motions and selects a single solution, typically by minimizing an objective function. This section will then introduce widespread strategies to generate a feasible robot motion for the assigned task, commonly referred to as motion planning algorithms.

Before describing in more depth the definition of motion planning, a distinction between the terms path and trajectory must be first introduced. A (geometric) path denotes a geometric description of the robot motion, that is, it constitutes a sequence of poses, in the joint or operational space, which the manipulator pursues to execute the assigned task. In contrast, a trajectory comprises a path coupled with a timing law, in which the velocity and/or higher time derivatives are specified at each pose.

Following the notation used by Gasparetto et al. [52], motion planning involves computing a sequence of feasible system configurations that reproduce a given trajectory without collisions and satisfying velocity and/or acceleration constraints. In order to deal with the complexity of planning a robot motion, the algorithm is subdivided into two parts: path planning and trajectory planning. The goal of a path planning algorithm is to generate a collision-free path from a predefined set of waypoints, either in the joint or operating space. Trajectory planning consists of assigning a time condition into the generated path such that ensures the execution of a path within the acceptable range of velocities and accelerations.

Motion planners were early divided into two branches: offline and online motion planners [53]. Offline planners compute a complete sequence of robot configurations before the motion itself begins. By having a

complete knowledge about the environment, repeatably and predictability of the robot's set of movements are guaranteed, vital properties to ensure robustness of industrial processes. On the other hand, online planners incrementally generate or modify a trajectory while in motion and perceiving the environment through sensor readings. In spite of decades of work developing motion planning algorithms, both offline and online methods were undermined by the technology at the time, with offline processes requiring complex problem solving and online processes unable to find the optimal solution fast enough [53]. Nowadays, computers are powerful enough to even plan densely sequence of points in the operational space. Consequently, most industrial processes have adopted an offline programming approach, where the robot motion is generated outside the production environment, though an offline planner, allowing to simulate and analyze the motion before its execution. Only offline methods will be described from now onward.

2.4.1. Path planning

A popular distinction between path planning paradigms is done between sampling-based and optimization-based path planners. A sampling-based path planner searches for a least-cost path by selectively sampling the robot's configuration space and building a graph that progressively explores its entire space. One strong advantage of this approach is the possibility of always finding the global optimum solution of the motion problem. However, it usually returns paths that are far from optimal if not enough time is given, especially in higher dimensional problems [54]. In Figure 2.10, the impact of time in the sampling search is visualized by two graph trees obtained with different iteration numbers. When this happens, a secondary step may be introduced to smooth the output of the path planner, as proposed in [55]. In contrast, an optimization-based path planner finds an optimal solution by minimizing a cost function subject to inequality and equality constraints. Although rapidly converging towards one local minimum, the resulting path ultimately depends on how the path is initialized [56]. Consequently, there is a chance that the converged solution becomes trapped into a poor quality local minimum, never capable of reaching the global minimum. In such cases, collisions might still occur [57]. Furthermore, it is possible that no solution is found when using large point clouds [57].

Adopting ROS gave the opportunity to test refined path planners, previously conceived for mobile or humanoid robots, in industrial manipulators. Three state-of-the-art path planning algorithms will be analyzed in this project, two based on sampling schemes (RRTConnect and Descartes) and the one remaining in optimization methods (TrajOpt). Each one of these planners will be briefly described in the next few paragraphs. Note that the mathematical description behind their collision avoidance will exceed the scope of this thesis.

In 1998, LaValle [58] introduced the Rapidly-exploring Random Tree (RRT) concept as a path planning algorithm that builds randomized data structure and examines unexplored robot configurations in search of an optimal path from a predefined initial and goal configurations. The original RRT concept aimed to paths with a reduced joints displacements, but in subsequent years this concept was rapidly extended to meet other different objectives. Two years later, a variation of RRT method named RRTConnect improved efficiency by growing two RRTs simultaneously, one from the initial configuration and a second from the goal configuration, followed by an attempt to connect them [59]. The simplicity of the algorithm has made RRTConnect one of the most reliable path planners in the Open Motion Planning Library (OMPL) [60], a library that stores state-of-the-art sampling-based path planning algorithms.



Figure 2.10: Representation of RRT graph trees after 45 iterations and 2345 iterations. Cited in [54], p.230.

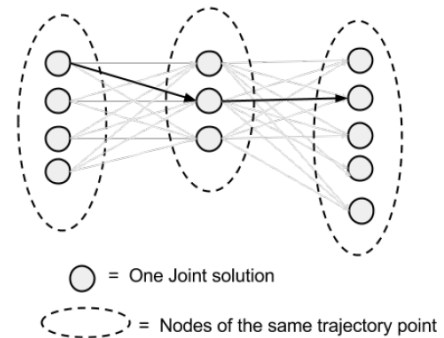


Figure 2.11: Schematic representation of the Descartes Dense algorithm. Cited in [61], p.4.

Descartes was released for ROS-I as a deterministic and globally optimum cartesian path planner, intentionally developed to handle tasks where the robot has to follow an exact tool trajectory. Yet, its developers

did not publish a formal description of the theoretical foundation behind the algorithm. This absence was dealt with in [61], where an analysis of its source code was conducted, followed by an experimental evaluation for arc welding applications. In essence, Descartes offers two choices to how to plan a robot motion, either using a brute force algorithm, called Dense planner, or a lightweight version of it, named Sparse planner. Descartes Dense planner is at the core identical to RRT, except randomness is removed by evaluating all possible robot configuration for all path points, hence always returning the global optimum path. On the other hand, Descartes Sparse planner only performs inverse kinematics for a reduced number of waypoints and the remaining path is interpolated in the joint space. Without the need to analyze all the waypoints, the Sparse planner it is generally faster than the prior Dense graph search [61]. Still, this oversimplification may result in erroneous paths, hence an additional step is included in the Sparse planner to checks the accuracy using forward kinematics and, if it is outside a tolerance range, inverse kinematics is performed to a previously interpolated waypoint one at a time, until an acceptable path is achieved. In Figure 2.11, the construction of a graph tree using Descartes Dense is exemplified. For simplicity, that graph was developed for a path containing three waypoints (also called nodes or vertices) and for a single joint. In this example, in the first waypoint four valid joint positions were found, in the second waypoint three positions and in the third waypoint five positions. All possible choices for path segments (also named edges) between two consecutive waypoints were evaluated and only one was selected based on the smallest difference of two joint positions.

In 2018, the ROS-I repository received a highly anticipated path planner by the name of TrajOpt. As stated in [56], TrajOpt solves the non-convex problem that is robotic path planning by dividing it into a series of sequential convex problems. The algorithm minimizes the sum of squared displacements between consecutive waypoints and uses a iterative penalty method to satisfy kinematic (joints position, cartesian pose and collision) and even some dynamic (joints velocity, acceleration, jerk and also cartesian velocity) inequality and equality constraints. As stated by its developers, TrajOpt can efficiently compute locally optimal, collision-free paths, even with an infeasible initial path. Its superiority to other optimization path planners is based on the fact that TrajOpt does not include any elements of stochasticity [56].

2.4.2. Trajectory planning

When assigning a time condition to each path point, dynamic constraints are imposed, either in joint or operational space. In the interest of finding a minimum-time trajectory that complied with torque limits at the robot joints, Shin and McKay [62] introduced in 1985 a methodology that ended up becoming the foundation of trajectory planning, known as time parameterization. Their approach lied in dividing the problem into two elements: path geometry and timing law. The path geometry is obtained by interpolating the geometric path with a piecewise polynomial function (Splines, B-splines or NURBS), allowing to describe the set of robot configurations using a single parameter, named path parameter. The timing law is then a function that relates this path parameter with time. How to find the timing law that satisfies velocities, accelerations and higher-order time derivative is what distinguishes most research approaches. The key aspect in time parameterization is the reduction of the multidimensional space, in which the robot operates, into a single or double degree-of-freedom system, defined by the speed and/or acceleration in function of the path parameter [53].

Trajectory planning for industrial applications typically relies on generating trajectories in the operational space that are path-accurate and follow a desired velocity profile [63–67]. Still, most efforts in this field have been dedicated to minimizing an objective function in the joint space. Examples include minimum-time planners [68–72], searching for the shortest execution time trajectory according to the given points, and minimum-jerk planners [73], focusing on reducing vibration, precision tracking and ensuring durability over the joint life. Hybrid approaches that combine both minimum time and jerk were also developed in [74–76] to enhance simultaneously the efficiency and smoothness of the trajectory.

Two strategies of trajectory planning using time parametrization will herein be presented. The first approach is widely employed on CNC machines [63–67] and consists of assigning a speed profile to the end-effector frame, commonly known as Feedrate Scheduling. The second approach, called Time-Optimal Path Parameterization (TOPP), seeks to minimize the traversal time by limiting each joint motion into two speed limit curves imposed by their maximum velocity and acceleration [69–71].

Trajectory planning strategies in the operational space, such as Feedrate Scheduling, are scarce and mostly applied to machining processes, where changes in the end-effector speed (in CNC named feedrate) can impair the surface finish [65]. The goal of Feedrate Scheduling is to plan a trajectory for a desired set of tool coordinates which guarantees an almost constant feedrate throughout the task. Most of these algorithms start by sketching an end-effector speed profile through analyzing the requested path geometry and cartesian dynamic constraints. Afterwards, the timing law is extrapolated using the chain rule and either the Tay-

lor expansion or predictor-corrector methods. With the timing law obtained, the inverse kinematics can be performed to obtain the respective joints trajectory. A graphical representation of one of these algorithms is visualized in Figure 2.12.

Evoking a constant speed at the end-effector for the whole path is physically unfeasible since robots cannot instantly switch the speed acting on a joint from zero to a desired value (demands infinite accelerations). In this way, the classical approach in Feedrate Scheduling is to define the tool acceleration with a bang-bang profile and its corresponding speed with a trapezoidal profile, separating the motion into three stages: acceleration, constant speed and deceleration. Still, to ensure complete continuity of the jerk and other higher time derivatives, complex curves are often adopted to characterize the tool's motion, with bell profiles (also known as double S-curves) becoming a logical choice for the tool speed [77], as represented in Figure 2.13.

As indicated earlier, this approach of trajectory planning was purposely developed for CNC machines and has been broadly employed for paths interpolated with B-Splines [63, 64] and NURBS [65–67]. Olabi et al. [65] exploited the properties of Finite Impulse Response (FIR) filters to smooth trapezoidal velocity profiles, allowing to analyze jerk at the end-effector and suppress vibrations caused by their discontinuities. Lee et al. [66] proposed an additional offline step that anticipates crucial points with large curvature shifts in a NURBS, followed by its division into several NURBS sub-curves, each with a feedrate profile build according to the block length and the imposed limits of cartesian acceleration and jerk.

Planning in the cartesian space with Feedrate Scheduling bestows hindrances that compromise the use of a robotic system at its full capabilities. The maximum acceleration at the end-effector is imposed by the lowest acceleration to saturate one of the joint actuators. Naturally, such conservative approach cannot fully utilize the performance of the remaining joints. Additionally, most of the literature has adopted robot systems that contain prismatic joints to coordinate the end-effector position, making the premise of restricting the maximum velocity and acceleration at the tool exactly to the joint limits reasonable. However, in robot manipulators with revolute joints, such assumption is not as intuitive and ultimately requires a post-processing step to check if the joint limits have been surpassed. As visualized in Figure 2.12, in case the robot limits are exceeded, a new velocity profile followed by inverse kinematics would have to be computed again, which is rather computational demanding.

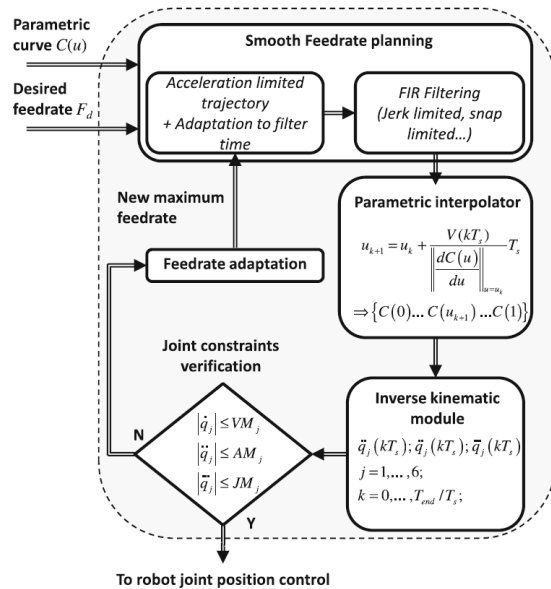


Figure 2.12: Schematic representation of a feedrate scheduling algorithm for a revolute manipulator. Cited in [65], p.472.

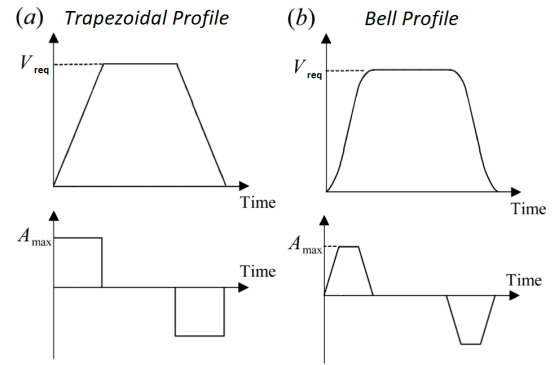


Figure 2.13: Cartesian trapezoidal and bell velocity profiles and their corresponding accelerations profiles.

For the joint space, TOPP algorithms can be traced back to the mid 1980's [68] and has since become the leading research topic regarding trajectory planning. Rather than replicate a requested velocity profile, the goal of a TOPP algorithm is to find the fastest trajectory by optimizing the timing law while respecting dynamic constraints imposed by the actuator capabilities or the desired task. Different methodologies have been developed, but in essence, a TOPP approach consists of confining the joint dynamic variables into an interval, which in return imposes boundaries to the time derivatives of the path parameter in the form of limit curves. A representation of these limit curves is visualized in Figure 2.14.

The time optimal velocity profile is generated by integrating a bang-bang acceleration profile, alternating between maximum acceleration and maximum deceleration along the path. The selection of waypoints where acceleration modifies, also known as switch points, constitutes one of the most computational demanding components of these algorithms and is usually a result from the intersection between two limit curves. Given an initial and final velocities for each joint trajectory, most algorithms start by integrating forward with maximum acceleration. When the next waypoint overcomes the velocity limit curve, the algorithm searches for the next switch point. Then, starting from the switch point, the algorithm integrates backward with minimum acceleration until the current point is reached. A representation of the algorithm can be visualized in Figure 2.15.

Based on this approach, Kunz and Stilman [69] developed a TOPP algorithm that is currently supported in MoveIt. In their approach, the trajectory must stay below two limit curves, one originated by joint acceleration constraints and the other by joint velocity constraints, as visualized in Figure 2.15. Pham [70] generalized Kunz and Stilman's work by developing a faster TOPP algorithm that included the dynamic motion equation and torque constraints. The speed of the algorithms was improved by simultaneously integrating forward from the start point following the maximum acceleration curve and backward from the end point following the maximum deceleration curve, terminating by combining their result. A couple years later, Pham [71] reorder his prior approach such that switch points are no longer required to be computed.

Trajectory planning based on TOPP has disadvantages that should be taken into account. As any planner in the joint space, the end-effector pose is hardly foreseeable. Since TOPP algorithms are performed in a post-processing step, the generated path may diverge from the desired path. To ensure that no obstacle is hit, additional collision checks may be necessary [53]. Also, neither of the three mentioned TOPP is jerk limited, hence the trajectory may contain hard acceleration switches.

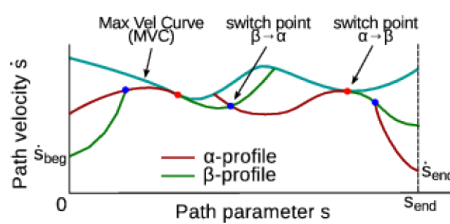


Figure 2.14: Representation of TOPP limit curves in a velocity plot, with s representing the arc length. The path acceleration lower and upper bounds are given by α and β , respectively. Cited in [70], p.3.

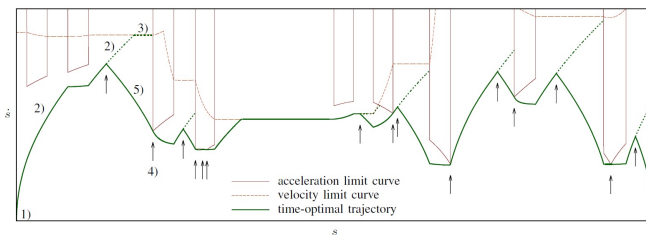


Figure 2.15: Schematic representation of a TOPP algorithm, where 1) Initial velocity, 2) Forward Integration, 3) Reach the limit curve, 4) Search for a switch point and 5) Backward Integration. Cited in [69], p.5.

In order to generate trajectories that are path-accurate, Lange and Albu-Schäffer [78] proposed a trajectory planner that synchronizes with the desired path, while complying with velocity, acceleration and jerk constraints. Instead of forward and backward integration, forward scaling and backtracking are executed iteratively, where the dynamic terms are modified by a scaling factor if the obtained pose does not correspond with the desired pose. Even though it only works in the joint space, jerk limits can be taken into account, contrary to previous TOPP algorithms described.

3

Research Project

Initial experiments proved the weight-saving benefits of VSL over traditional composite design, in particular for cases where buckling and post-buckling are critical. Yet, a tendency to optimize VSL layup sequence without proper experimental validation was likewise observed in the literature. The burden of programming curvilinear tow courses with current robot software systems, with the latter optimized for a limited set of industrial applications, comprises one of the main reasons for this absence. The inherent inaccuracy of industrial manipulators to execute complex cartesian paths demands performing a series of long and repetitive trials to refine the robot motion and reach the aimed VSL tow course.

In addition, the nondisclosure surrounding the modern robot controllers' architecture has hindered having control over the layup process, thus preventing researchers from taking full advantage of the robot's capabilities. The implementation of state-of-the-art path planning algorithms could drastically improve the robot performance, but this requires an openness from the machine suppliers. A collaboration between academia and industry is therefore crucial to successfully ease robot integration into novel industrial processes, with the ROS framework seeking to provide a pathway for efficient and economical software development.

3.1. Research question and objectives

Analyzing the behavior of robot manipulators for composite production has rarely been addressed in previous research and major improvement sparsely documented. At least three challenges are currently undermining the VSL production: unpredictable layup performance, laborious integration of curvilinear tow courses and irregular laydown speeds. This thesis aims to fill these gaps and is formulated to answer the following research question.

How to enhance control of industrial manipulator systems to manufacture variable stiffness laminates?

To answer the main research question, four sub-questions were formulated as follows:

- How should a discrete representation of the tow centerline be arranged to generate smooth robot motions?
- For a six-axis industrial robot, which of the available path planning algorithms yields superior performance for fully-constrained cartesian paths?
- Can a desirable laydown speed remain constant when tows with reduced turning radii are deposited?
- Does external control of an industrial manipulator ensure an accurate execution of the assigned task?

The research objective of this thesis project was to reduce the variability of layup control and the iterative experimental steps associated with programming fiber tow courses in industrial manipulators. This objective was achieved by developing a process step which automatically generates suitable robot configurations to first simulate and then accurately execute a given tow course with constant laydown speed.

3.2. Hypotheses

Modern robot controllers are designed to provide a simple and intuitive user experience. Smooth robot motions are conveniently obtained without the user exercising a complete oversight of the exact planned trajectory. In contrast, with external control, data between the robot controller and an external driver is exchanged without passing through the several high-level layers of the system software. This direct communication allows for more freedom in describing the motion task without the need of linear or circle motion types, ultimately providing a greater customization of the robot movement. Bear in mind that the actual control loop of the system is still performed by the robot controller. External control is just a way to communicate at a much lower level. One hypothesis to be corroborated in this thesis is that a precise representation of the intended tow course can be achieved with external control of an industrial manipulator.

In regards to path planning performance, recent planners Descartes and TrajOpt, accessible through ROS-I repository, are predicted to outperform older RRT algorithms for industrial tasks. The reasons to support this claim are in the fact that both Descartes and TrajOpt are deterministic and were developed while taking into consideration operational space tasks. Since RRT and other OMPL algorithms are inherently probabilistic, a same request may return different solutions, which might create unexpected unnatural robot motions.

The developed process step is designed to address the irregularities in laydown speed while laying down VSL tow courses and is based on the hypothesis that a constant laydown speed improves the quality of automated composite processes. In fact, variability in laydown speed was never clearly identified in previous literature. For conventional laminates, linear motions provided in industrial systems can maintain a constant end-effector speed. Additionally, most of VSL manufactured were performed with a simple circular motion, again where the end-effector speed can be kept constant. However, for complex curvilinear tows courses, variations in the end-effector speed will be noticed when changing from linear to circular motion and vice versa, possibly leading to temperature fluctuations and therefore variability in the bond quality.

3.3. Limitations

Throughout the literature, most efforts in designing and manufacturing VSL have been conducted using ATL and AFP deposition technologies. For this reason, the developed process step was focused on creating robot motions for the last two technologies. Its extension to Filament Winding (FW) would require the introduction of one extra degree-of-freedom to rotate the mandrel which, in addition to the already 6 degrees-of-freedom of the manipulator, would lead to a redundant system. With redundant systems, the motion problem becomes significantly more complex, hence the generation of robot motions for FW was kept outside the scope of this thesis.

Although the developed process step is agnostic to the different robot manufacturers, experiments conducted in this thesis were restricted to a KUKA industrial robot. KUKA provides its robots with a real-time communication interface, allowing them to be used at their maximum performance. Other robot brands may adopt different forms of communication, possibly leading to a different robot behavior. However, this should not have much impact on the performed end-effector path, at worst the robot may not execute the command due to requesting a high torque at one of the joints. The latter could be prevented by reducing the acceleration limits hardly imposed.

A high number of ROS packages employed in this project, including for example TrajOpt, its motion planning framework and even the robot drivers, still have an experimental status. This term is used by the ROS community in cases where a package has known issues or missing functionality and is still under heavy development. This dependence on others open-source libraries meant that some existing features that could improve the analysis were not available or were not explained how to set them up. The results presented here are confined to what is available in them at the time of writing.

III

Framework Developed

4

Experimental Setup

Before addressing the framework developed, some prior knowledge about the tools and equipment employed in this project should be first introduced. On this account, this chapter aims to describe the several components which were brought into play when trials were performed. The experimental setup was divided into three groups of components: first were the robot components, designed and delivered by the robot manufacturer; second were the ROS components, open-source software libraries to simulate and execute robot motions; and at last were the deposition specifications, details on how the composite should be lay down. Describing the components of these three groups is the purpose of this chapter.

4.1. Industrial manipulator components

The experimental tests were executed on a KUKA industrial manipulator. One should note that any other industrial robot supported by ROS would also be suitable. At the time of this thesis, the manipulator system was located at Delft Aerospace Structures and Materials Laboratory (DASML), part of TU Delft. The several constituents of this system are described in the next paragraphs.

- **Robot Manipulator:** The robot model used was the KR 210 R2700 extra (Figure 4.1), where the first number denotes its maximum payload (210 kg) and the second value its working radius (2700 m). Even though the robot was anchored on a linear track, the former remained stationary for all activities of this thesis. For safety purposes, a machine safety fence surrounded the robot cell. Additionally, the AFP-XS deposition tool from AddComposites was attached to the tool plate. The robot cell is depicted in Figure 4.2. In regards with its specifications, the manufacturer indicates a pose repeatability (largest discrepancy for a tool pose) of 0.06 mm and in Table 4.1 the range of each joint and their maximum velocities are provided. Unfortunately, KUKA does not supply the limits for higher time derivatives of each joint position, namely its acceleration, jerk, snap and so on. For this project in particular, the absence of accelerations limits required to make some assumptions. Instead of listing a maximum acceleration for each joint, a singular acceleration limit of 2 rad/s^2 , around $114.6^\circ/\text{s}^2$, was assumed for all joints. This value is definitely very conservative, but sufficient to prove the concept evaluated in this project.
- **Robot Controller:** To coordinate the manipulator movement, the robot controller KR C4 was adopted. For his study case, the robot controller received a command from an external source, a computer with ROS installed. When an external motion command is received, the robot controller interpolates and smooths the motion so that no discontinuities in the joints position and their time derivatives are exhibited. With this approach, however, the validity of the command is never checked by the robot controller before the robot motion is initialized. If a velocity, acceleration or jerk limit is exceeded, the manipulator comes to a halt state and the motion program has to be restarted. It is therefore crucial to ensure that the external command does not overcome the robot limits. For simplicity, power cables, Ethernet cables and all remaining necessary cables were also included in this component.
- **Software:** The robot controller was equipped with the KUKA System Software (KSS) version 8.3 [35] and with it came a proprietary teach pendant with a touch screen. In order to quickly jog the robot, the

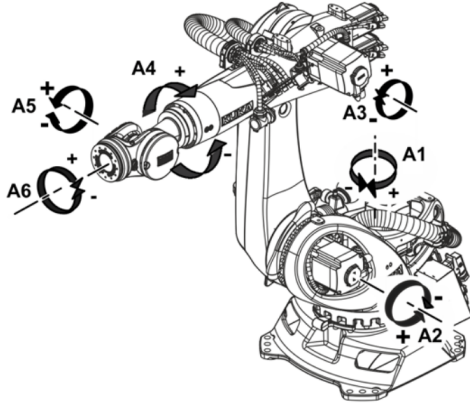


Table 4.1: KUKA KR 210 R2700 extra specified limits. Cited in [79], p.17.

Axis	Range	Speed with rated payload
A1	$\pm 185^\circ$	$123^\circ/\text{s}$ (2.147 rad/s)
A2	-5° to -140°	$115^\circ/\text{s}$ (2.007 rad/s)
A3	-120° to 155°	$112^\circ/\text{s}$ (1.955 rad/s)
A4	$\pm 350^\circ$	$179^\circ/\text{s}$ (3.124 rad/s)
A5	$\pm 125^\circ$	$172^\circ/\text{s}$ (3.002 rad/s)
A6	$\pm 350^\circ$	$219^\circ/\text{s}$ (3.822 rad/s)

Figure 4.1: KUKA KR extra: a 6 revolute degrees-of-freedom robot manipulator. Cited in [79], p.30.

teach pendant includes a set of buttons to modify the joints position and even a 6D mouse to intuitively change the end-effector pose. For industrial tasks, robot motions are typically programmed using their KUKA Robot Language (KRL) [36]. With external control however, much of the robot's higher-level path planning abstractions provided by KRL was bypassed. To enable external control, only a generic KRL program was executed, which basically granted permission to activate external control, connected the robot controller with the ROS computer and waited for a motion command.

- **External communication interface:** The connection via Ethernet between the robot controller and the ROS computer was established through an extension package provided by KUKA named Robot Sensor Interface, or simply RSI [80]. With the RSI package, robot and ROS data are cyclically exchanged at a frequency up to 250 Hz (sampling time of 4 ms). This high bandwidth and low latency interface enables real-time processing of data from an external sensor (here the ROS computer) which can subsequently be used to influence the robot's behavior. The sensor output consists of either cartesian or joint space configurations and their respective time conditions. RSI then uses this data to interpolate, for each sampling time, an offset displacement (or, as KUKA calls it, sensor correction) to reach the desired configuration from an initial configuration. With this approach, neither velocity nor higher time derivatives can be directly submitted to the robot but instead are implicitly imposed by the time conditions. Through using the command `RSI_MOVECORR`, the robot was controlled purely by means of these sensor corrections and no motion types provided in KRL were required to specify. The maximum allowed corrections are ± 5 mm for cartesian displacements and $\pm 5^\circ$ for joint displacements. In addition to the previous capabilities, RSI also provides other advanced functions for controlling and filtering the sensor data. Yet, these were kept to a minimum, as the focus is on the framework developed, agnostic from robot suppliers.



Figure 4.2: Robot cell adopted for the experiments in this thesis.

4.2. ROS components

A ROS application is organized into packages, each containing a collection of files either with executables, datasets or other parameters. These packages are then distributed via git repositories. Several packages were adopted for this project, and their repositories will be referenced as footnotes within this section.

A basic background on ROS nomenclature and how it operates is next described. The term node is used to designate an independent executable process. A typical ROS application is formed by a cluster of nodes, with each node responsible for a certain task within the application. By creating a launch file, all nodes of a ROS application are started in succession. When initiating a ROS application, a master node is created which registers the parameters and individual nodes in operations, enabling nodes to locate each other and exchange messages. The communication between nodes is carried out using topics, services and actions, as presented in Figure 4.3.

To describe a robot model, ROS employs a dedicated file format named Unified Robotic Description Format (URDF). An URDF file contains all the details about every link and every joint of the robot, which combined constitutes the tree structure of the robot. Links are described by their coordinate frame, inertia and geometric design. The geometric design is derived from a 3D mesh of standard CAD programs. Typically, a refined 3D mesh is used to visualize the links in a graphical interface. For collision checking, however, the computation on those complex meshes becomes too demanding. Hence, it is common to describe a link with two CAD files, one refined for visualization purposes and a simplified surface mesh for collision checking. Joints are described by their axis of motion and the pair of links that they are connected to. Furthermore, limits to their range and velocity have to be specified. Another adjacent file that is relevant to mention is the Semantic Robot Description Format (SRDF), which contains the wording used to describe the each link, joint and the complete robot chain.

Due to the free and open-source philosophy behind ROS, this ecosystem is best supported on Ubuntu operating system. The experiments presented in this thesis were conducted on a computer installed with Ubuntu 16.04.6 64-bit and ROS Kinetic, equipped with an Intel i5-6500 CPU (Central Process Unit) and 8 GB of RAM (Random-Access Memory). The following paragraphs group the various libraries that were used to build the deposition framework.

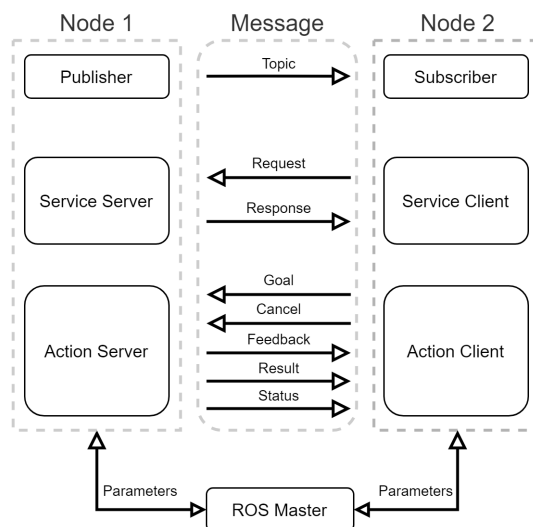


Figure 4.3: Schematic representation of ROS mechanics for message passing. When adopting an one-way transport mechanism, one node publishes a message on a topic, while a second node receives the message by subscribing the corresponding topic. Services enable a two-way communication mechanism, where the client node sends a request to the server node, and the latter computes and replies with the corresponding response. Action server/client denomination can be seen as an asynchronous approach to services, where the client has a constant feedback on the incremental progress towards the goal and is allowed to abort at any moment if a specified tolerance is violated.

- **Robot model and drivers:** The ROS-I community has been successfully creating and maintaining drivers for a wide range of industrial manipulator's manufacturers. At the time of writing, KUKA robots' files were stored in the `kuka_experimental`¹ repository. As the name insinuates, this repository is still under much development and further testing are required to validate its usage. For this project, the

¹https://github.com/andreflorindo/kuka_experimental (accessed 5 April 2020)

kuka_experimental (from an in-house fetched repository) provided not only the URDF file for the robot in question, but also a custom hardware interface. A hardware interface is the layer that connects the various ROS controllers with the real robot. To fulfill its purpose, the task of a hardware interface is as follows: read the current state of the robot, communicates the state with the respective ROS controller(s), update the next desired command and write that command back to the robot. As RSI does not accept either joints velocity or acceleration values, only the joints position are transmitted. As such, the hardware interface offered by kuka_experimental is subdivided into two interfaces, the Joint State Interface, which receives the joints state at every control loop, and the Joint Command interface, which publishes the joints command from ROS to the robot controller. Figure 4.5 presents a flow diagram of this pipeline. Without direct connection with the robot controller, simulations could also be executed with industrial_robot_simulator², a generic driver for industrial manipulators. Although not a precise representation of the drivers from kuka_experimental, works great for quick visualization of the motion when no real testing is required.

- **Robot Cell Description:** Similar to how a robot model is described, an URDF file was developed to characterize the whole automation cell. The automation cell used in this project is depicted in Figure 4.4 and was composed of a KR 210 R2700 extra, fixed to a linear track, a table, where fiber tows will be laid down, and lastly the AFP-XS deposition tool³.
- **Graphical Interface:** Within a URDF file, each link provides a path that specifies the location of the CAD mesh. In order to display these meshes, a package named Rviz was adopted. Rviz is a 3D visualization tool that comes embedded with the ROS installation. Besides rendering the automatic cell and updating the robots poses as it moves, it also displayed the requested tow course. The robot poses were constantly updated by subscribing to a topic published by the node robot_state_publisher. The node robot_state_publisher converts joint space vectors into a series of transformation matrices, one per link's frame, and publishes the cartesian position and quaternions for each link.
- **Controller:** Abstract controllers and hardware interfaces are contained in a collection of ROS packages of name ros_control [81]. Figure 4.5 exemplifies how data flows in ros_control. The custom hardware interface provided in kuka_experimental adopts one of these generic controllers. In this project, the position of a joint was controlled by the joint_trajectory_controller. This controller receives the desired joint position, velocity and acceleration trajectories and with them construct a new joint (position) trajectory using a quintic spline. The interpolated waypoints are then provided to the Joint Command interface at a specified rate. Since neither velocities, accelerations or torques can be controlled with RSI, the ROS controller actually does not require to be tuned with any proportional, integral or differential gain. The robot controller handled the position error and therefore the trajectory controller simply forwarded the desired joints positions to the robot.

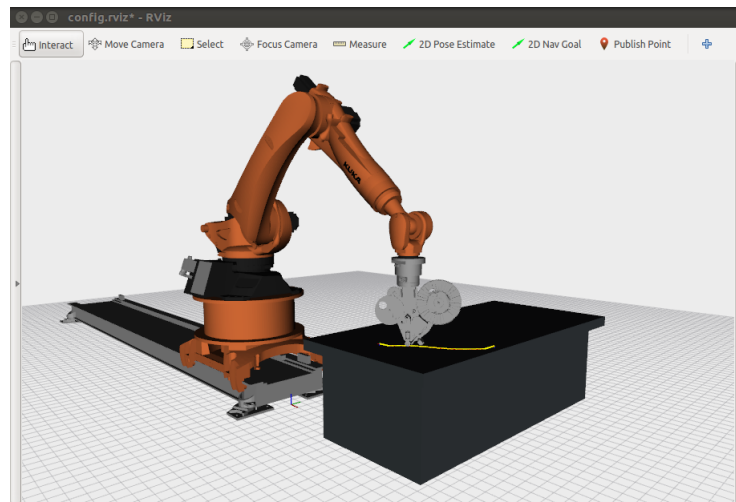


Figure 4.4: Replica of the experimental robot cell in Rviz.

²https://github.com/ros-industrial/industrial_core (accessed 5 April 2020)

³https://github.com/andreflorindo/kr210_cell_support (accessed 5 April 2020)

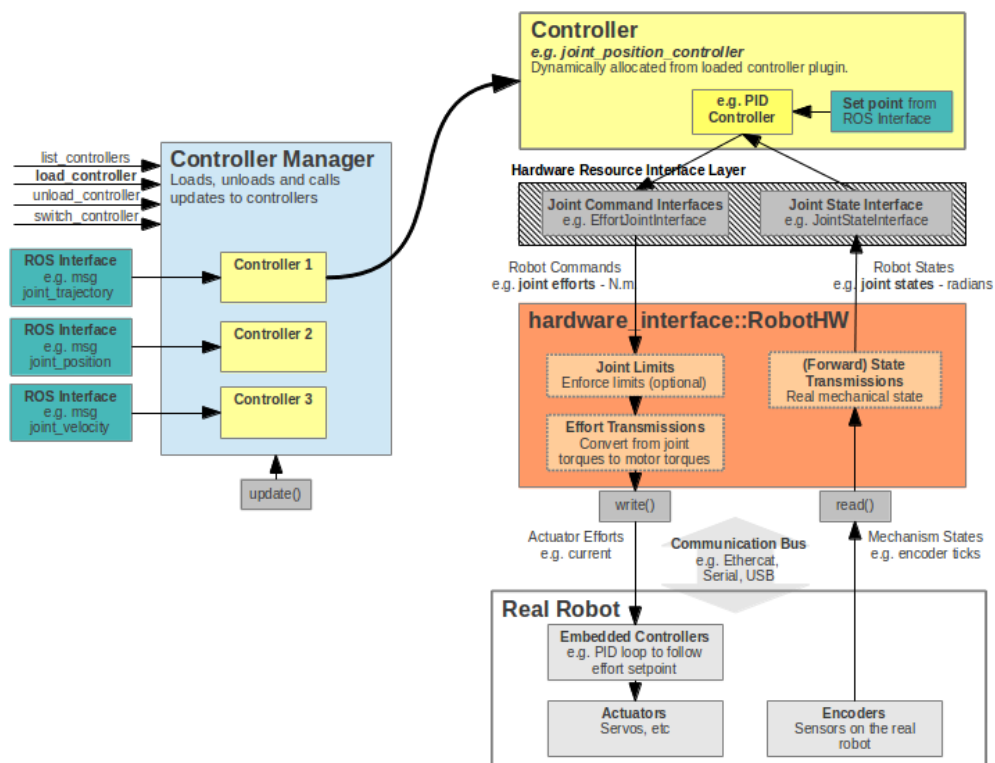


Figure 4.5: Schematic representation of ROS control. Cited in [81] , p. 2.

- Motion planning framework:** A motion planning framework aggregates a set of tools for interpreting the robot description, monitoring the robot state and computing non-collision robot motions. Three motion planning framework were tested in this project: MoveIt⁴, Descartes⁵ and Tesseract⁶. MoveIt is a widely popular motion planning framework with a vast community currently supporting its development. It provides intuitive plugins for programming free-space robot motions and supports several libraries for path planning, kinematics solvers and time processing. Within MoveIt, the inverse kinematics solver can be selected among numerical algorithms (KDL), where the solution is reached through iterations, or analytical approaches (IKFast), in which trigonometric equations are solved to obtain the solution. The inverse kinematics solver selected was IKFast, as it is more efficient and stable than numerical approaches for simple robot manipulator without any external axis. In MoveIt, collision checking is performed in each discrete robot state, hence the motion between robot states may not be collision free. Rather than free-space motions, Descartes is primarily focused on offline planning of cartesian tasks. Although Descartes reuses some functionalities from MoveIt, it is considered a standalone motion planning framework. When Descartes was used, it was also adopted the same kinematic and collision solvers previously described for MoveIt. Finally, Tesseract is a recent lightweight motion framework designed specifically to tackle industrial applications. Tesseract adopts KDL as the kinematics solver, together with a novel approach for collision detection, allowing continuous collision checking. Being newer than MoveIt, Tesseract still faces major updates every so often.
- Path planners:** In this project, a path planner received a set of operational space vectors and computed a suitable joint space motion. There are several path planners available in ROS but only three were analyzed here: RRTConnect, Descartes and TrajOpt⁷. RRTConnect is available through the Open Motion Planning Library (OMPL), an open-source library that provides high-quality well-tested randomized planners. Every OMPL planner, and so RRTConnect, is compatible with MoveIt. Descartes path planner was developed specifically to deal with offline planning of industrial cartesian task, always obtaining the global optimal robot path. Lastly, TrajOpt is an optimization path planning algo-

⁴<https://github.com/ros-planning/moveit> (accessed 5 April 2020)

⁵<https://github.com/ros-industrial-consortium/Descartes> (accessed 5 April 2020)

⁶<https://github.com/ros-industrial-consortium/tesseract> (accessed 5 April 2020)

⁷https://github.com/ros-industrial-consortium/trajopt_ros (accessed 5 April 2020)

rithms only supported in Tesseract. While RRTConnect is a well established planner, both Descartes and TrajOpt are very recent planners and seeming more suitable for industrial cartesian applications, although both lack testing from the research community.

- **Messages:** In ROS, nodes share information through messages that are transferred either with topics, services or actions. For a complete abstraction, every ROS message follows a simple data structure, divided into headers and fields. The header identifies the message and contains the time at which the message has been created and the associated coordinate frame. The field stores the actual data to be transmitted by the message. Several types of messages were adopted in this project (some already available, others designed for this project) and their names are attributed to the data that they carry. For example, the message *PoseArray*, included in *geometry_msgs* package, was used to carry an array of end-effector positions and orientations in relation to the base of the robot. The *PoseArray* message is deconstructed in Figure 4.6.

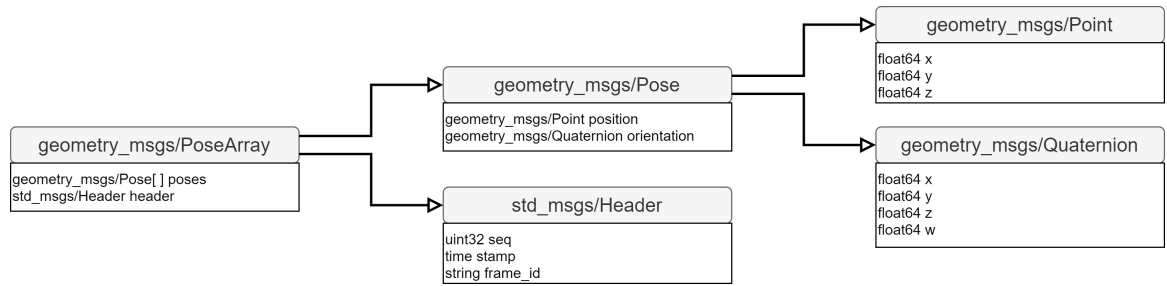


Figure 4.6: Example of a ROS message: *PoseArray*. The digits at the side of the data type int and float represent the number of bits that it can hold.

4.3. Deposition specifications

With the objective of analyzing the robot motion for laying down VSL with in-situ consolidation, the deposition process was specified by the laminate description and the laydown speed.

- **Laminate Description:** A laminate is comprised of layers, each containing multiple fiber tow courses. Each tow course was described by a sequence of cartesian waypoints, originated from the discretization of its centerline. Since the cartesian coordinates already take both the tow width and thickness into account, these do not need to be explicitly defined. In this project, the cartesian positions were provided by the outcome of the VSL optimization method proposed in [12].
- **Laydown speed:** As stated in [8], when steering fibers with in-situ consolidation, good bonds were achieved when the laydown speed was either 3 m/min (0.05 m/s) or 6 m/min (0.1 m/s) using a 3 kW diode-laser as the heat source. While seeking for the highest productivity, a laydown speed of 6 m/min (0.1 m/s) was implemented throughout this thesis.

The LayLa Framework

As a proof of concept that ROS can be adopted as a middleware for offline programming of complex composite deposition, the outcome of this thesis consisted of a framework named LayLa¹ (Laying Laminates). This chapter is dedicated to expose an overview of LayLa's capabilities and workflow.

5.1. LayLa capabilities

The LayLa framework is a compilation of open-source packages that enables automatic generation of robot trajectories for composite fiber placement. LayLa's main design driver was to allow researchers to effortlessly manufacture composites with a direct implementation of the output of their own fiber tows distribution strategies. Fiber tow courses with a complex geometry, such as found in VSL, can therefore be produced without the need to manually adjust them. Moreover, LayLa adopts transparent algorithms to generate optimal robot motions, algorithms that have been perfected over years by the robotic research community and, most importantly, are not constrained to any robot manufacturer. With an additional post-processing step, a constant laydown speed is hardly imposed, avoiding the superfluous speed variations spotted when tow courses contain curvature changes. After a successful generation of a robot trajectory, the achieved deposition stage is then simulated in a graphical interface, to be later forwarded to a real robot as an external command.

Several commercial solutions for identical purposes include components to first distribute the fiber tows in a given mold and then calculate a suitable robot motion sequence to execute them. In contrast, LayLa is exclusively focused on generating accurate robot motions, laying down the requested tow courses as closely as possible. As input, neither CAD surfaces or ply boundary information are read, but rather a set of cartesian waypoints that characterize each fiber tow centerline. Consequently, the design of the laminate must be conducted beforehand in a separate step. This approach makes it no different whether a straight or curvilinear course is being requested. At its core, LayLa is compatible with tow courses defined in a three-dimensional space. As such, laminates manufactured using complex mold geometries, for instance single or double curved surfaces, can also be executed and not only those specified on flat surfaces. However, as most studies on VSL have centered on flat molds, experiments in this thesis were only performed for the latter.

Each time LayLa is executed, the robot trajectory corresponding to a single tow course is generated. The respective layer and course to be performed can be selected via the command-line, so there is no obligation to recompile the code. The tow course to be followed can be selected between using the exact or an approximated solution. With the exact solution, the supplied cartesian coordinates are directly used to generate each robot configuration. It was observed, however, the possibility of creating jerky motions if the tow course discretization did not guarantee a smooth tangent distribution. In those cases, an approximation solution can be adopted, which basically interpolates the tow course with a quintic B-spline, generating smoother motions without great loss of accuracy. For the latter solution, tow courses must have at least six waypoints.

Besides providing the cartesian coordinates of the tow centerline waypoints, the user is also entitled to modify various parameters via the command-line. The position of the tow course in relation to the robot base may be modified to adjust how close or far the mold surface is to the robot. Other options include to select which of the compatible path planner is to be used, to change the laydown speed and also to modify

¹<https://github.com/andreflorindo/layla>

the scale of joints velocity while performing secondary transport motions. In case of simulations without the real robot, a generic hardware interface can also be selected. For the end-effector to approach the tow course with a predefined angle, an extension of the path planned may also be introduced.

Regardless of the shape and size of the tow course, Layla enforces a constant laydown speed throughout the course. It may happen that the requested laydown speed is not achievable for the entire course, as it would require one joint to surpass any of its safe limits. In those cases, the laydown speed is automatically reduced by 10% and the time parameterization step is repeated at a lower speed. The process of reducing the laydown speed is repeated until an achievable uniform laydown speed is reached or the maximum number of iterations (10, laydown speed is reduced by 65%) is exceeded. In the case of the latter, Layla reports that it failed to achieve a constant laydown speed and the program is terminated.

With ROS supporting different programming languages, the Layla's nodes were written in Python and C++ languages, following the rules of object-oriented programming. The node responsible for arranging the end-effector poses, designated Pose Builder, was developed using Python due the library SciPy delivering great interpolation functions, particularly when the knot vector is unknown. In regards to the construction the motion problem and also the time parameterization, performed by the two nodes named Path Planning and Time Parameterization respectively, C++ was used as is the standard language in which ROS libraries are developed.

5.2. LayLa workflow

In this section, the LayLa workflow sequence to convert a set of cartesian waypoints into a robot trajectory is described. To this end, the initialization procedures and the interaction between the main components of Layla are outlined, while briefly exposing the methodology and outcome of these components. In later chapters, each individual component is discussed in more detail, along with experimental data.

To initialize the framework, the user must supply a set of input data about the deposition specifications and the robot system. The laminate description has to be arranged in a *.txt* file with a specific format, exemplified in Figure 5.1. Besides the laminate description, the user must also provide the number of the layer and the number of the course where the desired tow course is located, together with the desired laydown speed. As described in the last chapter, the description of the robot system is provided through an URDF file. Additionally, if the framework is to be operated on a real robot, the hardware interface for that specific robot must also be arranged (either building one from the ground up or searching in the ROS-I repository).

```

Layer 1

Course 1

0.000000 0.388822 0.000000
0.014463 0.394463 0.000000
...

Course 2

0.000000 0.366261 0.000000
0.010316 0.370316 0.000000
...

...

```

Figure 5.1: Format of the *.txt* file containing the laminate description. The cartesian coordinates follow the order *x*, *y* and *z* and are given in meters.

A simplistic flow chart of the Layla workflow is displayed in Figure 5.2 and is outlined in this paragraph. The developed framework for composite deposition was divided into three components: Pose Builder, Path Planning and Time Parameterization. The Pose Builder block is responsible for reading the laminate description file and building the sequence of end-effector poses to accomplish the requested tow course. The Path Planning block acts as the central node of the framework. It communicates with the Pose Builder through

a ROS service, in which the requested layer and course are sent, to later receive its corresponding cartesian path. Afterwards, the motion problem is constructed using one of the compatible path planning algorithms, obtaining the set of joints positions to perform the requested tow course. These joints path are then forwarded by another ROS service for the Time Parameterization block, which introduces a time condition for each waypoint. Here, a constant end-effector speed is enforced while ensuring that velocity and acceleration of each joint is within the robot limits. The obtained six joint trajectories and their respective velocities and accelerations are subsequently sent to the Trajectory Controller, which interpolates a new joints trajectories with a quintic spline, returning the joints command. Finally, through the Hardware Interface these joints command are submitted to the real/simulated robot system, returning the actual six joint trajectories performed. The latter can then be visualized in the computer within a simulated environment.

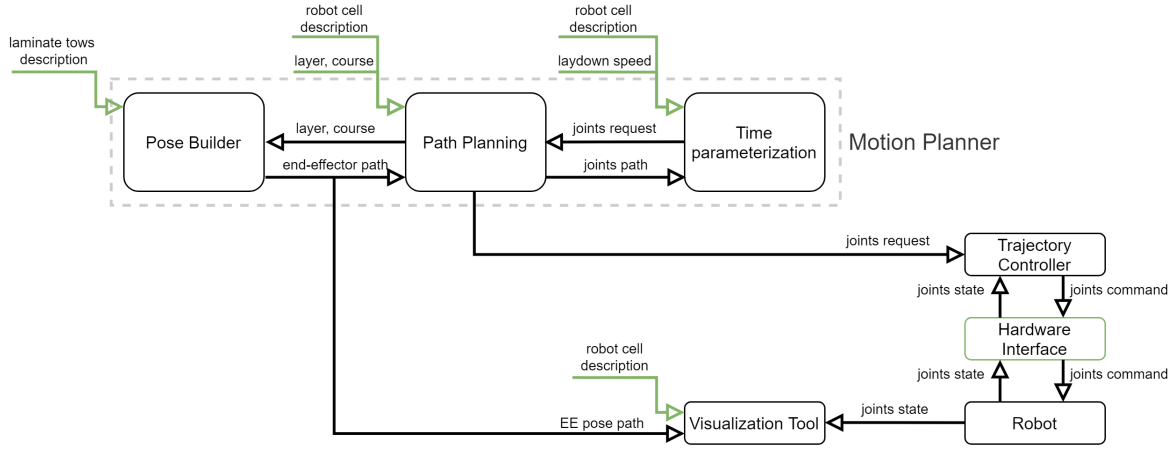


Figure 5.2: LayLa simplified workflow diagram. At green are represented elements provided by the user.

As proposed in Figure 5.2, the three components that make up the motion planner will be further described in the next subsections. Before doing so, it is relevant to introduce the meaning behind the information shared in this framework.

- **end-effector path:** Contains the position and orientation of the end-effector for all tow course's waypoints, without any information about the time. The end-effector position is provided by cartesian coordinates, while its orientation by quaternions, having a total of seven variables per waypoint. It is carried by the message *PoseArray*, from the *geometry_msgs* library, as previously shown in Figure 4.6.
- **joints path:** Contains the joints position for all tow course's waypoints, without any time condition. For a 6 degrees-of-freedom manipulator, a joints path contains six variables per waypoint, one for every joint. It is carried by the message *JointTrajectory*, from the *trajectory_msgs* library.
- **joints request:** Contains the joints positions and the corresponding velocities and accelerations to execute a tow course, after a time condition is imposed. Three variables are therefore required per joint, plus a time condition, making a total of nineteen variables per waypoint. It is carried by another message *JointTrajectory*, from the *trajectory_msgs* library.
- **joints command:** Contains the joints positions after interpolated by the ROS controller. Besides the joints positions, a time condition is also provided, making a total of seven variables per waypoint. It is carried by the message *JointTrajectoryPoint*, from the *trajectory_msgs* library.
- **joints state:** Contains the actual joints position performed by the robot every sampling time. As the joints command, it contains seven variables per waypoint, the joints position and respective time conditions. It is carried by the message *JointTrajectoryPoint*, from the *trajectory_msgs* library.

5.2.1. Pose Builder

The first task of the Pose Builder is to read the laminate description and store the cartesian coordinates of the tow centerline associated with the requested layer and course. The reading task was tailored to solely store a requested tow course each time the Python script is executed. A single laminate may contain hundred if not thousands of courses, each with dozens of waypoints. If all the tow courses provided in the laminate description file were stored in a single step, memory that could be vital for the path planning had already been consumed. To avoid this excessive memory usage, at a first stage the file is lightly read, only to find the total number of layers and the respective number of course provided within the file. If requested a layer or course that does not exist in the file, then no further reading is necessary and the node is terminated. If, on the other hand, the layer and course requested are available, only the waypoints for the desired course are stored and processed for the next stages.

With the cartesian coordinates of the tow course defined, the following task consists of computing a sequence of end-effector poses to perform the provided tow course. In short, each waypoint is placed in a three dimensional space by assigning a coordinate frame in relation to a reference frame. A smooth pose transition was achieved after fitting the tow course on a quintic B-spline and distributed the waypoints at the same distance from each other. Chapter 6 will address how this task was performed, along with experimental results of the fitting process. The achieved end-effector path is then stored in a proper message and sent to the Path Planning node.

5.2.2. Path Planning

As a central node, Path Planning receives the user input on which layer and course in the laminate description file is located the desired tow course to be laid down, sends it to the Pose Builder and waits for its response. The three compatible path planning algorithms adopt their own classes to initialize the robot model and construct the motion problem. Path planning of cartesian tasks involves performing inverse kinematics and checking the minimum displacement between consecutive configurations, while simultaneously avoiding collisions. In Chapter 7, a comparison between the three path planners will be made when applied to an industrial manipulator. After the path planning is completed, the joints path is submitted to the Time Parameterization block, which returns with the joints trajectory. The latter is then forwarded to the Trajectory Controller through the action *FollowJointTrajectoryGoal*, available at the `control_msgs` library.

For a deposition process, the robot motion was divided into three steps: reach a predefined home position, move to the first waypoint of the tow course and execute the tow course. The first two motions are commonly referred as transport motions and are not essential to the actual deposition, therefore were solved using the simplest path planner RRTConnect, ideal for such free-space motion. In these transport motions, a 5% scaling factor as applied for the joints limits. Being the spotlight on the execution of the actual tow course, no further analysis was conducted on those transport motions.

5.2.3. Time Parameterization

In the Time Parameterization block, a time condition is inserted for every waypoint of the joints path. How the time is calculated varies from method to method, with Chapter 8 containing an analysis of the time parameterization methods currently available in MoveIt and the solution developed to maintain a constant end-effector speed. The general idea behind these methods is to calculate the shortest time it takes the robot to reach the next configuration. This shortest time is dictated by the joint that takes the longest to change to the next position when limits are imposed on its velocity or acceleration. These limits can be either the robot limits, a proportional fraction of the latter or dependent on the end-effector speed. LayLa will display if the requested laydown speed can be reached, or if not, the highest laydown speed that can be utilized. The deposition process is only continued if the user accepts the obtained laydown speed. After a time condition is imposed for every waypoint, the velocity and acceleration of each joint are calculated and stored in the joints trajectory, which returns to the Path Planning.

IV

Data Analysis

6

Tow Course Fitting

In this chapter, the methodology behind the Pose Builder is presented, indicating how the end-effector poses were generated from a set of cartesian waypoints. In the interest of achieving an overall smooth robot motion, with the latter being an indicator of the process quality, the strategy adopted to eliminate any superfluous motion that may arise is equally described herein.

From a flat laminate sample, two courses were selected to undertake the experiments. The first tow course, depicted in Figure 6.1, is one of the simplest VSL path shapes and is characterized by an unique concave arc. The second tow course, represented in Figure 6.2, is a much more elaborate VSL path shape, made of a concave arc followed by a convex arc, both separated by an inflection point. Due to the higher complexity of the Tow Course 2, throughout this chapter the analysis of the Pose Builder activities is predominantly established for the latter. Nevertheless, the cartesian path obtained for both Tow Course 1 and 2 is illustrated at the end of this chapter.

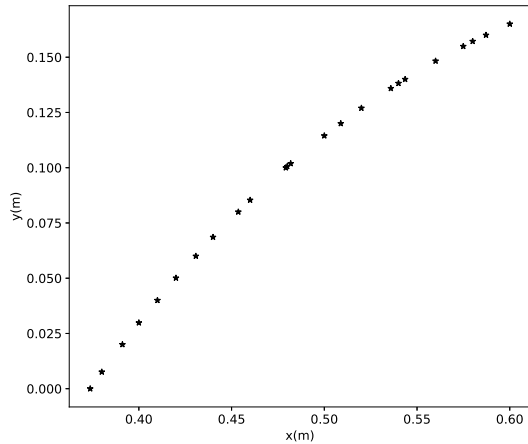


Figure 6.1: Tow Course 1, made of 24 waypoints, a length of 0.2833 m and minimum turning radius of 0.473 m.

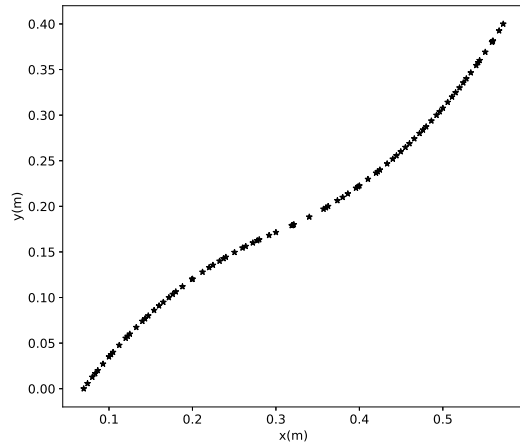


Figure 6.2: Tow Course 2, made of 88 waypoints, a length of 0.6522 m and minimum turning radius of 0.420 m.

6.1. Waypoint frame assignment for a space curve

Once the sequence of cartesian coordinates portraying the tow course is obtained, the next task of the Pose Builder is to describe that same tow course in a three dimensional (3D) space modeled after the real robot cell. This is accomplished by assigning a fixed coordinate frame at each waypoint with respect to a reference frame. Where this reference frame is located can be arbitrary, though it is a standard practice to use the frame at the robot cell origin, also known as world frame. The relation between a waypoint frame and the world frame is established with a homogeneous transformation matrix given by the expression

$$T_i = \begin{bmatrix} R_i & p_i \\ \mathbf{0}_{1 \times 3} & 1 \end{bmatrix}, \quad (6.1)$$

where R_i and p_i represent the rotation matrix and the position vector of the waypoint i in relation to the world frame, respectively.

Contrary to the world and waypoints frames, the coordinate frame allocated to the end-effector does not remain fixed and its motion is dictated by the task to be conducted. For tasks where the end-effector has a fixed orientation with respect to the motion direction, as is the case with composite deposition, the end-effector frame coincides with the corresponding waypoint frame as the motion progresses. Thereby, the end-effector position at the i th waypoint is specified by the vector p_i and its orientation is provided by the rotation matrix R_i . From p_i and R_i , the cartesian coordinates and quaternions that describe the end-effector position and orientation can be computed, respectively.

By having the cartesian coordinates of the tow course waypoints, assembling the vector p_i for the position of the end-effector is straightforward. In order to bring the tow course up to the table, a known distance from the cell origin to the top of the table was added to the waypoints coordinates, thus obtaining p_i . Finding the end-effector orientation along the tow course, that is, constructing the rotation matrix R_i , is where the complexity of the problem lies.

A conventional approach to specify the orientation along a curve in a 3D space involves assigning the so-called Frenet frames. A Frenet frame is a mutually orthogonal frame featuring three unit vectors denominated tangent, normal and binormal. The tangent vector provides the direction in which the curve is heading. The normal vector follows the direction in which the curve is turning, or in other words, captures how the tangent evolves along the curve. Lastly, the binormal vector indicates how the curve is twisting, being orthogonal to the plane created by the tangent and normal vectors. Notice that the normal vector herein introduced is not pointing in the direction of the surface normal, a role handled instead by the binormal vector. An example of a Frenet frame is depicted in Figure 6.3.

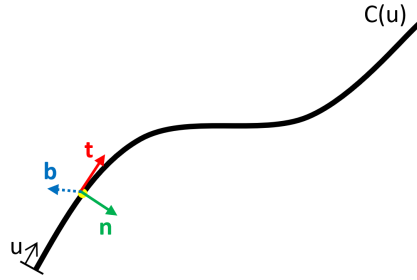


Figure 6.3: Representation of a Frenet frame at a curve $C(u)$, where u is the parameter to which the curve is parameterized. t , n and b represent the unit tangent, normal and binormal vectors, respectively.

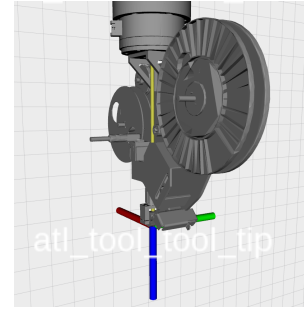


Figure 6.4: Convention adopted for the TCP coordinate frame.

Assuming the convention for the Tool Center Point (TCP) frame proposed in Figure 6.4, the rotation matrix R_i described with Frenet frames is given by

$$R_i = [-t_i \quad -n_i \quad -b_i], \quad (6.2)$$

where t_i , n_i and b_i represent the unit tangent, normal and binormal vectors at the waypoint i , respectively. According to the definition of a Frenet frame, the three orthonormal vectors are obtained by analytically evaluating the curve function. The tangent results from the derivation of the curve with respect to the parameter which is parameterized. The normal is calculated with the second derivative of the curve or the derivative of the tangent in relation to the same parameter. The binormal is then orthogonal to both previous vectors and is given by their cross product. The following equations reflect what was previously described.

$$t_i = \frac{\frac{dC(u_i)}{du}}{\left\| \frac{dC(u_i)}{du} \right\|}. \quad (6.3)$$

$$n_i = \frac{\frac{d^2C(u_i)}{du^2}}{\left\| \frac{d^2C(u_i)}{du^2} \right\|}. \quad (6.4)$$

$$b_i = t_i \times n_i. \quad (6.5)$$

When the previous expressions were employed for the Tow Course 2, the shortcomings of the Frenet frames became apparent. Being the second derivative zero at inflection points, the normal vector, and consequently the binormal vector, could not be directly defined at these locations. Moreover, as inflection points separate regions in the curve with opposite curvature directions, both the normal and binormal vectors had their direction inverted after passing an inflection point, as visible in Figure 6.5. In practice, this was evidenced as the robot tried to rotate the end-effector 180° around the unit tangent, which is naturally unreachable due to collision with the table. One solution to avoid this rough twisting could be to impose a same sign in the binormal components, while ensuring that no inflection points are used to describe the tow course. However, this solution would only be suitable for plane curves, where the binormal is kept constant.

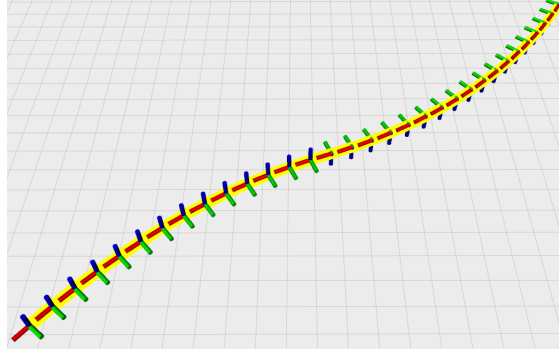


Figure 6.5: Using the Frenet frames method to compute the orientation along the Tow Course 2. As in Figure 6.3, the unit tangent, normal and binormal are represented by the colors red, green and blue, respectively.

To address the issues caused by the dependency on the second derivative, an alternative method for assigning frames was proposed in [82] based on the concept of parallel transport. By feeding the unit binormal at the first waypoint, this method computes the next binormal vectors by inducing the same amount of rotation as evident from the prior to the current tangent vectors. Hereby, frames are allocated with a minimal rotation, avoiding the large twists frequents with analytical approaches. With this method, the unit tangents are calculated using again the equation 6.3, yet by assuming an initial binormal, the unit binormal and normal are computed using the next set of equations. In those, the expression $R(e, \theta)$ represents the rotation by the angle θ around the vector e .

$$\mathbf{b}_{i+1} = \begin{cases} \mathbf{b}_i & \text{if } \mathbf{t}_i \times \mathbf{t}_{i+1} = 0, \\ R(\mathbf{e}, \theta) \cdot \mathbf{b}_i & \text{otherwise, with } \mathbf{e} = \mathbf{t}_i \times \mathbf{t}_{i+1} \text{ and } \theta = \arccos(\mathbf{t}_i \cdot \mathbf{t}_{i+1}). \end{cases} \quad (6.6)$$

$$\mathbf{n}_i = \mathbf{t}_i \times \mathbf{b}_i. \quad (6.7)$$

The outcome of the parallel transport method is presented in Figure 6.6, effectively resulting in a more coherent frame assignment. In a similar approach, an initial unit normal could be provided by simply reformulating the equations such that the binormal is now given by the cross product. For composite deposition, the choice fell on selecting an initial binormal since these vectors convey the geometry of the mold adopted, which in this case was a flat, horizontal table. Note that if any other surface is employed, whether surfaces at a vertical position or even curved molds, this initial binormal has to be manually modified to an appropriate vector.

For a discrete set of waypoints, the unit tangent vector can be calculated using the linear distance between two consecutive coordinates and divided by the resulting norm. However, this method was proved to be very inconsistent. For starters, one waypoint was left without a tangent, which implied making an assumption either at the start or the end of the course. But above all, it depended on the user's ability to provide a set of waypoints with a smooth tangent distribution. As the robot follows the tow course, even small perturbations at the unit tangent may result in a highly jerky robot behavior. These perturbations were evident when this method was applied for the Tow Course 2. By having the discrete tangent, the binormal and normal were calculated using the parallel transport method previously described. Being a plane curve, the binormal remained constant along the curve. Furthermore, laying down in a horizontal surface, the tangent and normal vectors only had components in the x and y -axis. Through using this approach, the unit tangent and normal components along the x -axis obtained for the Tow Course 2 are shown in Figure 6.7.

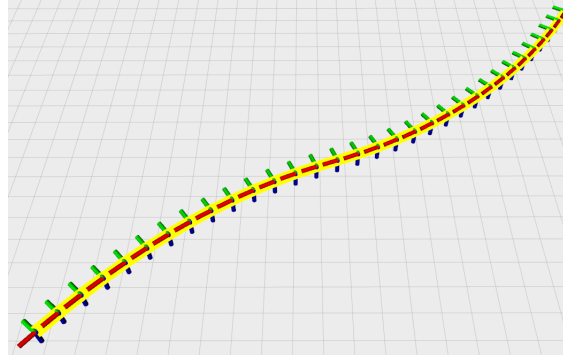


Figure 6.6: Using the Parallel Transport method to compute the orientation along the Tow Course 2. As in Figure 6.3, the unit tangent, normal and binormal are represented by the colors red, green and blue, respectively.

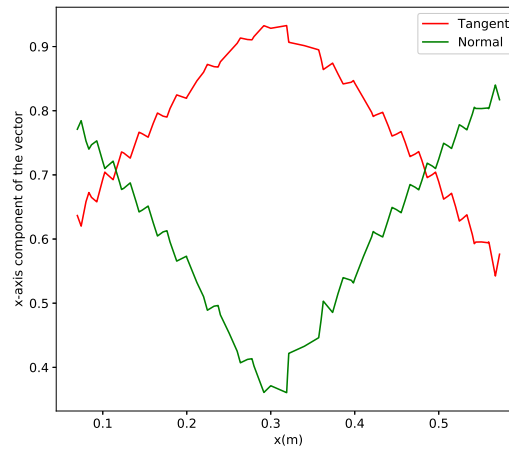


Figure 6.7: Unit tangent and normal x -axis components in function of the x coordinate if adopted the waypoints provided in the laminate description file, for Tow Course 2.

In order to understand what would happen to the end-effector, keep in mind that when the tangent x -axis component increases, the end-effector rotates clockwise, while when it decreases, the end-effector rotates counterclockwise. From Figure 6.7, the obtained unit tangent and the normal displayed a sequence of relative maxima followed by relative minima. In a real robotic system, these perturbations would cause the end-effector to be repeatedly rotated clockwise and then counterclockwise over a very short distance. As a result, the joint A6, responsible for rotating the end-effector about its symmetry axis, would be operated by jumping from its maximum to minimum acceleration limits, handicapping the other joints motions from operating at higher velocities. This kind of oscillation behavior is certainly not ideal and a different approach to find the end-effector path had to be developed.

6.2. Smoother curve approximation

The solution employed to avoid such oscillation behavior relied on discarding the waypoints provided and approximating the original curve to a rather smoother one. The interpolation of tow courses in a 3D space was conducted with B-splines and the new waypoints were selected to be equally spaced. However, as with many parametric curves, a B-spline is constructed using control points instead of points of the curve. The latter could be accomplished by adding multiple interior knots for each waypoint, yet this ultimately created kinks at the tangent, ultimately leading to a solution identical to Figure 6.7.

Within the interpolate module of the SciPy library, there is a function called *splprep* that calculates a B-spline representation of a set of waypoints, which is then used as an input for another function called *splev* that constructs the actual B-spline. To eliminate superfluous changes in the tangent vector, in *splprep* it is possible to introduce a smoothing condition s to the curve representation. As explained in the manual of the function, if a smoothing condition is applied, the algorithm rearranges the interpolated curve such that the sum of the

squares of the discrepancies between a new and the original curves is lower or equal to the value carried by this smoothing condition. With s kept at zero, a curve that passes through all the waypoints is constructed using multiple interior knots. With a higher s , a smoother curve is built at the expense of becoming less accurate to the original curve. A trade-off between closeness and smoothness was therefore necessary to be evaluated. In Figure 6.8 the tangent and normal x components as a function of s are presented, while in Figure 6.10 their corresponding absolute position error to the original path are shown. For discrepancies of a few millimeters, the square of the discrepancies has a magnitude of 10^{-6} , hence the values of s tested ranged from 1×10^{-7} to 12×10^{-7} , starting from smaller discrepancies to later reach the magnitude of millimeters.

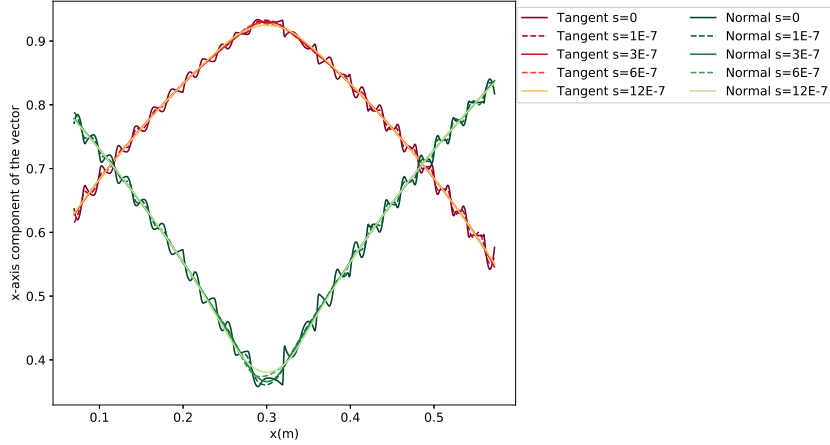


Figure 6.8: Unit tangent and normal x -axis components in function of the x coordinate for curves with different smoothing condition, for Tow Course 2.

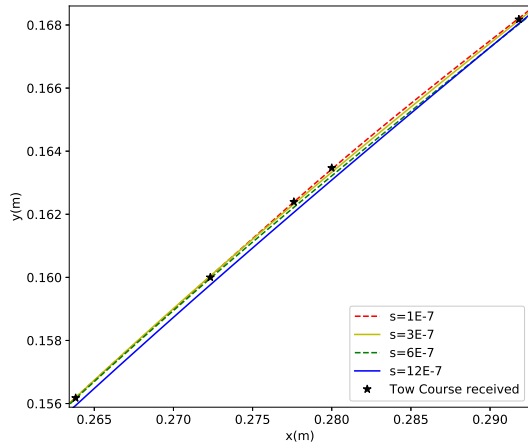


Figure 6.9: Close up of the region prior to the inflection point at the Tow Course 2, using different smoothing conditions.

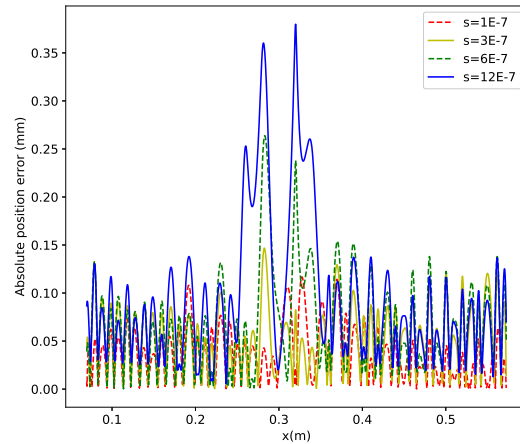


Figure 6.10: Discrepancy induced when approximating the Tow Course 2 by cubic B-splines with different smoothing conditions.

As visualized in Figure 6.8, the oscillation phenomenon found in the unit vectors decreased significantly even for the lowest values of s , disappearing completely when s reached 3×10^{-7} . For s equal or higher than 3×10^{-7} , the distribution of the tangent and normal components along the curve acquired a shape similar to a parabola. When the s parameter was raised to 12×10^{-7} , the parabola shape became wider and its absolute maxima was decreased. Therefore, after 3×10^{-7} an increase in s enlarged the region containing the inflection point, becoming closer to a linear curve. This result is corroborated by Figure 6.10, as discrepancies between an area around the inflection point intensified for larger s parameters. In comparison, both the start and end of the curve, regions that are outside the influence of the inflection point, remained practically unchanged for larger s .

The absolute position error curves exhibited a fluctuating pattern. In Figure 6.9 a close up of the approximated curves is presented to help explain such a pattern. Focusing on the path where s is 3×10^{-7} , the first two waypoints were on top of the provided curve, diverging below for the next two waypoints and terminating again close to the last waypoint. Thus, the linear interpolation of the exact path received seems to be intertwined around the approximated curves, leading to this oscillation in the discrepancies. Nevertheless, with this approach, a position error is inherently present when the path is send to the path planner. To avoid large displacements, an iterative step was developed that selects the lowest smoothing parameter from which the number of extremum points at the tangent was significantly reduced to the minimum. In case of the Tow Course 2 presented in the above plots, the value s automatically selected was 3×10^{-7} .

6.3. The degree of the curve

Another aspect that deserved attention was the degree of the interpolated tow course. With inverse kinematics, the joints position vectors are calculated by solving a series of trigonometric and inverse trigonometric functions. Since these functions are infinitely differentiable, the degree of continuity in a joint position curve $J(u)$ is dependent on the cartesian curve $C(u)$ provided, on the existence of collision object and how successful the path planner is in selecting the optimal solution. Without collision objects actively blocking the course and assuming that the path planner generates optimal paths for all joints, the generated joints position curves will keep the degree of continuity for which the cartesian curve was mapped. Additionally, when using equidistant waypoints combined with a constant end-effector speed, while staying below the robot limits, the parameter u (for example the arc length) of the curve will be proportional to time. Hence, not only the joints position curves retain the degree of continuity in function of the path parameter, but also maintains it in function of time. As such, continuity in the joints velocity and other higher time derivatives curves can be imposed by fitting the cartesian path with a high order curve.

When a smoothing condition is applied, non of the interior knots are multiplied. Without those, a B-spline behaves identically to a typical piecewise polynomial, in which a B-spline of degree n guarantees the continuity of its $n - 1$ derivatives. In the prior section, the tow courses were fitted using a smoothed cubic B-spline, thus ensuring continuity up to its second derivative. Ideally, at least jerk continuity should be preserved in order to reduce possible vibrations at the end-effector. For this reason, two curves with a higher degree were also tested, a smooth quintic B-spline and a Bezier curve. The Bezier curve can be seen as a B-spline with a single segment, in which its degree is given by the number of waypoints minus one and its knot vector filled with a first half of zeros and a second half of ones (meaning that it is clamped at both ends and uses the remaining waypoints as standard control points). In Figure 6.11 the three course can be seen, with their discrepancies and unit vectors depicted in Figure 6.3 and 6.13, respectively.

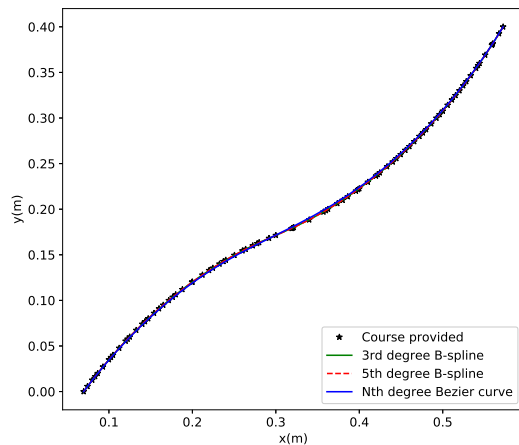


Figure 6.11: Tow Course 2 interpolated using curves with different degrees

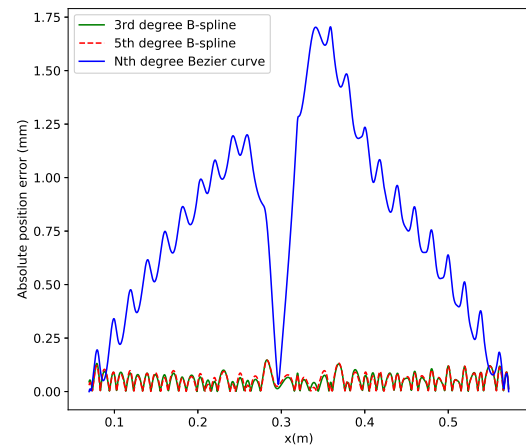


Figure 6.12: Discrepancy induced when approximating the Tow Course 2 using curves with different degrees.

From Figure , it can be seen that the position error between cubic and quintic B-splines were mostly identical, both with a maximum disparity of 0.146 mm. In contrast, the Bezier curve revealed a much higher discrepancies along the path, reaching almost 1.705 mm after the inflection point, that being 11.7 times higher

than that for the two previous curves. Still, it should be noted that, even for the Bezier curve, the inflection point is closely crossed. As expected, the lower degree curves more closely resembled the received tow course. With respect to the unit vectors, from Figure , again no major noticeable differences were detected between cubic and quintic B-splines. The Bezier curve behaves in accordance with increasing the smoothing condition, obtaining a flatter distribution of the unit vectors. Moreover, since for the Bezier curve both ends were clamped, a small inconsistency in the unit vectors was evident in these regions.

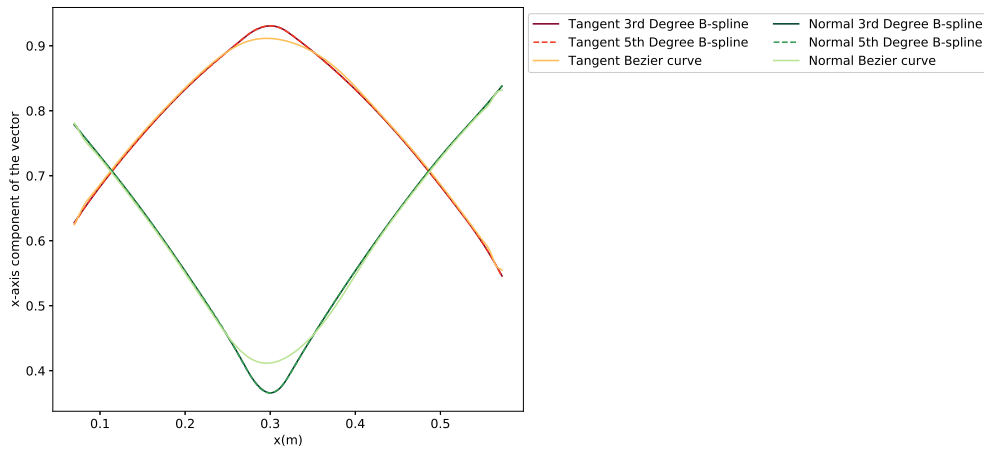


Figure 6.13: Unit tangent and normal x -axis components in function of the x coordinate for curves with different polynomial degrees, for Tow Course 2.

To check on the robot trajectory, the three distinct interpolation curves were finely discretized and executed in LayLa, while using Descartes Dense as the path planner and imposing a constant laydown speed of 0.1 m/s. Figure 6.14 shows the acceleration requested at the joint A6.

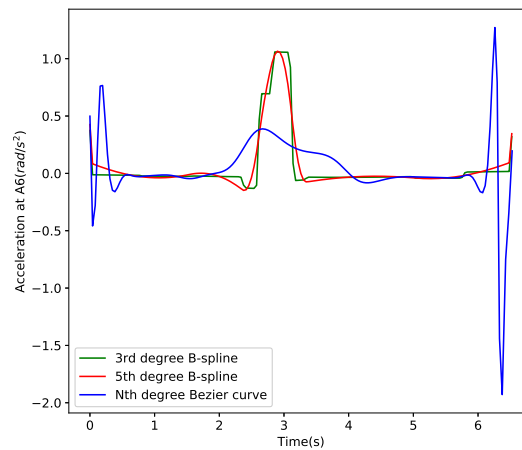


Figure 6.14: Acceleration requested at the joint A6 when approximating the Tow Course 2 by curves with different polynomial degrees.

By visualizing the acceleration at the joint A6, the differences between using a degree of three or five became more noticeable, even though the cartesian curves and unit vectors seemed indistinguishable. For the time being, it is asked to focus on the center of the plot, comprising the region where the inflection point is contained. In this region, the unit tangent undergoes major changes in direction while passing through its peak, as evident in Figure 6.13. Those modifications in the orientation were solely executed by the joint A6. For the cubic B-spline, the acceleration clearly proceeded linearly, while the quintic B-spline resembled a parabola shape. As for the Bezier spline, the changes in the unit vectors occurred much sooner and were less accentuated, demanding lower accelerations for a longer period. Moreover, the rather complex shape of the acceleration curve does imply a large degree of continuity.

Figure 6.14 gives strong indications that the joint position does in fact retain the degree of continuity of the interpolated cartesian curve. However, as the degree of the curve is increased, so are the position errors. A quintic B-spline proved to be a good compromise between aiming for low position discrepancies while still producing jerk continuous motions. For the remaining activities, it was decided to interpolate the set of waypoints using a quintic B-spline.

Bear in mind that by discretizing the interpolated curves, the velocity and acceleration profiles will depend on the number of waypoints used. Additionally, the joints position greatly depend on how time parameterization is performed. The last paragraph is only applicable when all six joint velocities and accelerations are below the limits imposed by the robot. The latter is usually the case for industrial cartesian task, such as welding or composite deposition, where velocities requested are far from the robot maximum performance. For example, in the case presented, if the laydown speed cannot be reached due to exceeding the acceleration limits at the joint A6, the time is directly imposed by those limits and therefore loses the continuity properties. This is in part what happened at the end of the Bezier curve, where the limit of 2 rad/s^2 is hard imposed due to the inconsistent tangent at the end, suppressing the advantages of a soft curve.

6.4. Number of interpolated waypoints

With an approximated tow centerline defined, the last step to be established was how to discretize the new curve. An ideal scenario would be to enforce a waypoint at every sampling time of the robot until the course is completed. In most instances, particularly for long cartesian paths, the high number of waypoints necessary would quickly overload the path planning process, ultimately making such desirable approach computational unaffordable. To prevent such issue, the conventional approach in ROS consists of sending the path planner's output to a position controller and only then is discretized to a more refined frequency. As a consequence, the number of waypoints actually submitted to the path planning step can be reduced. Such reduction is dependent on the shape of the tow course, yet due to the manufacturing constraints imposed while designing VSL paths, simple shapes are expected and the number of waypoints may be drastically reduced.

By imposing a constant end-effector speed in conjunction with equidistant waypoints, the time between each pair of waypoints also remains constant. In LayLa, the distance between waypoints and the end-effector speed can be altered using the command-line. If the end-effector speed is kept unchanged while the distance between waypoints is modified, the changes in time between waypoints can be viewed as a rate at which a new end-effector pose is published. The higher the rate, the shorter the distance between waypoints, requiring more waypoints to describe the curve. An analogy with the rate of the robot can therefore be performed. In the Table 6.1, the number of waypoints used and the distance between them is given for a frequency of just 1 Hz to even the robot rate at 250 Hz (ideal scenario), assuming a constant end-effector speed of 0.1 m/s. The position errors originated from the interpolated curves using different frequencies are presented in Figure 6.15.

Table 6.1: Discretization elements for the fitted Tow Course 2, assuming a constant end-effector speed of 0.1 m/s.

Rate (Hz)	Distance between two waypoints (mm)	Number of waypoints
1	100	6
2	50	13
5	20	32
25	4	163
50	2	326
125	0.8	815
250	0.4	1630

From the Figure 6.15, discretizing the tow curve at a rate of 1 Hz, hence using only 6 waypoints, proved to be insufficient to achieved reduced discrepancies. However, by just doubling the frequency, a significant reduction of the position error was achieved. Starting from 5 Hz, the position error practically became constant, with increasing the number of waypoints having little effect on the curve's accuracy. Therefore, the curve was well defined by just using the lowest frequency of 5 Hz. With this approach, a distance between waypoints of 20 mm used 50 times less waypoints that those required if they were published at the same rate as the robot.

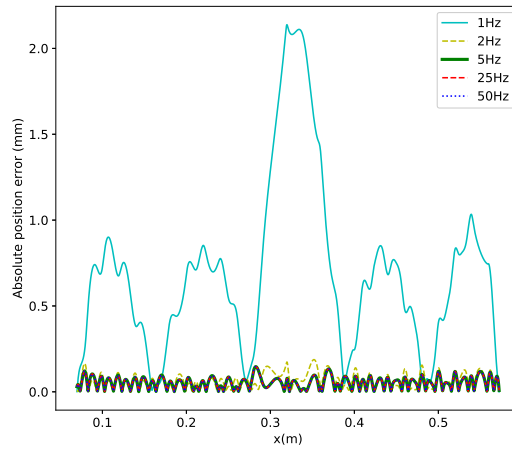


Figure 6.15: Discrepancy induced by approximating the Tow Course 2 with difference number of waypoints.

To summarize, creating an end-effector path with the exact waypoints received led to a non-smooth tangent distribution, which in return produced jerky robot motions. To prevent this robot behavior, Pose Builder node fits the received tow centerline waypoints into a smoothed quintic B-spline, with the new waypoints separated by 20 mm, which can be seen as sending a waypoint at every 5 Hz, assuming a laydown speed of 0.1 m/s. In the next figures the obtained approximated discretization of the fitted tow course and the respective absolute position error inherent with the approximation for both Tow Course 1 and Tow Course 2 are displayed. In both cases, changes in path length and minimum turning radius were insignificant. The estimated discrepancies were in the order of 0.1 mm, a small amount that is not expected to greatly affect the properties of the designed laminate.

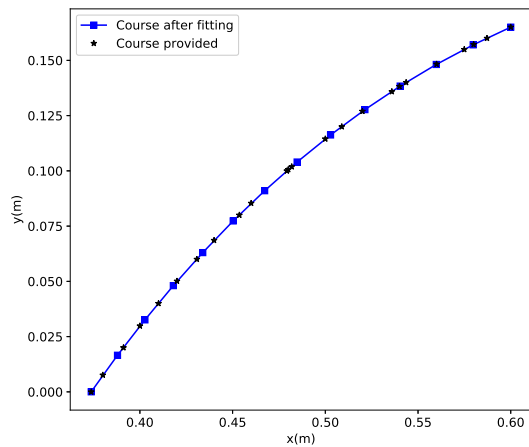


Figure 6.16: Tow Course 1 prior and after fitted with a smoother quintic B-spline.

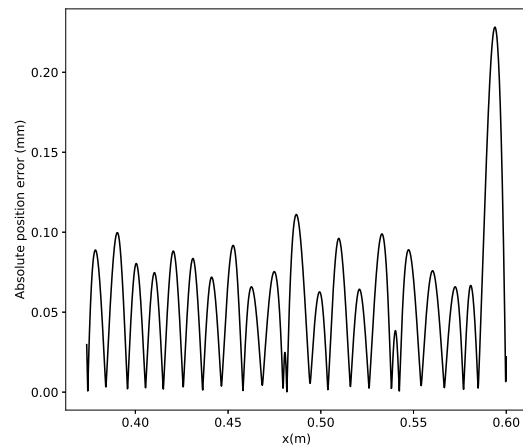


Figure 6.17: Discrepancy induced by approximating the Tow Course 1 with a smoother quintic B-spline.

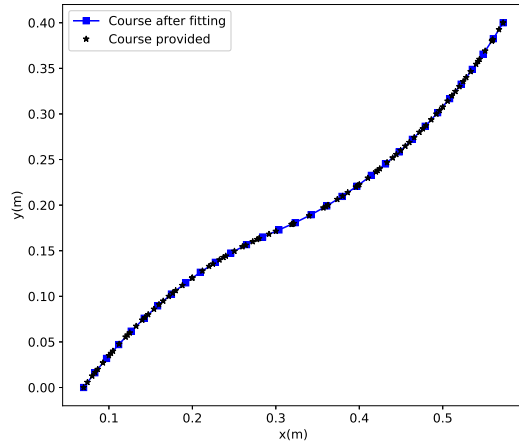


Figure 6.18: Tow Course 2 prior and after fitted with a smoother quintic B-spline.

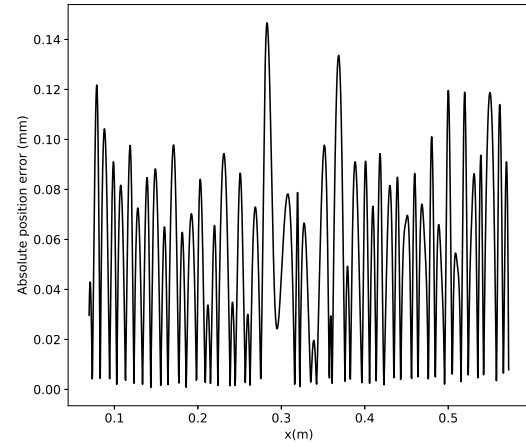


Figure 6.19: Discrepancy induced by approximating the Tow course 2 with a smoother quintic B-spline.

Path Planning Benchmark

With a tow course fitted into a smooth B-spline curve and discretized with equidistant points, the next step consisted of specifying the robot's joints path so that its end-effector would emulate that very same course. This task was achieved by one of the several state-of-the-art path planning algorithms available within the ROS framework. While some planners are more suitable for free-space tasks, where freedom of movement between a start and goal poses is allowed, others were developed to accurately perform semi-constraints or fully-constraints tasks. Composite deposition with AFP falls into the latter category and selecting the ideal planner requires first to benchmark the most probable picks for path planners.

In this thesis, three path planners were selected for scrutiny: RRTConnect, Descartes and TrajOpt. A brief theoretical background on these algorithm was provided in Chapter 2. This chapter will describe how the motion problem is set up for each of these path planners, followed by a comparison between their performance when requested the fitted Tow Course 2. The resultant joints path obtained for the fitted Tow Course 2 will be presented at the end of this chapter.

7.1. Motion problem construction

At the time of writing, setting up the three path planners analyzed was a laborious challenge as they were compatible with different motion planning frameworks, each one with their own extensive libraries. RRTConnect was performed with MoveIt, while TrajOpt was conducted with Tesseract and Descartes was integrated into a framework of the same name. Tesseract developers are currently making efforts to integrate all these three planners into their framework, but these were not made available during this project.

Besides receiving the end-effector pose as input, each path planner adopts additional parameters to set up the motion problem. One common parameter to all is the tolerance. By adding tolerances, whether in the operational or joint spaces, a wider range of robot configurations is available, generally enabling smoother but less accurate paths. With AFP being a fully-constrained task requiring high accuracy, no tolerances were added so that the end-effector strictly follows the position and orientation allocated for each waypoint frame.

Of all the motion planning frameworks adopted in this project, MoveIt is the longest in development and, as such, it is well equipped with classes to simplify programming a robot motion, in addition to publishing extensive guides on how to use them. With RRTConnect, or any other planner within OMPL, the simplest approach to construct a motion problem is by creating a *MoveGroupInterface* object, which automatically initializes everything that is required to solve the problem. For a desired deposition task, the optimal robot motion is then calculated with the function *computeCartesianPath*. This function simply calls the inverse kinematics solver to find valid robot configurations for each end-effector pose and later performs a graph search using the selected path planner. In RRTConnect, only a single parameter needs to be defined, named *range*, which represents the maximum length of a joint path segment to be added to the graph tree. MoveIt automatically calculates this parameter based on the boundaries of the joint space, therefore no tuning is required. In these random planners, the maximum time to plan a motion is also required to specify, which was selected to be at 10 s, but it will be shown that RRTConnect was able to solve the motion problem in a small fraction of that time. By combining *MoveGroupInterface* and *computeCartesianPath*, it is easy to configure the motion problem with MoveIt, but it lacks in providing choices on how to specify the motion problem.

With Descartes, the motion problem is constructed by initializing three objects, one for the robot model, another for the selected planner and a last one for the path information. The robot model contains the forward and inverse kinematics solvers and provides additional validity checks to ensure compliance with the robot range and that no collision occurs. The planner can be chosen between Dense or Sparse planners. With the Descartes Dense planner, inverse kinematics is performed for all path waypoints, which are then used to assemble a graph tree containing all possible robot configurations for the complete course. With the Descartes Sparse planner, inverse kinematics is initially performed for a tenth of the path waypoints and the remaining ones are obtained by linear interpolation in the joint space. An additional check confirms that the joints path accuracy is within a given tolerance, otherwise the Descartes Sparse outcome is continuously re-planned with more waypoints derived from inverse kinematics and less interpolated, until the predefined tolerance is achieved. As such, the Descartes Sparse planner inherently adds tolerances, which will be shown to affect the path accuracy. The path information encapsulates the logic behind the search for the optimal robot path and can be defined either in joint space or operating space, with or without axial symmetry. As symmetry is not applicable for composite deposition, a cartesian path without axial symmetry was selected.

Compared to the two previous planners, TrajOpt offers a greater flexibility in describing the motion problem. After initializing the robot model, the motion problem is detailed through a *ProblemConstructionInfo* object, dividing the problem description into three elements: *BasicInfo*, which contains the general information of the optimization algorithm; *InitInfo*, holding information about the initial condition of the optimization solver; and *TermInfo*, containing the constraints members added into the optimization. In *BasicInfo*, the optimization algorithm can be selected among a list of compatible algorithms for either free or commercial use. In this project, an optimization algorithm (named OSQP) that came freely bundled with the TrajOpt distribution was adopted. Commercial algorithms require licenses that were not available at the time. TrajOpt optimization performance depends on an initial seed path, defined in the joint space and specified in *InitInfo*. In a first attempt, a naive initial path was adopted, which consisted of a path with the same number of waypoints as the tow course, all equal to a same predefined robot configuration. In a second approach, a high-quality seed path was used, provided by the outcome of the Descartes Sparse planner. In the latter, TrajOpt is applied rather to smooth and shorten paths generated by some other sampling path planners while satisfying additional equality and inequality constraints. Equality constraints are conditions that must be satisfied by the optimizer, while inequality constraints are desired conditions to be minimized. As a sequential convex optimization algorithm, TrajOpt adopts a penalty method to turn infeasible constraints into penalties and eventually driving all constraint violations to zero. These equality and inequality constraints are defined in *TermInfo* and can be selected from a variety of variables, along with their lower and upper boundaries and initial penalty coefficient. For this project, inequality constraints were induced to collision avoidance and joints velocity, acceleration and jerk, while equality constraints were applied to the end-effector position and orientation. Higher initial penalty coefficients were placed at the end-effector position and orientation and joints jerk in order to obtain a solution that combines both accuracy and smoothness.

7.2. Benchmark

From the three path planners, a total of five trials were conducted in the respective order: RRTConnect, Descartes Sparse, Descartes Dense, TrajOpt (provided with a naive initial path) and Descartes Sparse + TrajOpt. Assessing the performance of a path planner for a specific task typically incorporates metrics such as the percentage of waypoints in collision, the runtime, the memory consumed and the task accuracy [56, 60, 61]. For safety reasons, the trials were conducted with the tow course one meter above the table. Consequently, for the task at hand, collision with the table was no longer a major occurrence to be analyzed. As a result, this section focuses on comparing the runtime, memory consumed and task accuracy when different path planners were executed for the fitted Tow Course 2 divided with rates of 5 Hz, 25 Hz and 50 Hz. As mentioned in the last chapter, dividing the fitted Tow Course 2 with a rate of 5 Hz, 25 Hz and 50 Hz is the equivalent saying the course was divided with 32, 163 and 326 equidistant waypoints, respectively. The measured runtime and memory using different planners are presented in Table 7.1 and their end-effector position errors given in Table 7.2. Since RRTConnect bases its search on randomness, the values presented in these two tables were the average of requesting the same tow course three times. Descartes and TrajOpt planners are deterministic, hence a single run was executed on both.

From Table 7.1, RRTConnect proved to be the fastest planner, with both Descartes planners not taking much longer, being for the most part lower than 0.1 s. An unexpected outcome was Descartes Dense being slightly faster than Descartes Sparse, even without the latter having performed any re-planning. This outcome

Table 7.1: Runtime and Memory consumed while planning the fitted Tow Course 2 with different algorithms.

Path Planner	Runtime (s)			Memory (MB)		
	5Hz	25Hz	50Hz	5Hz	25Hz	50Hz
RRTConnect	0.026	0.052	0.074	43.245	45.658	47.690
Descartes Sparse	0.035	0.063	0.105	53.741	54.460	56.295
Descartes Dense	0.031	0.054	0.087	52.453	54.149	56.037
TrajOpt	0.360	4.511	11.334	309.022	1272.384	2469.606
Descartes Sparse + TrajOpt	0.084	0.724	2.457	362.661	1325.770	2524.827

may be derived by virtue of the inverse kinematic solver IKFast being used with these planners, possibly making the direct inverse kinematics calculations actually faster than the cycle to interpolate the linear segment between two waypoints. When a naive initial seed path is applied to TrajOpt, runtimes between 10 and 100 times longer than the previous sampling planners were evident, being these differences larger when more waypoints were used. However, when provided an initial path from the Descartes Sparse planner, a reduction in time of about 80% compared with a naive path was evident. Nevertheless, it was still about 3 to 30 times higher than the previous sampling planners. One possible reason for these larger times could be that TrajOpt adopted the numerical inverse kinematics solver (KDL) instead of an analytical solver (IKFast) like the previous planners. Yet, without providing tolerances, which leads to more possible robot configurations, it is unlikely that this increase in time was simply caused by using a different inverse kinematics solver, but mostly owing to the optimization. Recent efforts made the IKFast compatible with TrajOpt, yet at the moment few explanation were given on how to initialize it.

The memory figures presented in Table 7.1 are for the whole Path Planning node, hence containing not just the memory consumed to compute the optimal path but also used to store the problem construction information and other small side tasks (for example, communication with other nodes). However, these side tasks are identical for all planners, so comparisons can be made regardlessly. With this in mind, in RRTConnect and both Descartes planners, an increase in the number of waypoints was followed by a slight increase in memory usage, between 40 to 60 MB with a single tow course. With TrajOpt, it was observed that the memory consumed increased linearly proportional to the number of waypoints. The same effect was evident when an initial path from Descartes Sparse was provided to TrajOpt. When the tow course was divided with a rate of 5 Hz, TrajOpt used approximately 6 times more memory than the previous sampling planners. Clearly TrajOpt computation performance was worsened by adding more waypoints into the problem, consuming more than 2 GB when the Tow Course 2 was divided with a rate of 50 Hz.

Table 7.2: End-effector position discrepancies obtained while planning the fitted Tow Course 2 with different algorithms.

Path Planner	Max. position error (μm)			Av. position error (μm)		
	5Hz	25Hz	50Hz	5Hz	25Hz	50Hz
RRTConnect	0.029	0.029	0.029	0.010	0.010	0.010
Descartes Sparse	95.979	111.06	129.983	21.186	29.410	56.248
Descartes Dense	0.029	0.029	0.029	0.010	0.010	0.010
TrajOpt	2.211	0.024	0.024	0.218	0.011	0.011
Descartes Sparse + TrajOpt	0.026	0.139	0.150	0.011	0.015	0.022

As for the discrepancies induced by path planning, Table 7.2 shows that RRTConnect and Descartes Dense provided the same values for both maximum and average absolute error. Since the global optimal robot path is achieved with Descartes Dense, it can be inferred that RRTConnect, even with its randomness, also computed that very same path. Although unexpected, this result is not unreasonable to think due to the requested task being largely constrained, that is, without adding tolerances and with the manipulator possessing the same degrees of freedom (six) as those required to execute the deposition task. Moreover, in both planners, the addition of more waypoints did not improve the accuracy, it already converged with few waypoints. The linear interpolation performed by Descartes Sparse revealed large sporadic discrepancies, as demonstrated by the large difference between the maximum and average errors. Increasing the number of waypoints just increased these sporadic discrepancies magnitudes and occurrences. On the other hand, TrajOpt's perfor-

mance in terms of task-reaching ability improved considerably from 5 Hz to 25 Hz. For the latter rate, while the maximum position error was lower than those obtained with RRTConnect and Descartes Dense, its average was just slightly higher, probably caused by the additional inequality constraint associated with joints jerk, making it slightly smoother. In the case of TrajOpt initiated with Descartes Sparse, for 5 Hz, clear improvements were found, remaining closer to the requested tow course than when starting with a naive path. However, it quickly got polluted by the errors of Descartes Sparse as the number of waypoints was increased.

Nevertheless, compared with other sources of errors, such as the curve approximation performed in the last chapter, all the discrepancies induced by performing path planning were at a significant lower scale, μm instead of mm. Therefore, it safe to say that all three path planning algorithms studied do follow the requested cartesian path closely. In Figures 7.1 and 7.2, the joints path obtained when used Descartes Dense for the Tow Course 2 with 5 Hz is represented. With the two plots represented with an equal scale, the joint A6 is clearly the joint that moves the most, followed by the joint A1 and A3, with <http://www.meteo.pt/pt/index.html> the joint A4 not moving at all.

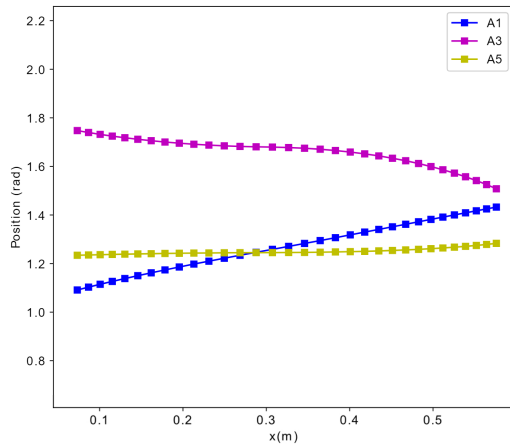


Figure 7.1: Joint A1, A3 and A5 paths obtained with Descartes Dense for Tow Course 2, with requested waypoints distributed at a rate of 5 Hz.

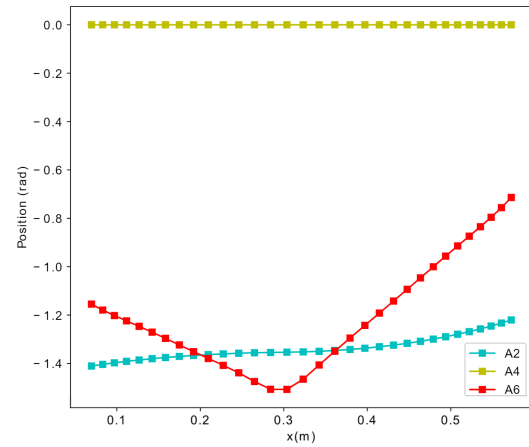
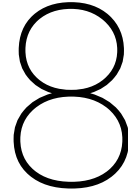


Figure 7.2: Joint A2, A4 and A6 paths obtained with Descartes Dense for fitted Tow Course 2, with requested waypoints distributed at a rate of 5 Hz

In conclusion, for a fully-constrained and collisions-free cartesian task, trials conducted revealed that both Descartes Dense and RRTConnect sampling planners provided similar results, greatly outperforming the TrajOpt optimization planner in terms of computational efficiency. TrajOpt with a naive initial path produced slightly smoother joints path if provided with a very refined cartesian path, at the expense of higher runtime and specially memory consumed. The potential of initializing TrajOpt with the outcome of a sampling planner allowed a closer performance to the previous RRTConnect and Descartes Dense planners in case of receiving few waypoints, as the interpolation performed by Descartes Sparse decreased the closeness to the requested paths as more waypoints were added. The flexibility offered by TrajOpt in specifying the motion problem allowed to obtain slightly smoother robot paths without compromising accuracy too much and should be explored in more detail in future research.



Constant Laydown Speed

The final process step to fully define a robot motion consists of establishing a timing law that aggregates the obtained joints path with the imposed dynamics limits. In this chapter, the joints position and respective velocity and acceleration profiles in function of time will firstly be calculated with trajectory planning algorithms available in MoveIt, and later the process step developed to maintain a constant laydown speed will be introduced. To avoid overcrowding this chapter with too much data, only the motion at the joint A6 will be analyzed here, as it will be susceptible to larger changes in position when Tow Course 2 is executed.

8.1. Available time parameterization in MoveIt

The three motion planning frameworks tested follow different procedures to tackle time parameterization. In Descartes, a maximum travel time between two successive cartesian waypoints can be directly assigned, with subsequent check of whether they comply with velocity limits of each joint. With such method, acceleration limits are never taken into account, eventually hampering the robot motion with extreme accelerations shifts. At the moment, Tesseract does not possess any trajectory planning algorithm, however in its associated path planner TrajOpt, a time condition can be assigned to the constraints introduced in the optimization problem. Unfortunately, for the time of writing, it was unclear how to properly configure TrajOpt optimization problem with time, as no concrete example is provided in its libraries. On the other hand, MoveIt is endowed with several classes to introduce a time condition, either by directly decreasing joints velocity and acceleration limits by a fraction (known as soft limits) or by complying with dynamic constraints in a post-processing step. It is in this post-processing step that bridges MoveIt with the research topic of trajectory planning, inserting the timing law to a joints path regardless of which path planner was adopted.

With the joints path created by either RRTConnect, Descartes or TrajOpt, time parameterization was initially conducted using the three different trajectory algorithms integrated in MoveIt, named Iterative Parabolic Time Parametrization (IPTP), Iterative Spline Parametrization (ISP) and Time-Optimal Trajectory Generation (TOPP). Note that these algorithms not only allocate a time stamp to each waypoint, but also assign a joint velocity and acceleration value. A time stamp consists of the travel time between two consecutive points plus the sum of all previous travel times. The first waypoint is always assigned a time stamp of zero.

As mentioned in Chapter 2, time parameterization involves dividing the desired trajectory into two elements, path geometry and timing law. Of these three algorithms, IPTP is the sole one where just the timing law is computed, thus preserving the path geometry originated by the path planner. With IPTP, the travel time between two waypoints is iteratively calculated by using first order finite differences and by assuming that one joint is always moving to its highest capabilities. Which joint is moving to its highest capabilities is dictated by the one that takes more time to shift from its current to its next position. Whether this time corresponds to a joint moving with its absolute maximum velocity or acceleration is evaluated in two stages. The velocity limits are initially imposed, with the travel time given by the displacement between current and next joints positions and divided by their maximum velocities. Subsequently, IPTP checks whether the acceleration limits are respected for the travel time previously computed with the velocity limits. For this purpose, two velocities are arranged using three waypoints (prior, current and next), implying that two travel times are being simultaneously applied to obtain a single acceleration value. If the travel time imposed by the velocity limits does not overcome any of the acceleration limits, then the motion between the current and next waypoints is constrained by the absolute maximum velocity. If, on the other hand, any acceleration limit

is exceeded, then the motion is constrained by the absolute maximum acceleration and each travel time is iteratively modified. IPTP starts by iterating forward (starting from the first waypoint) while expanding the time between the current and next waypoints by a constant factor of 1% until the acceleration is reduced to its maximum value. After the last waypoint has been reached, the iteration is repeated backwards, expanding the time between the previous and the current waypoint in the same manners.

With the time stamps achieved, a heuristic method is used by IPTP to set the joints velocity and acceleration values on each waypoint, described by Equation 8.1. Represented with \mathbf{a}_i , \mathbf{v}_i and \mathbf{q}_i are the joints acceleration, velocity and position vectors at the i waypoints and dt_i the travel time between current and next waypoint. Special case is applied for the first and last waypoints, where the $i - 1$ and $i + 1$ waypoints are replaced by the $i + 1$ and $i - 1$ waypoints, respectively.

$$\mathbf{v}_i = 0.5(d\mathbf{v}_1 + d\mathbf{v}_2) \quad \text{and} \quad \mathbf{a}_i = \frac{d\mathbf{v}_2 - d\mathbf{v}_1}{0.5(dt_{i-1} + dt_i)}, \quad \text{with} \quad (8.1)$$

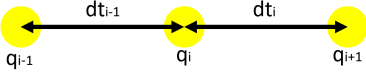
$$d\mathbf{v}_1 = \frac{\mathbf{q}_i - \mathbf{q}_{i-1}}{dt_{i-1}} \quad \text{and} \quad d\mathbf{v}_2 = \frac{\mathbf{q}_{i+1} - \mathbf{q}_i}{dt_i}$$


Figure 8.1: Illustration of the joint position and time nomenclature used for IPTP method, with exception for the first and last waypoints.

Instead of adopting the above equations, the ISP fits the joints position trajectory into cubic splines makes use of their first and second time derivatives to set up the joints velocity and acceleration. Prior to that, the travel time between waypoints is calculated much identical as was previously explained for IPTP. In order to specify an initial and final accelerations, the ISP can allocate two additional waypoints next to the first and last waypoints of the joints path, effectively modifying their path geometry. At both first and last points, the velocities and accelerations are then set to zero, ensuring their continuity at the junction with previous and next motions.

Distinctly different and more elaborate than the previous two are TOPP algorithms, with their common theoretical backbone already detailed in Chapter 2. MoveIt adopts the TOPP algorithm proposed in the article [69]. This algorithm seeks to obtain the joints trajectory that maximizes the time derivative of the path parameter (in this case represented by the arc length) at every point along the path. Thereby, its first step consists of parameterizing the joints path by their arc length and blending them into a sequence of linear and circular segments. As a result, these new joints position curves are inevitably constructed by approximating the outcome of the path planner, with expected inaccuracies being limited to a specified maximum discrepancy value (in radians). Next, two limits curves for the time derivative of the arc length are inferred by assuming the joints to be moving at their highest velocity and highest acceleration. These limit curves are mathematically established by equalizing the joints velocity and acceleration limits with the first and second time derivative of the joints course, respectively, followed by separating the time derivative of the arc length that arises from applying the chain rule to the previous equalities. Further details of these relations are referred to the article [69]. A list of switch points (points where an acceleration step occurs) is then formed by the intersection of the velocity and acceleration limit curves. The time derivative of the arc length profile can now be built by first integrating forward with maximum acceleration, until a segment containing a switch point is reached, to then backwards integration is used from this point using minimum acceleration until the start of the segment is again reached. With the time derivative of the arc length derived, the joints velocity for each segment can be derived and so the travel time between waypoints. The resulting joints position, velocity and acceleration trajectory are finally obtained by resampling with new waypoints, separated by a fixed and predefined travel time.

Next, the results of the direct implementation of these three time parameterization algorithms when performed the fitted Tow Course 2 with waypoints separated by 20 mm (a total of 32 waypoints) are presented. To scale the end-effector speed closer to the desired 0.1 m/s, the velocity limits of the joints were reduced to a mere 5% of the robot maximum capabilities. As such, the velocity at the joint A6 was constrained to a maximum of 0.191 rad/s (5% of 3.822 rad/s), while the acceleration limits remained at 2 rad/s² for all joints. Figure 8.2 contains the laydown speeds obtained and Figures 8.3, 8.5, 8.6 the resulting position, velocity and acceleration trajectories at the joint A6, respectively. In Figure 8.4, the discrepancies of the obtained cartesian paths after time parameterization are presented.

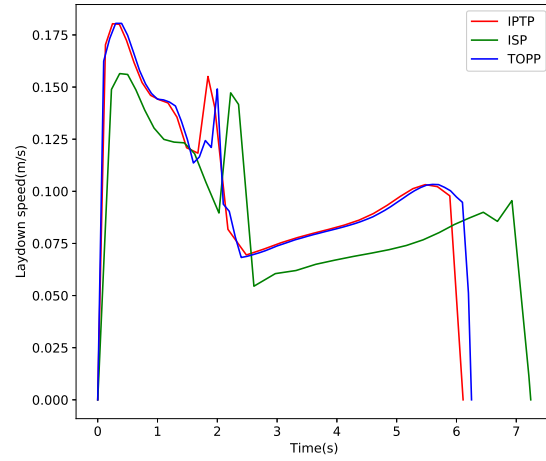


Figure 8.2: Laydown speeds for fitted Tow Course 2 obtained using time parameterization methods available in MoveIt.

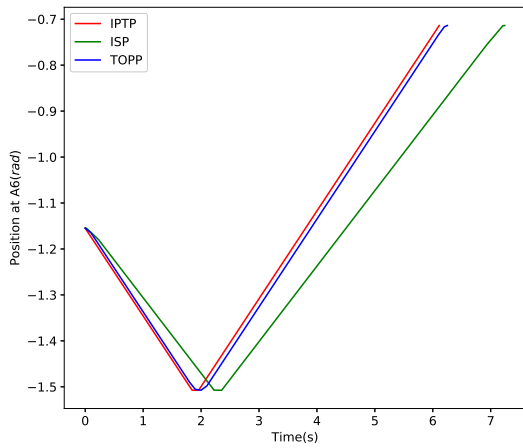


Figure 8.3: Joint A6 trajectory for fitted Tow Course 2 using time parameterization methods available in MoveIt.

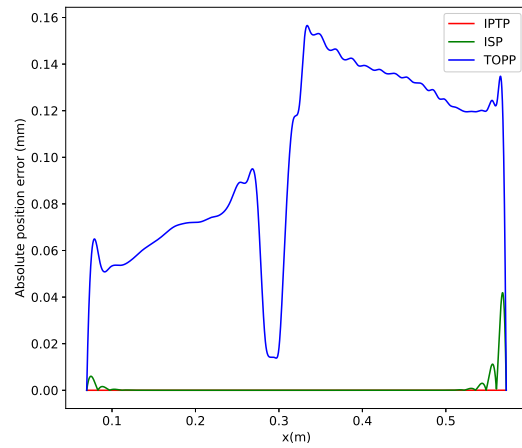


Figure 8.4: Discrepancy induced at the fitted Tow Course 2 by time parameterization methods available in MoveIt.

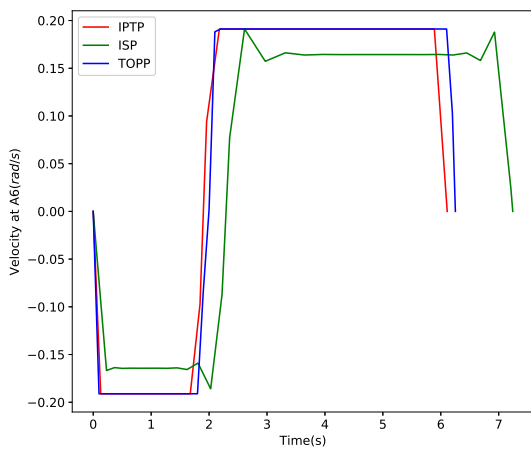


Figure 8.5: Joint A6 velocity for fitted Tow Course 2 using time parameterization methods available in MoveIt.

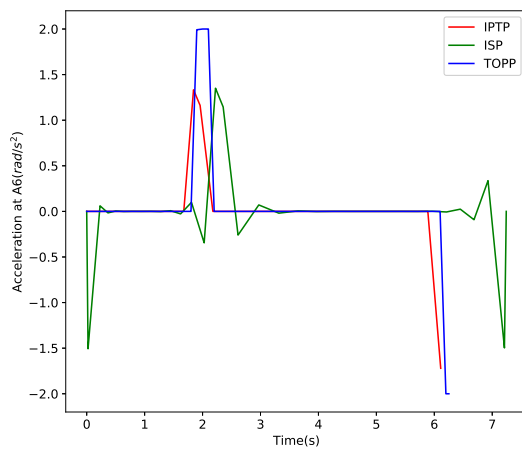


Figure 8.6: Joint A6 acceleration for fitted Tow Course 2 using time parameterization methods available in MoveIt.

Since all three algorithms were developed to minimize motion time, from Figure 8.2 it can be inferred that, for complex VSL courses, the end-effector speed would vary significantly along the tow course if such algorithms were adopted. These would be the expected laydown speed profiles if increasing productivity was the desired outcome. For VSL deposition, it became clear by Figure 8.5 that the joint A6 was the limiter factor of the robot motion, with its velocity almost always at its maximum value. With the joint A6 velocity kept constant, Figure 8.2 shows that the laydown speed had to be reduced in order to compensate regions with a smaller turning radii (right before and after the inflection points). Nevertheless, considering that the velocities at the joints were even reduced to only 5% of its maximum performance, an increase of laydown speed up to 20 times could be expected, if the robot acceleration limits allow it.

An overall comparison between the outcomes of the time parameterization algorithms is now carried out. The IPTP provided the shortest total travel time of the three, with 6.11 s, while the TOPP performed with a slight increase of 2% and the ISP evidenced a significant raise of 18% of the time. The large time disparity at ISP can be understood by checking the velocity at the joint A6 in Figure 8.5. The ISP seemed to undermine the joint performance, remaining primarily below its maximum of 0.191 rad/s. Of all the three trajectories, the TOPP trajectory is the only one where the joint velocity and acceleration followed a well defined profile, with a trapezoidal velocity and an almost step-shaped acceleration. However, this was achieved by modifying the original joints path, with even a small tolerance in the joint space of 0.001 rad resulting in a discrepancy of 0.156 mm in the tow course, as shown in Figure 8.4. When inspecting Figure 8.6, it can be observed that both IPTP and TOPP do not take into consideration the velocity increase that occurs at the start of the course when constructing the acceleration function. The reason for this is that both algorithms force a zero acceleration at the first joint waypoint to maintain continuity with previous motions. This inconsistency could eventually lead to unpredictable acceleration peaks at the start when the position controller discretizes the joints trajectory, which will be reflected in the actual system. Since the ISP algorithm adds two additional waypoints near the beginning and end of the joint path, this initial acceleration input is recorded.

8.2. Impose a constant laydown speed

Aiming at in-situ consolidation, a constant heating time along the fiber tow course can be pursued by holding a constant laydown speed. Given that none of the three time parameterizations available in MoveIt adds restrictions to the end-effector speed from the ground up, a different approach had to be designed. Nevertheless, these algorithms were suitable to endure that the robot does not exceed its dynamic limits, so reusing part of the code was a valid option.

It was decided to adapt an identical method to the IPTP algorithm, so that the path planning output was accurately followed and no further discrepancies were added. The developed algorithm workflow is represented in Figure 8.7. From the joints path obtained with the selected path planner, the respective end-effector position at each waypoint is calculated with forward kinematics and the travel time computed by subtracting two consecutive end-effector positions and dividing it with the desired laydown speed. Subsequently, it is checked whether the travel time just computed complies with the joints velocity and acceleration limits, as proposed in IPTP. If the travel time computed for a constant laydown speed has to be increased to comply with the robot limits, the obtained end-effector speed will be reduced between those waypoints. After calculating the travel time for the entire course, a following check evaluates the achieved end-effector speed. If the achieved end-effector speed is not equal to the desired laydown speed $\pm 1\%$ in at least 80% of the tow course, the desired laydown speed will be reduced by 10% and the algorithm will be repeated again for a lower desired speed. When a timing law that maintains a constant speed is finally achieved, the velocities and accelerations attached to the joints trajectory are calculated with Equation 8.1.

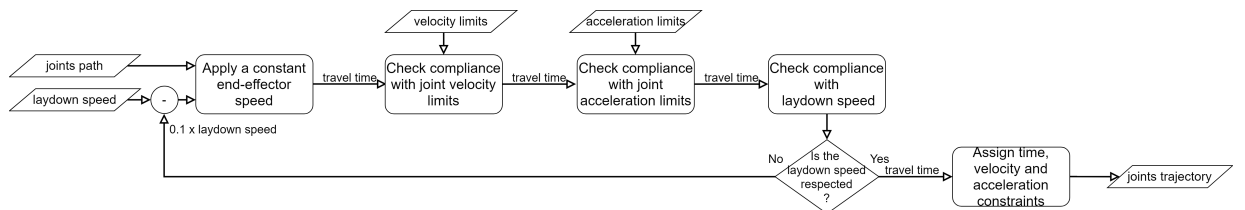


Figure 8.7: Workflow of the time parameterization algorithm developed to keep a constant laydown speed.

The developed time parameterization algorithm was tested for the fitted Tow Course 2 at a desired laydown speed of 0.1 m/s, with the joint A6 position, velocity and acceleration trajectories obtained displayed in Figures 8.8, 8.9 and 8.10, respectively. Note that the joints velocity is no longer constrained by a 5% soft limit, while the maximum acceleration still remains limited to 2 rad/s^2 .

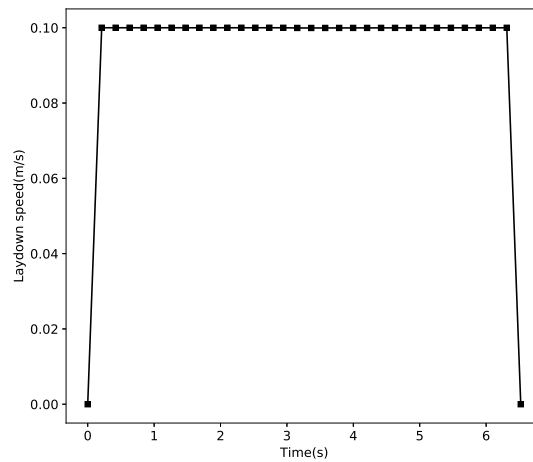


Figure 8.8: Laydown speed for fitted Tow Course 2 when enforced a constant end-effector speed of 0.1 m/s.

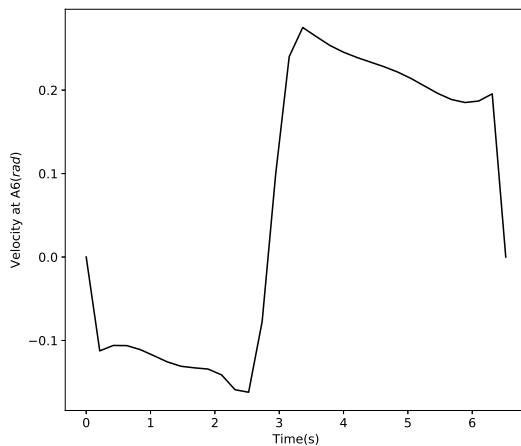


Figure 8.9: Joint A6 velocity for fitted Tow Course 2 when enforced a constant end-effector speed of 0.1 m/s.

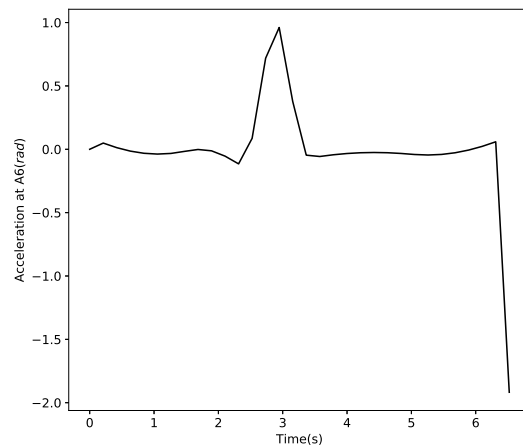


Figure 8.10: Joint A6 acceleration for fitted Tow Course 2 when enforced a constant end-effector speed of 0.1 m/s.

As shown in Figure 8.8, for the Tow Course 2, the developed algorithm was able to generate a robot trajectory with the desired laydown speed of 0.1 m/s. While in the first and last waypoints the robot was held stationary, the end-effector speed for the remaining waypoints was kept at approximately 0.1 m/s, with an almost insignificant average error of $\pm 0.018 \text{ mm/s}$ and median of -0.011 mm/s , making it actually slightly lower than 0.1 m/s for most of the tow course. Excluding the last waypoint, the maximum acceleration was detected near the inflection point and measured 1.2 rad/s^2 , well below the maximum imposed acceleration of 2 rad/s^2 . As such, higher constant laydown speeds are expected to be reached.

Two problems became apparent by adopting a method inspired by IPTP. By enforcing a zero velocity at the first and last waypoints of the tow course, the laydown speed was not kept constant between its first and last segments. Assuming that the laser reaches the desired power almost instantaneously, in these few millimeters of the course the tow will be overheated. The second issue concerns the initial and final accelerations that were requested at the joint A6. The algorithm was unable to detect the acceleration peak that should have occurred at the beginning, being assumed a zero acceleration. At the end, the joint A6 decelerated with maximum acceleration to completely stop the motion. While still within the acceleration limits, the robot

may not be able to execute the command due to requesting an elevated jerk. If, on the other hand, the jerk requested is within the robot limits, the command can be executed, yet with oscillations in the tool center may lead to inaccuracies. With this time parameterization, the joints jerk values are dictated by how far the two first and last waypoints are from each other. When equidistant waypoints are adopted, the more waypoints were used, the closer they were from each other, hence the jerks applied at the joints to change from 0 to 0.1 m/s was higher. Therefore, joints jerk were solely controlled by the number of waypoints that the course was divided, without checking their dimensions beforehand.

In order to better control the jerk evolution, either a method like TOPP is adopted, where the trajectory would be modified to obtain high degree acceleration profiles, or a jerk limited algorithm is implemented. The former should be avoided for tasks where a high level of accuracy is required, however, current MoveIt release does not contain any jerk limited time parameterization. Next section, a solution is presented to prevent these jerk problems from affecting the tow course.

Nevertheless, Figures 8.9 and 8.10 give a good overview of how the robot will behave when steering VSL tow courses at a constant laydown speed. Starting from an initial zero velocity, the joint A6 initiates its trajectory by rotating counterclockwise. As the turning radius is reduced, the velocity at the joint A6 is increased in order to maintain a constant laydown speed. To get through the inflection waypoint, the joint A6 has to change from rotating counterclockwise to clockwise while keeping a constant end-effector speed. This complete change in the velocity direction in a very short space is associated with a significant acceleration step in the joint A6. This may justify why robot controllers have problems keeping a constant end-effector speed during deposition of VSL courses. Soft limits imposed by the robot controller reduce the maximum acceleration allowed, thus having to reduce the speed to avoid these large acceleration changes. After the inflection point, the turning radius continuously increases and the joint velocity is again reduced until the end of the course. It is worth noting that, by itself, even a small minimum turning radius of 0.420 m had little impact on the robot motion, requesting a minor acceleration response from the joint A6.

8.3. Soft approach to the tow course

With the objective of guaranteeing a constant laydown speed throughout the entire tow course and anticipating/postponing the respective acceleration spikes at the start/end of the joints trajectory, a soft approach to the tow course was incorporated to the start and end of the requested end-effector path. This soft approach was composed of two segments: an angular segment, where the end-effector descends/ascends with a certain angle; and an extension segment, which is a direct expansion of the tow course. For the sake of simplicity, from now on, the emphasis is on the soft approach added at the beginning of the path. In Figure 8.11 the designed soft approach for a flat laminate is depicted, with θ and $L_{angular}$ representing the angle and length that characterize the angular segment and $L_{extension}$ the extension segment length. The three variables θ , $L_{angular}$ and $L_{extension}$ can be changed to be as large or as small as necessary. Figure 8.12 illustrates the Tow Course 2 with a soft approach at both its start and end.

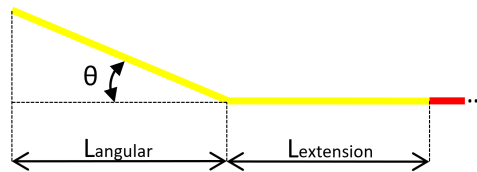


Figure 8.11: Illustration of the soft approach integrated at the start of a flat tow course. The red line represents the actual tow course to be performed.

Each segment of the soft approach was specifically designed to solve one of the problems that arise with the previous method. In the angular segment, the robot joints will accelerate until the desired laydown speed is reached. The reason this was done at an angle was to make sure the tow is under the roller and in the intended direction at the start of the course. The angular segment was divided into several discrete waypoints such that the velocity can be slowly induced, avoiding excessive torques. Furthermore, when performed on a real robot, an overshoot in the laydown speed was noticeable when poor discretization occurred. The end-effector positions during this segment were derived by separating the waypoints with equal distance and following the tangent of the extension segment while linearly adding a component in the z -axis. The end-effector orientation was kept the same throughout this segment and was equal to the end-effector orientation

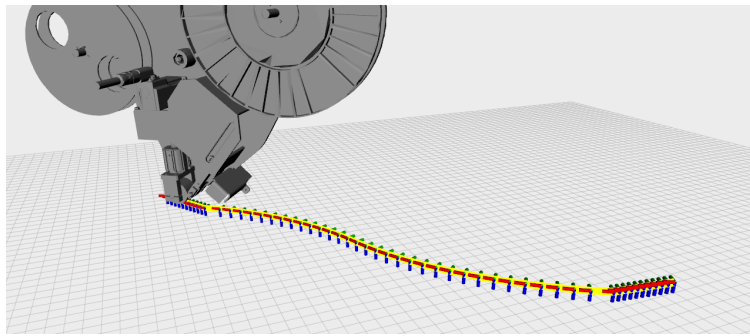


Figure 8.12: Representation of the soft approach combined with Tow Course 2, with an approach angle of 10°

at the start of the extension segment. During the angular segment, the end-effector velocity was increased by a fraction of the desired laydown speed, fraction which consisted of i/n , where $i \in [1, n]$ is index of the current waypoint for a total of n waypoints. With waypoints separated by the same distance, the travel time between waypoints was derived by t_1/i , where t_1 corresponds to the travel time between the first and second waypoints. As such, the end-effector speed profile during the angular segment will resemble a hyperbola. To have a linear increase of the end-effector speed, the distance between waypoints would also have to increase by the previous fraction.

The actual deposition starts at the extension segment. This segment was added to withstand any acceleration left so that a constant laydown speed can be achieved without disturbances. As this segment is cut off in a subsequent step, changes in the joints velocity and acceleration will not impact the final product. In an initial attempt, the extension segment was designed as a straight line, holding the same tangent as the first waypoint of the tow course, being represented in Figure 8.13. With this linear approach, however, it was noticed that the joint A6 was only changing its position to retain a same end-effector orientation while the other joints were moving, with significant changes in its position only experienced at the start of the tow course, as depicted in Figure 8.14. Hence, the acceleration peak at the joint A6 occurred at the first waypoint of the tow course, shown in Figure 8.15, leading to a high jerk command affecting the start of the tow course.

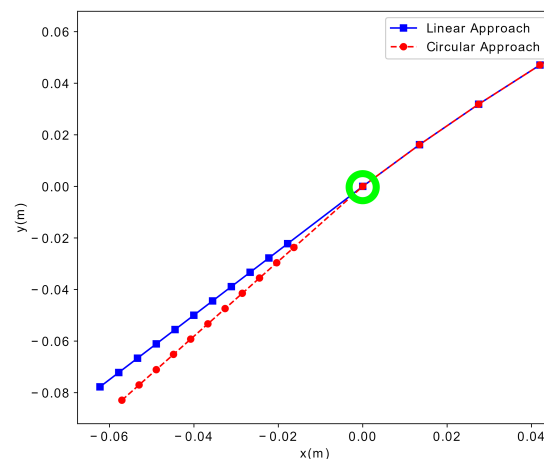


Figure 8.13: Soft approach integrated into Tow Course 2 with linear and circular extension segment. The green circle represents the first waypoint of the actual tow course.

In a second attempt, the extension segment was designed with a circular shape, following the turning radius at the start of the tow course. As deduced from Figure 8.15, with this circular approach, the high jerk command on the joint A6 was anticipated to occur during the soft approach rather than affecting the actual tow course. Even with this efforts, an acceleration peak still took place at the initial course waypoint. Ideally, this peaks in acceleration at the start should be completely anticipated during the circular extension. The key factor causing this occurrence was the simplification of a circular shape into a single segment, depicted in Figure 8.13. This worked well for the linear extension segment, but too simple for a circular shape. Nonethe-

less, a reduction of 60% in the acceleration at the first waypoint of the tow course compared to the linear extension supports the hypothesis that accelerations will be anticipated with a circular extension. For best results, the extension course should be included when the tow course is fitted into a quintic spline, back in the Pose Builder node.

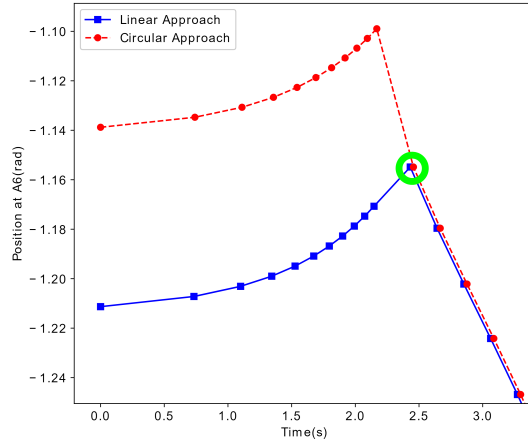


Figure 8.14: Close up at the soft approach effects into the joint A6 position with a linear and a circular extension segment. The green circle represents the first waypoint of the actual tow course.

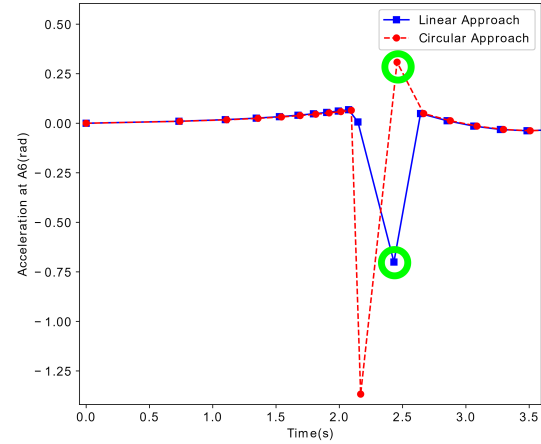


Figure 8.15: Close up at the soft approach effects into the joint A6 position with a linear and a circular extension segment. The green circles represent the first waypoints of the actual tow course.

Finally, Figure 8.16 contains the achieved laydown speed when incorporated a soft approach at both start and end of the Tow Course 2, with angular approach performed with length 70 mm and angle of 10° and extension approach with length 30 mm.

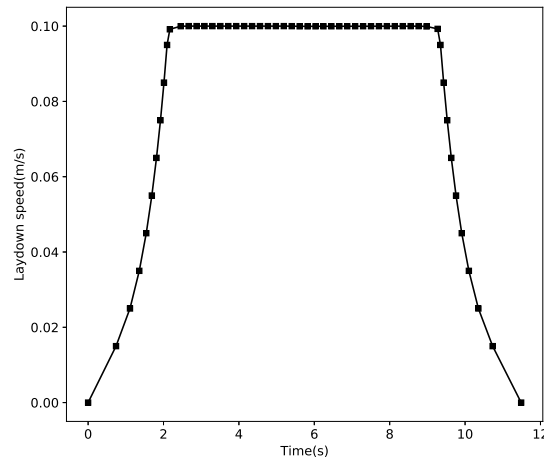


Figure 8.16: Laydown speed for fitted Tow Course 2 when enforced a constant end-effector speed of 0.1 m/s and incorporated a soft approach at both start and end of the tow course.

Concluding, a constant laydown speed was achieved by introducing a soft approach at both ends of the tow course and by assigning a time condition to each tow course waypoint equal to the cartesian distance between waypoints divided by the desired laydown speed. Velocities and acceleration for each joint were provided using the same method as in the IPTP algorithm from MoveIt. A desirable laydown speed of 0.1 m/s was able to be reached without surpassing any of the robot limits. The accelerations caused by changes in the turning radius were shown to be fairly low, yet coupled with an inflection point may induce challenges to the robot motion as high accelerations are required.

Accuracy of External Control

With the joints path bearing a timing information, the LayLa offline programming step is completed and the sequence of motions that forms the deposition task can be transmitted to the robot controller. The current chapter aims to investigate the experimental robot performance acquired when the desired tow courses are executed with the LayLa framework. In a first stage, the path performed by the real joint A6 and its velocity and acceleration, all in function of time, will be compared with the outcome of the offline step. Afterwards, the executed end-effector path and its speed profile will be examined, to then draw conclusions about the accuracy of controlling a KUKA robot with an external source of commands.

In this chapter, the robot performance for the fitted Tow Course 1 and 2 will be analyzed, both computed using Descartes Dense as the path planner and kept a constant laydown speed of 0.1 m/s. Transport motions included in the deposition task were performed with standard RRTConnect and ITP for time parameterization, with joint velocities constrained to 5% of their limits. For the purpose of only analyzing the motion during the actual tow course, these transport motions were excluded from the assessment. The same argument was applied to discard the soft approach applied to both start and end of the tow course from the analysis. As performed throughout this thesis, only the joint A6 will be further analyzed in more detail. As for the end-effector pose, only its position and speed will be here analyzed. The complete description of the robot configuration during the tow course, that includes the six joints and end-effector components, is exposed in Appendix B.

9.1. Experimental joint trajectory

Throughout this chapter, data collected from three sources was compared. After time parameterization, all the course waypoints had associated a time, joints position, velocity and acceleration condition, to which this collective of data was called joints request. With KUKA manipulators, the RSI driver provides control capabilities over the joints position, hence neither velocity or acceleration commands can be directly submitted to the robot controller. Consequently, the selected ROS controller utilized the previous joints request to compute a new set of joints position, at a predefined rate, that followed the requested velocities and accelerations profiles. This new joints trajectory were designated as joints command. Via hardware interface, these commands were periodically forward to the respective robot controller, which then creates, executes and returns the robot configurations for each sampling time, obtaining the designated joints state. For the joints command and state, their velocities and accelerations were calculated using Equation 8.1.

Within the `ros_control` repository, several types of controllers are offered to oversee the robot motions. They range from having a direct control over the position, velocity, acceleration to even the torque of each joint. Since only the joints position can be controlled externally, there are few options of controllers that can be adopted. The most direct approach would be to use the so called `position_controller`, in which the joints position would simply be forwarded at the specified rate. However, previous users reported that, in these industrial robots, the motions generated with `position_controller` were extremely jerky, with abrupt vibrations not only visible but also audible. The latter hindrance can be addressed, to some extent, by adopting a different controller named `joint_trajectory_controller`. With this controller, a preliminary step is carried out in which each joints trajectory is mapped with a quintic spline, with the requested velocities and accelerations used as the respective first and second derivatives of the starting and ending for each spline segment.

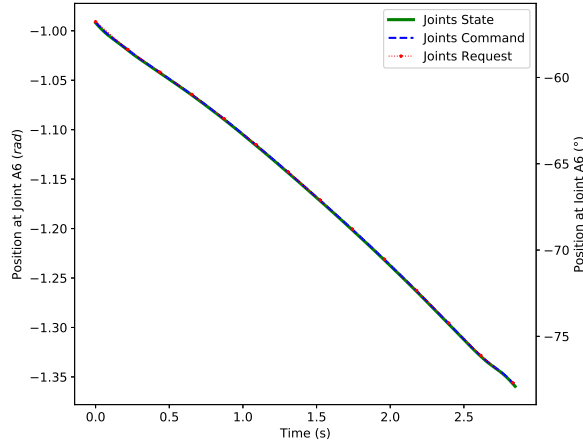


Figure 9.1: Joint A6 trajectory for fitted Tow Course 1, with requested waypoints distributed at a rate of 5 Hz

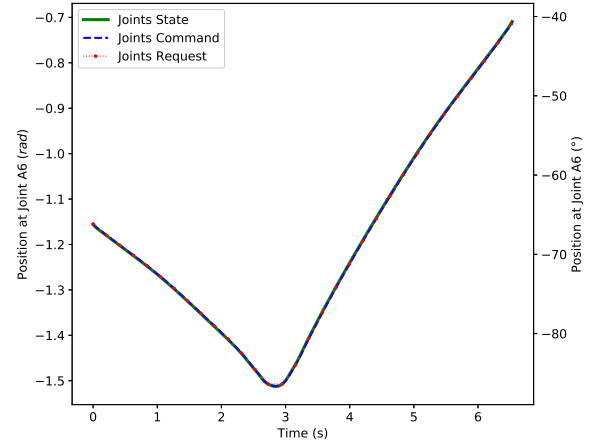


Figure 9.2: Joint A6 trajectory for fitted Tow Course 2, with requested waypoints distributed at a rate of 5 Hz

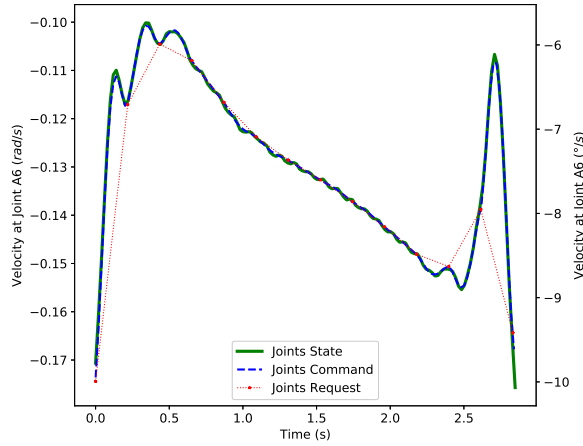


Figure 9.3: Joint A6 velocity for fitted Tow Course 1, with requested waypoints distributed at a rate of 5 Hz

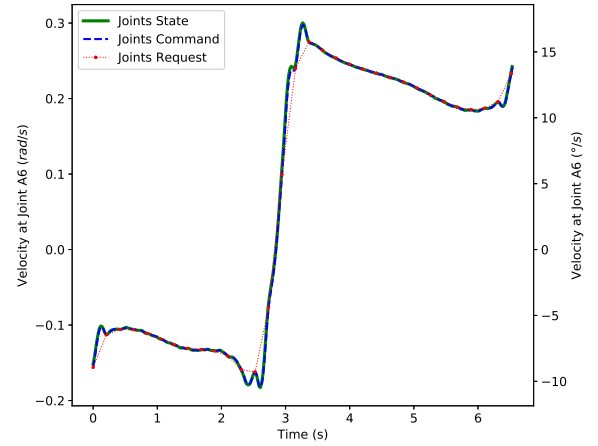


Figure 9.4: Joint A6 velocity for fitted Tow Course 2, with requested waypoints distributed at a rate of 5 Hz

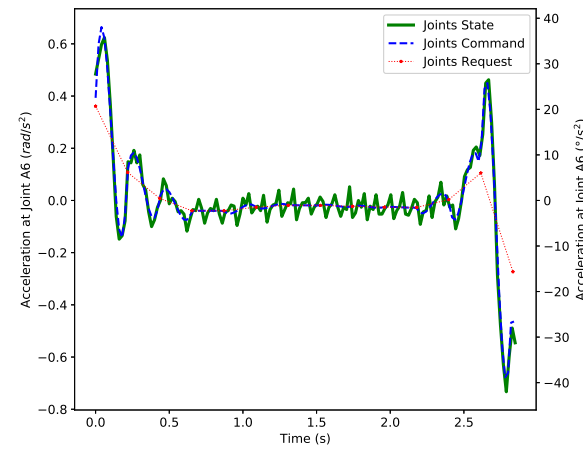


Figure 9.5: Joint A6 acceleration for fitted Tow Course 1, with requested waypoints distributed at a rate of 5 Hz

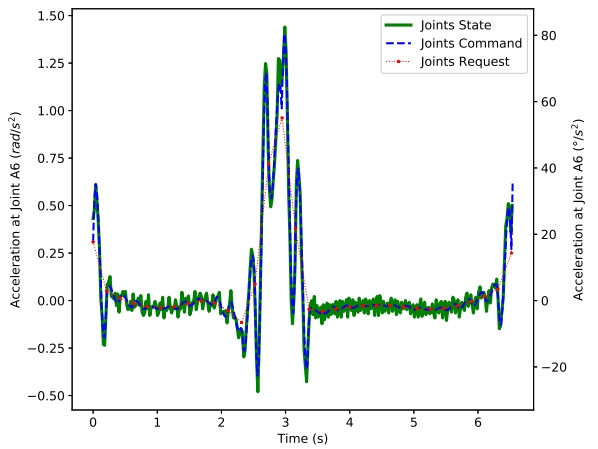


Figure 9.6: Joint A6 acceleration for fitted Tow Course 2, with requested waypoints distributed at a rate of 5 Hz

First and foremost, it is necessary to clarify the trajectory generated by the `joint_trajectory_controller`. In this study case, the joints request was built with two consecutive waypoints separated by 200 ms, corresponding to a requested publish rate of 5 Hz (assuming 0.1 m/s as the laydwon speed). By selecting a publish rate of the joints command higher than the one used to divide the tow course, the commanded output will add new waypoints within each of the requested segments, rounding off any possible discontinuities and therefore achieving an overall smoother robot motion. To then check the controller capabilities, a publish rate of 50 Hz for the joints command was selected. As a result, eight new commanded waypoints were created between two requested waypoints. With Figures 9.1 to 9.6 the joint A6 position, velocity and acceleration are displayed, being on the left represented for the Tow Course 1 and on the right for the Tow Course 2.

From Figures 9.1 and 9.2, it can be inferred that the requested joint A6 trajectory was closely followed by both ROS controller (joints command) and robot controller (joints state). However, differences became noticeable when the same comparison was applied for the velocity and acceleration profiles. In Figures 9.3 and 9.4, it was detected that, while at the requested waypoints all three velocities were identical, between those waypoints the response was characterized by an underdamped oscillation, more pronounced in cases where larger velocity shifts were requested. This was a direct compromise that comes with the `joint_trajectory_controller`, as the interpolation replaced the requested linear velocity segments with quintic splines in order to smooth out any possible edges. Nevertheless, the velocity state returned by the robot controller did seem to indicate a close follow-up of the commanded joint velocity.

If any position errors exist, these will be exacerbated when the acceleration is calculated. And that is the case, with the differences in the three curves becoming more perceptible when compared the acceleration with Figures 9.5 and 9.6. Oscillations in the commanded joints position were more pronounced and distinctions between the robot real motion and the outcome of the ROS controller became apparent. The real acceleration was characterized by a white noise (having discrepancies with an identical frequency and amplitude), likely caused by some encoders reading errors combined with the small number of decimal places stored by the RSI and sent to the hardware interface. This evidence is reinforced by the fact that even when the robot was completely stopped, some minor position changes were still detected (visible at the joint A4 in the Appendix B). Since the minimum recorded size coming from the robot was of 0.0001° ($\approx 1.75 \mu\text{rad}$), compared to 0.1 nrad from the ROS controller, some rounding must have occurred. If a $\pm 0.0001^\circ$ displacement was detected between two waypoints, the minimum acceleration recorded between time of 4 ms would be a significant $\pm 6.25^\circ/\text{s}^2$ ($\approx 0.109 \text{ rad/s}^2$), specially noticeable when working with low accelerations, as is the case in this study. To reduce both the frequency and amplitude of this white noise, the acceleration plots presented in this thesis were constructed with only a fifth of the data provided by the robot. Thus, the minimum acceleration recorded was reduced to $\pm 0.25^\circ/\text{s}^2$ ($\approx 0.004 \text{ rad/s}^2$) and the acceleration profile became closer to what was expected. It should be noted that, in reality, it is likely that these errors will not be noticed by the robot. If such was the case, then the robot would definitely complain about reaching the torques limit (hence a halt state) or noisy vibrations would occur, which was not the case.

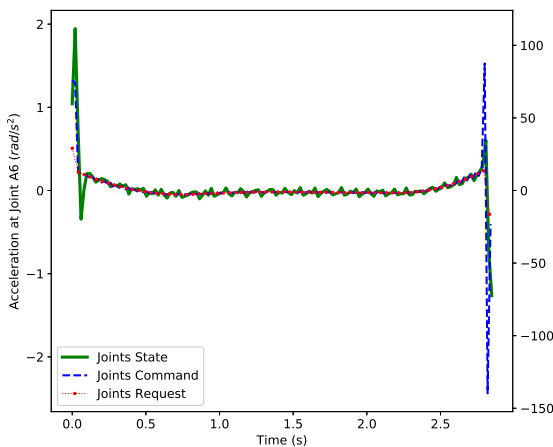


Figure 9.7: Joint A6 acceleration for fitted Tow Course 1, with requested waypoints distributed at a rate of 25 Hz

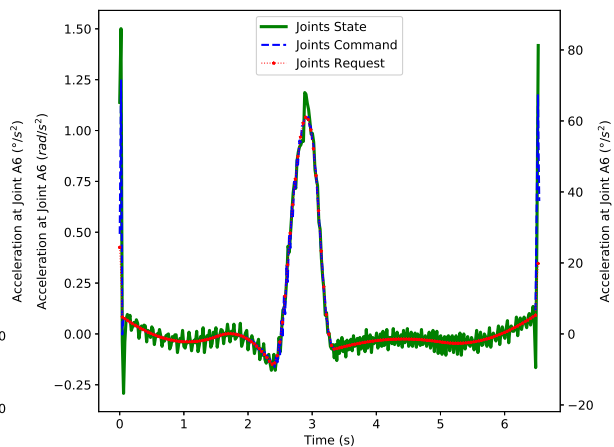


Figure 9.8: Joint A6 acceleration for fitted Tow Course 2, with requested waypoints distributed at a rate of 25 Hz

Given that the tow courses were already fitted into quintic B-splines prior to the path planning, the underdamping effect originated by the `joint_trajectory_controller` can be reduced by approximating the publish rate of both joints request and joints command. Figures 9.7 and 9.8 present the acceleration obtained for the respective Tow Course 1 and 2 when courses were discretized with a distance of 4 mm between waypoints, that is a rate of 25 Hz, adding a single new state waypoint between two requested waypoints. By dividing the tow courses with more waypoints, the velocities and accelerations profiles were overall followed more accurately, but it comes at the cost of increasing computational time and also the accelerations at first and last waypoints. As explained in the previous chapter, the poor discretization of the extension segment resulted in a significant deceleration during the first segment and acceleration in the last segment of the tow courses. If a higher rate is applied to divide a tow course, time between waypoints is reduced, leading to accentuated acceleration at those waypoints.

The effects of the turning radius on the robot motion were captured through the joint A6 velocity plots. For Tow Course 1, the almost linear decrease of the turning radius along the course was verified by a likewise linear increase of the absolute value of the velocity, until it reached the minimum turning radius at 2.4 s. By requesting a linear increase in the joint A6 velocity, its respective torque is kept constant, allowing very smooth motions. The robot behavior for the Tow Course 2 was already extensively explained in previous chapters, observing the same pattern of joint A6 velocity and turning radius (with the exception of the inflection point, occurring at the peak of acceleration, 2.9 s) and reaching its minimum right after the region affected by the inflection point, at 3.5 s.

9.2. Experimental end-effector trajectory

In addition to provide the joints position, the RSI driver also returns the end-effector pose, calculated based on simple robot kinematics. With the ultimate goal of performing a task in the operational space, on top of analyzing the joints trajectory it is required to investigate the end-effector motion, assessing the accuracy of the performed motion with respect to the desired cartesian path. In KUKA robots, an end-effector pose is characterized in space by the six coordinates X, Y, Z, A, B and C. As the nomenclature suggests, X, Y and Z variables specify the cartesian coordinates in relation to a reference frame. The letters A, B and C provide the rotation about the x , y and z -axis which constitute the reference frame, respectively.

Inaccuracies in the end-effector pose are not a new subject, with a large research community dedicated to investigating the origin of these errors and developing solutions to mitigate those either based on calibration or online compensation [83–85]. Of the several sources of errors, the most relevant is given by the elastic behavior observed in a joint when torque is applied, often designated as joint finite stiffness. To measure those errors, an advanced tooling setup, often involving camera(s) or laser tracker(s), is necessary. However, this measurement would demand for a much more focused study on this subject. For this thesis, it was decided to evaluate the end-effector position provided by the RSI. Despite studies [84, 85] mentioning that RSI underestimated the value of the actual error, the place where the maximum error is located coincided with the results measured with a laser tracker. Note that most of these analyzes were performed using robot native languages, not by external control as herein proposed.

To compute the error induced by the external control, the first task consisted of correlating the path created by the robot controller with the one created by the ROS controller. The correlation of both executed and commanded paths was derived by shifting the former such that the maximum absolute error is reduced as much as possible. This method was divided into two steps: intersection of the executed path with the commanded path at half of their length; search of the lower maximum error between both paths. The two-step correlation method is depicted in Figure 9.9. As the executed path had five times more waypoints (250 Hz) than the outcome of the ROS controller (50 Hz) and their lengths did not exactly match (due to the prior noises, the executed path was micrometers longer), the intersection of both paths proved to be insufficient to make both path as close as possible. The remaining offset to be added to the executed path was calculated through an iterative step, starting by finding at which waypoint the maximum error occurred and followed by adding to the offset a fixed displacement on the axis where this error was greatest. This iterative step was repeated until identical maximum errors were found back and forth within the same two waypoints and without further reductions in the error.

The obtained end-effector positions for the fitted Tow Course 1 and 2 are represented in the respective Figures 9.10 and 9.12. Overall, the robot performed both tasks correctly, clearly preserving the shape of each tow course. Still, since the joints motion did suffer some minor modifications from what was commanded, it is wise to expect that those changes could have impacted the end-effector motion and is essential to quantify this error.

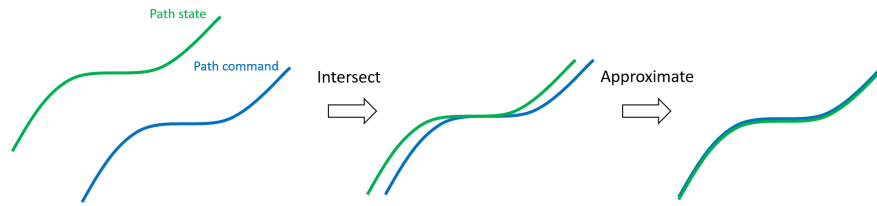


Figure 9.9: Illustration of the two-step method to draw the commanded end-effector path closer to the executed outcome.

The absolute position errors calculated with the two-step correlation method for both courses are presented in Figure 9.11 and 9.13, respectively. As expected, the errors detected were higher at the first and last waypoints, where the acceleration variations were more pronounced. Without inflection points requesting accelerations and decelerations, the absolute discrepancy in the Tow Course 1 was kept at an average of approximately 0.03 mm. On the other hand, Tow Course 2 detected an increase in the error at the inflection point, almost reaching 0.09 mm, and elevating the average discrepancy to approximately 0.05 mm. Both these average errors were lower than those reported in the literature (for similar manipulators, around 0.1 mm [85]) where proper tooling and compensation was applied. Therefore, it is expected that larger errors should be encountered in reality, yet the shape of the error curve should be similar.

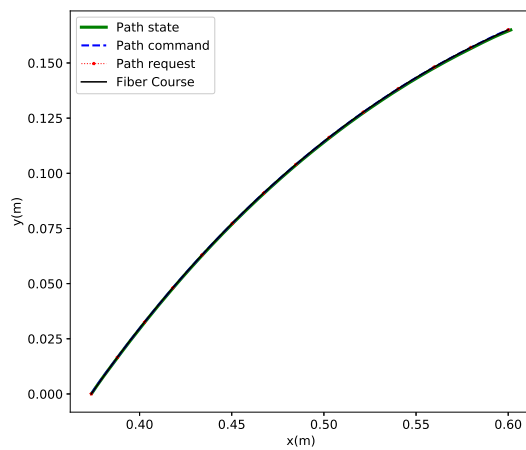


Figure 9.10: End-effector path for fitted Tow Course 1.

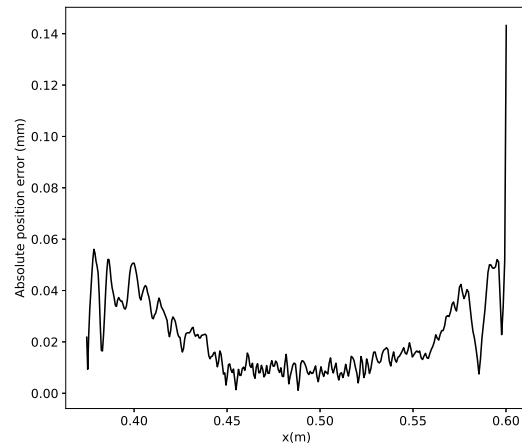


Figure 9.11: Discrepancy between the fitted Tow Course 1 and path performed by the robot.

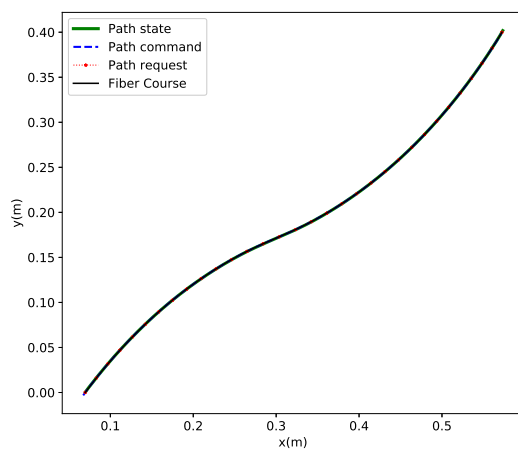


Figure 9.12: End-effector path for fitted Tow Course 2.

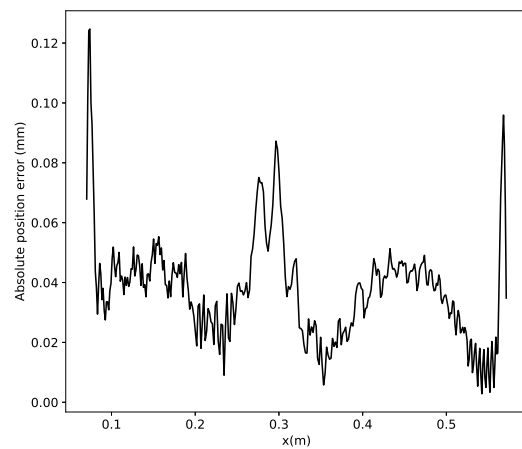


Figure 9.13: Discrepancy between the fitted Tow Course 2 and path performed by the robot.

Lastly, the actual end-effector speed was compared with the desired laydown speed. In Figure 9.14 and 9.15 the obtained laydown speed for Tow Course 1 and 2 are presented, respectively. In a real system, the end-effector speed was not completely kept constant along the path, with absolute errors reaching a maximum of 1.7 mm/s and 3.5 mm/s for the Tow Course 1 and 2, respectively. These maximum errors were observed in regions where a high acceleration step was requested, that the start and end of both courses and also at the inflection point for the Tow Course 2. In addition, a symmetry pattern in the absolute errors was detected, specially evident in the Tow Course 2. As the turning radius was being reduced, the laydown speed was kept slightly below the requested laydown speed of 0.1 m/s, and the opposite occurred when the radius was being increased. The same pattern was detected in Tow Course 1, but in this case without the symmetry, as the turning radius was being reduced for most of the path.

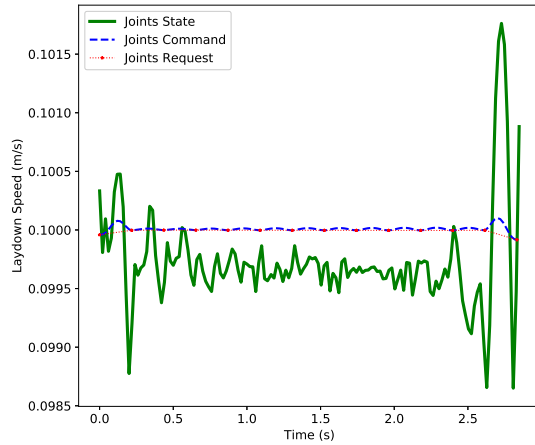


Figure 9.14: Laydown speed for the fitted Tow Course 1, with requested waypoints distributed at a rate of 5 Hz

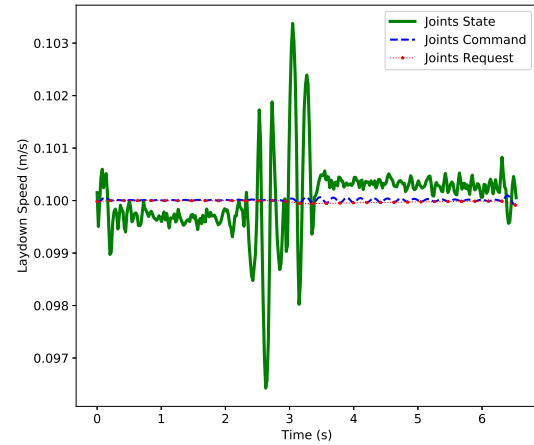


Figure 9.15: Laydown speed for the fitted Tow Course 2, with requested waypoints distributed at a rate of 5 Hz

In conclusion, creating and sending joint commands with ROS to a KUKA robot equipped with RSI resulted in an accurate trajectory tracking of the two analyzed fiber tow courses, both characterized with a reduced turning radius. With the adoption of the `joint_trajectory_controller`, the outcome of the time parameterization is interpolated to smooth out the requested motion, respecting the position, velocity and acceleration at the requested waypoints but not between these waypoints. Still, the low absolute errors reached at the end-effector position and speed corroborate the adoption of an external source to control the motion of KUKA robots.

V

Conclusion and Recommendations

10

Conclusions

Manufacturing laminates with a variable fiber orientation has been hindered by the inability to fully control industrial manipulators through their native programming languages along with the effort involved in adapting those languages for such complex cartesian paths. In an attempt to fill these gaps, an offline programming tool named LayLa was developed and tested for two VSL tow courses executed using AFP technology, without actual deposition. With the LayLa framework, the robot motion to accurately perform a requested tow course at a desired laydown speed is automatically generated. At its core, LayLa utilizes ROS and ROS libraries, developed over the years by the robotic research community, and its main body is divided into three pillars: Pose Builder, Path Planning and Time Parameterization.

The Pose Builder reads the laminate description and builds the sequence of end-effector poses to accomplish the requested tow course. In order to accept all kinds of paths, regardless of their geometry, each tow course was described as the discrete set of cartesian waypoints that constituted its centerline. Sudden changes in the tangent vector of the end-effector path were smoothed by interpolating the requested tow course with an approximated quintic B-spline. As with any approximation, discrepancies between the fitted and requested tow course are expected, yet these were estimated in the order of 0.1 mm, therefore not foreseeing large inaccuracies in the designed laminate.

As the name suggests, during the Path Planning the motion problem is constructed and solved using one of three state-of-the-art planners, obtaining the set of joints position to perform the fitted tow course. Trials were conducted for three compatible path planning algorithms, RRTConnect, Descartes and TrajOpt, when requesting a fully-constrained and collisions-free cartesian task to an industrial manipulator with six degrees-of-freedom. Both Descartes Dense and RRTConnect sampling planners returned identical robot paths and converged to the global optimal solution using fewer waypoints, hence greatly outperforming the TrajOpt optimization planner in terms of computational efficiency. Nonetheless, the flexibility offered by TrajOpt in describing the motion problem allowed to obtain slightly smoother robot paths without jeopardizing the accuracy significantly.

With Time Parameterization, a time condition that respects the robot velocity and acceleration limits is aggregated to all six joint path waypoints. Additionally, a constant laydown speed was enforced in order to improve the overall deposition process quality when in-situ consolidation is applied. A constant laydown speed over the entire tow course was achieved by extending both course ends and by assigning a time condition equal to the cartesian distance between waypoints divided by the desired laydown speed. Assuming in-situ consolidation with a 3 kW heat source, the maximum recommended laydown speed of 6 m/min (0.1 m/s) was reached, even with conservative acceleration limits of 2 rad/s^2 applied to all joints.

At last, experiments were conducted on a real robot, with LayLa's outcome being sent to the robot controller through external control. For tow courses defined on a flat surface, the joint in charge of rotating the end-effector about its symmetry axis was identified as the main source of issues, demanding higher accelerations to maintain a constant laydown speed while following their variable tangent vector. From the robot's perspective, tow courses with a constant or linear turning radius profiles could be performed without any difficulties, requesting low and almost constant accelerations at the previous joint. On the other hand, tow courses with a localized large shift in the turning radius, induced by inflection points, require greater precautions as the end-effector needs to completely change its rotating direction in a short period, only possible with an significant acceleration step. When large acceleration shifts were requested, it was visualized an overshoot

in the robot velocity and acceleration response, leading to slightly larger inaccuracies at the end-effector position and laydown speed. Nevertheless, the overall results revealed an accurate trajectory tracking of both joint and operating space variables, with discrepancies at the end-effector position around 0.05 mm and at the laydown speed of 3.2 mm/s, corroborating the use of LayLa to create external commands for composite deposition.

LayLa establishes an innovative proof of concept on how to tackle robotic systems for composite deposition. By eliminating the costly and time-consuming programming task that currently undermines VSL production and by enhancing layup control across the entire deposition process, it will enable researchers to effortlessly simulate and fabricate composites using a direct implementation of their own fiber tows distribution outcome. Furthermore, LayLa is agnostic from machine suppliers hardware and its openness allows other functionalities to be implemented by different parties in future iterations.

Recommendations

With the development of the LayLa framework, the proposed objectives of improving layup control, automating the programming task and preserving a constant laydown speed were achieved. LayLa serves as a proof of concept for achieving complete automation of the layup process using open-source frameworks. Nevertheless, additional functionalities could be introduced to LayLa in order to enhance the user's experience and extend its applicability to other automated composite technologies besides AFP. In addition, even though the trajectories computed by LayLa were tested on a real robot, its usage for manufacturing laminates needs to be experimentally validated with actual prepregs. At last, improvements could be done in certain ROS libraries to make the motion generated more trustworthy. Hereby, the most immediate recommendations for future work were divided into three sections: extension of LayLa capabilities, manufacturing laminates with LayLa and enhance ROS libraries.

11.1. Extend LayLa capabilities

- **One-step process to program a complete laminate:** For this initial effort, the LayLa framework was designed to program a single tow course when executed. A complete laminate can be achieved by individually requesting a new layer and course every time the robot completes the previous course deposition. Ideally, the entire robot motion for a complete laminate should be prepared in a single step, before the actual deposition process begins. Keep in mind that, when planning the robot motion for a complete laminate, additional transport motions between tows have to be defined, a problem often mentioned as sequence planning.
- **Add redundant axes to the robot system:** Automated composite technologies often rely on external axes, such as a linear track, for producing large structures, and a mandrel for closed-contour structures. With an end-effector pose completely defined with the six joints of a robot manipulator, an additional axis will make the robotic system redundant. Instead of having eight possible robot configurations for a single end-effector pose, redundant systems will have a much higher number of possible solutions (dozens if not hundreds, depending on the discretization applied for the redundant axis). Hence, more differences in the path planning outcomes will certainly be evident between different algorithms. Descartes Dense, being an optimal path planner, will easily become computationally extensive as it evaluates all possible solutions, one by one. In the presence of redundant systems, the developers of both Descartes and TrajOpt recommend using a hybrid approach, with Descartes Sparse providing an initial path to TrajOpt.
- **Incorporate fiber tow distribution strategies:** By receiving cartesian coordinates of the tow centerline, LayLa can accept any outcome of fiber tows distribution strategies, regardless of the course shape. Yet, such approach required to interpolate the waypoints so that a smooth tangent profile could be achieved, which ultimately led to approximations. Smoothing could be avoided by defining the tow courses not as cartesian coordinates but rather as piecewise polynomial curves. However, in the literature different VSL fiber tows distribution methods have adopted different curves and selecting one would compromise the desired universality of input. Instead, a parallel option could be developed by incorporating LayLa with fiber tows distribution strategies for both conventional and VSL laminates.

This would allow direct usage of the computed tow curves and LayLa would become a more completed and specialized framework for composite deposition.

- **Assess tow courses for curved structures:** In this thesis, tow courses defined on a flat surface were analyzed, where the binormal vector of the first waypoint is straightforward. For single or double curved molds, the same about the binormal vector cannot be said and a method to automatically select an ideal initial binomial should be developed. Furthermore, it would be worthwhile to analyze the robot behavior for these complex courses, since now the robot motion would require more the other joints besides relying mainly on the joint A6.
- **Intuitive interface:** LayLa relied on the command line to specify the deposition process, which can be seen as uninviting and intimidating for many users. It would be ideal to create a friendlier interface where all the process specifications could be easily selected, for example the number of the layer and course to be performed to also the laydown speed.

11.2. Manufacturing with LayLa

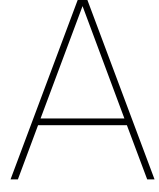
- **Measure the end-effector pose with proper tooling:** The accuracy assessment of the end-effector paths herein presented was made based on kinematics, that is, the end-effector poses were mathematically calculated using the joints positions. Previous literature [84, 85] stated that this approach underestimates the real discrepancies, thus an adequate methodology should be implemented with advanced tools, such as cameras or a laser tracker, to measure the real end-effector position error when joint commands are provided through an external source.
- **Test laying down fiber tows:** A comprehensive deposition quality assessment of LayLa needs to be carried out with actual prepreg, checking for tow misalignment, gaps, overlaps and others defects that might surge.
- **Adopt a laser power profile:** In this project, a constant heating time along the tow course was guaranteed by assuming a constant output power in the laser and enforcing a constant laydown speed. Still, maintaining these three variables constants will undermine the productivity, since, for the most part of the tow course, the robot is operating well below its maximum performance. Alternatively, a control system that relates the laydown speed to the output power of the source could be designed to maintain a constant heating time. A real-time control system could be performed, but delays from the measurement of the laydown speed to the laser response are very likely, especially if the rate that the laser and robot operate is different, causing local under- and/or over-heating of the prepreg. With LayLa however, since the laydown speed profile is calculated prior to the robot execution, another option would be to calculate the laser output power profile also offline and send it through external commands to the laser through ROS.

11.3. ROS libraries developments

- **Jerk limited time parameterization:** Even though jerk continuity can be guaranteed if the requested joint velocity and acceleration are kept below the robot's limits and the tow course is sufficiently discretized, this is by no means a replacement for a pure jerk limited parameterization algorithm. MoveIt does not currently possess a jerk limited time parameterization, but there are examples in literature [78] that could be adapted to ROS in the future.
- **Better alternative for the joint trajectory controller:** As seen from the experimental results, the quintic spline interpolation did not significantly impact the accuracy of the joints position, however, the same could not be said about their velocities and accelerations. Between the requested waypoints, the joints velocity and, more significantly, the joints acceleration profiles had some overshoots. Moreover, with a quintic spline it is not possible to control the jerk at other waypoints except the first and the last (which generally have a zero jerk). With LayLa, one solution could be to increase the number of waypoints which the tow course is divided in and use the position_controller instead of the joint_trajectory_controller, forwarding joints positions without any complicated interpolation. Since the end-effector path was already interpolated with a quintic B-spline, more waypoints allow to better replicate the joints trajectory with a high degree. Even so, this is far from optimal, being computationally expensive for the path planner.

VI

Appendix



Appendix A: Industrial Manipulator Kinematics

This appendix highlights the mathematical expressions that relate the end-effector pose with the robot configuration and enlightens two cases in which the general expressions are not applicable. This subject is definitely well-assessed in the classical literature [86, 87], however it constitutes a gap in most aerospace backgrounds. In addition to these references, Pires presented a well-written textbook [34] about the fundamentals of industrial robotics, in particular for robot manipulators, which will be used as guidance.

A.1. Manipulator geometric model

Before going much deeper into the geometric representation of a manipulator, some kinematic background must be first introduced. A rotation transformation, described by a 3×3 rotation matrix, establishes the relationship between the unit vectors of a coordinate frame A in relation to those of another coordinate frame B , with both frames having the same origin. Assuming \mathbf{v}^A and \mathbf{v}^B as the same vector represented in frame A and frame B respectively, the rotation transformation from one frame to another is given by $\mathbf{v}^A = \mathbf{R}_A^B \mathbf{v}^B$, where \mathbf{R}_A^B is the rotation matrix from coordinate frame B to frame A .

A coordinate transformation, described by a 4×4 homogeneous transformation matrix, combines translation and rotation between two coordinates frames A and B , now with both frames having different origins. Assuming \mathbf{v}^A and \mathbf{u}^B as vectors starting from the origin of their respective frame to an arbitrary point P , each coordinate transformation from one frame to another is given by equation A.1, where \mathbf{T}_A^B is the homogeneous transformation matrix from coordinate frame B to frame A .

$$\begin{bmatrix} \mathbf{v}^A \\ 1 \end{bmatrix} = \mathbf{T}_A^B \begin{bmatrix} \mathbf{u}^B \\ 1 \end{bmatrix}. \quad (\text{A.1})$$

A particularly interesting property in both rotational and transformation matrices for the robotic field is the possibility to express consecutive rotations or transformations by the product of their respective matrices. By sequentially combining all transformation matrices, the geometric model of a robot can be concise into a single coordinate transformation between the base frame and another frame at the end-effector. At a given joints position \mathbf{q} , the homogeneous transformation matrix that characterizes the robot is assembled using

$$\mathbf{T}_{n+1}^0(\mathbf{q}) = \prod_{i=1}^{n+1} \mathbf{T}_i^{i-1}(\mathbf{q}) = \begin{bmatrix} \mathbf{R}_{n+1}^0(\mathbf{q}) & \mathbf{p}_{xyz}(\mathbf{q}) \\ \mathbf{0}_{1 \times 3} & 1 \end{bmatrix}, \quad (\text{A.2})$$

where \mathbf{R}_{n+1}^0 describes the end-effector orientation by a rotation matrix from base to tool frames and \mathbf{p}_{xyz} represents the position vector of the end-effector in the base frame. With \mathbf{r}_i^{i-1} as the position of coordinate frame of link $i - 1$ relative to coordinate frame of link i , the homogeneous transformation between link $i - 1$ and i is computed as

$$\mathbf{T}_i^{i-1}(\mathbf{q}_i) = \begin{bmatrix} \mathbf{R}_i^{i-1}(\mathbf{q}_i) & \mathbf{r}_i^{i-1}(\mathbf{q}_i) \\ \mathbf{0}_{1 \times 3} & 1 \end{bmatrix}, \quad (\text{A.3})$$

The representation method previously explained does not provide a methodology on how to assign frames to each link, hence frames can be randomly assigned along the robot structure. However, for complex manipulators, the equations behind the rotation matrix may not be intuitive and the calculation of the transformation T_{n+1}^0 may end up being computationally expensive. Therefore, for consistency and computational efficiency, a standard convention for assigning frames is generally applied. Denavit and Hartenberg [88] proposed a convention that was largely embraced by the robotic community. In their convention, a coordinate frame is carefully placed such that the Z axis is assigned to the joint axis (in this case, axis of rotation) and the X axis is perpendicular to two consecutive joint axes, as represented in Figure A.1. By adopting the modified Denavit-Hartenberg notation proposed by Craig [86], the homogeneous transformation matrix between link $i - 1$ and i can be fully defined by a set of four parameters:

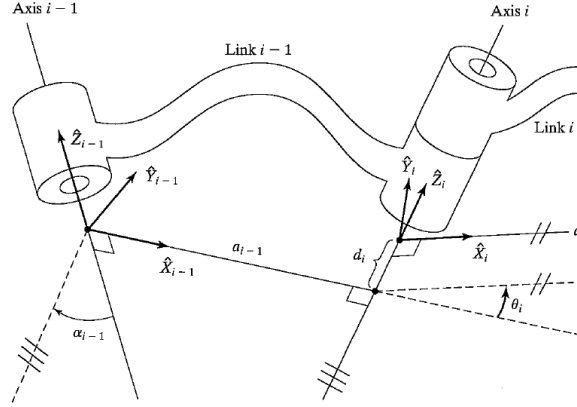


Figure A.1: Schematic representation of the Denavit-Hartenberg parameters proposed by Craig. Cited in [86], p. 68.

- Joint angle (θ_i), angle between X_{i-1} and X_i measured about Z_i ;
- Link twist (α_i), angle between Z_i and Z_{i+1} measured about X_i ;
- Link length (a_i), distance between Z_i and Z_{i+1} measured about X_i ;
- Joint offset (d_i), distance between X_{i-1} and X_i measured about Z_i .

The homogeneous transformation matrix between link $i - 1$ and i is then given by equation A.4.

$$T_i^{i-1} = \begin{bmatrix} \cos \theta_i & -\sin \theta_i & 0 & a_{i-1} \\ \sin \theta_i \cos \alpha_{i-1} & \cos \theta_i \cos \alpha_{i-1} & -\sin \alpha_{i-1} & -d_i \sin \alpha_{i-1} \\ \sin \theta_i \sin \alpha_{i-1} & \cos \theta_i \sin \alpha_{i-1} & \cos \alpha_{i-1} & d_i \cos \alpha_{i-1} \\ 0 & 0 & 0 & 1 \end{bmatrix}. \quad (\text{A.4})$$

For the KUKA manipulator KR 210 R2700 extra, Figure A.2 displays the symbolic structure of the robot using modified Denavit-Hartenberg notation. The Denavit-Hartenberg parameters are then represented in Table A.1. In order to simplify the model, the end-effector was neglected.

Table A.1: Denavit-Hartenberg parameters for a KR 210 R2700 extra.

Link i	θ_i (rad)	α_{i-1} (rad)	a_{i-1} (m)	d_i (m)
1	θ_1	0	0	$d_1=0.675$
2	$\theta_2 + \frac{\pi}{2}$	$\frac{\pi}{2}$	$a_1=0.350$	0
3	θ_3	0	$a_2=1.150$	0
4	θ_4	$\frac{\pi}{2}$	$-a_3=-0.041$	$d_4=1.200$
5	θ_5	$-\frac{\pi}{2}$	0	0
6	θ_6	$\frac{\pi}{2}$	0	$d_6=0.215$

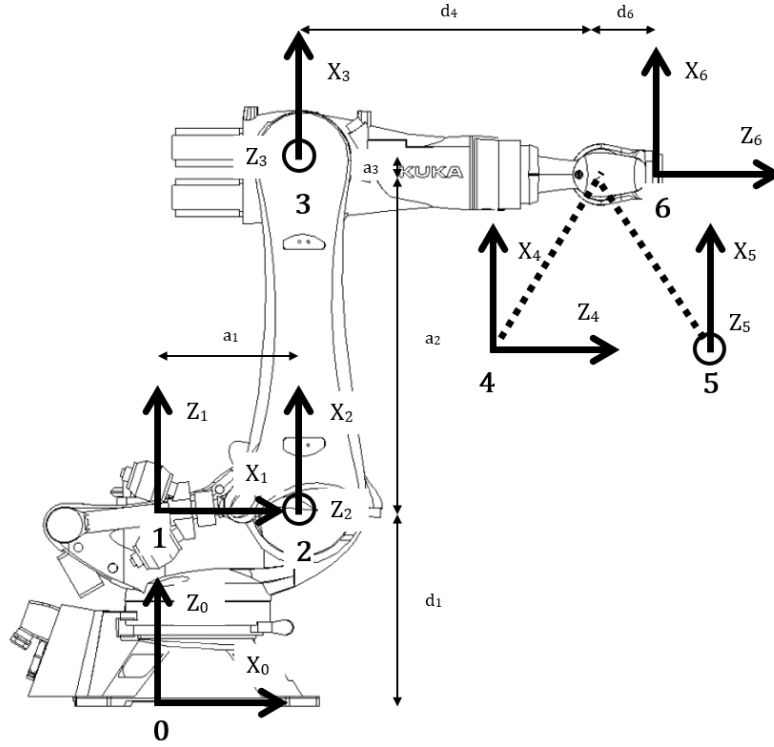


Figure A.2: KUKA KR 210 R2700 extra Denavit-Hartenberg frames.

A.2. Robotic kinematics models

The relationship between end-effector pose and joints position can be described using kinematics or dynamics models. Kinematics models describe the motion of rigid bodies and its higher-order time derivatives based on transformations between two different coordinates frames. In other words, kinematics models state how the motion occurs without taking into account what creates the motion. On the other hand, dynamics models describe the motion of rigid bodies based on the applied forces and torques. The additional complexity in the latter model allows the possibility to incorporate force control into the feedback loop of the robotic system, adding another option for control strategies.

Currently, industrial robot manipulator's tasks are narrowed to follow simple, plain and slow predefined paths within an unchanged and enclosed environment. The absence of quick robot interactions explains why robot dynamics are generally not considered when planning for industrial applications [89]. Hence, the remaining of this study will only focus on kinematics models.

Inspired by Pires [34] and Siciliano et al. [87] textbooks, in the following subsections will be formulated four kinematics models used to relate both joint space and operational space coordinates in robotic structures.

A.2.1. Forward Kinematics

A forward kinematics (or direct kinematics) model computes the position and orientation of the end-effector from a given joints position, $\mathbf{p} = \mathbf{p}(\mathbf{q})$. The mapping between both operational and joint space coordinates is established by assigning a coordinate frame to each link and performing the geometric analysis of the robot manipulator proposed in last section.

The homogeneous transformation matrix T_6^0 contains all the information needed to tackle a forward kinematics problem. The cartesian position of the end-effector \mathbf{p}_{xyz} can be directly obtained from the matrix T_6^0 , or by the equation A.5. Note that if the end-effector position is expressed in other coordinate system than the cartesian coordinate system, such as cylindrical or spherical coordinates system, the term \mathbf{p}_{xyz} must be multiplied by the respective coordinate transformation.

$$\begin{bmatrix} \mathbf{p}_{xyz}(\mathbf{q}) \\ 1 \end{bmatrix} = T_6^0(\mathbf{q}) \begin{bmatrix} 0_{3 \times 1} \\ 1 \end{bmatrix}, \quad \text{with} \quad T_6^0 = \begin{bmatrix} R_6^0(\mathbf{q}) & \mathbf{p}_{xyz}(\mathbf{q}) \\ \mathbf{0}_{1 \times 3} & 1 \end{bmatrix}. \quad (\text{A.5})$$

The end-effector orientation is described by the rotation matrix \mathbf{R}_6^0 , in which r_{ij} represents its entry in the i th row and j th column. Similar to the position of the end-effector, its orientation will also depend on the choice of coordinate system.

$$\mathbf{R}_6^0 = \prod_{i=1}^6 \mathbf{R}_i^{i-1} = \begin{bmatrix} r_{11} & r_{12} & r_{13} \\ r_{21} & r_{22} & r_{23} \\ r_{31} & r_{32} & r_{33} \end{bmatrix}. \quad (\text{A.6})$$

In the minimal representation, a set of three independent angles are sufficient to describe the orientation of the end-effector. Assuming the Euler angles α , β and γ as a rotation around the x -, y - and z -axes of the base frame in a counterclockwise direction respectively, \mathbf{R}_6^0 , also known as roll-pitch-yaw (XYZ) rotation matrix, is given by the next expression.

$$\mathbf{R}_6^0 = \begin{bmatrix} \cos \beta \cos \gamma & -\cos \beta \sin \gamma & \sin \beta \\ \cos \alpha \sin \gamma + \cos \gamma \sin \alpha \sin \beta & \cos \alpha \cos \gamma - \sin \alpha \sin \beta \sin \gamma & -\cos \beta \sin \alpha \\ \sin \alpha \sin \gamma - \cos \gamma \cos \alpha \sin \beta & \sin \alpha \cos \gamma + \cos \alpha \sin \beta \sin \gamma & \cos \beta \cos \alpha \end{bmatrix}. \quad (\text{A.7})$$

The angles α , β , γ can then be expressed as:

$$\beta = \text{atan2}(r_{13}, \sqrt{r_{23}^2 + r_{33}^2}). \quad (\text{A.8})$$

$$\alpha = \begin{cases} 0 & \text{if } \beta = \frac{\pi}{2} \text{ and } -\frac{\pi}{2}, \\ \text{atan2}\left(-\frac{r_{23}}{\cos \beta}, \frac{r_{33}}{\cos \beta}\right) & \text{otherwise.} \end{cases} \quad (\text{A.9})$$

$$\gamma = \begin{cases} \text{atan2}(r_{32}, r_{22}) & \text{if } \beta = \frac{\pi}{2}, \\ -\text{atan2}(r_{32}, r_{22}) & \text{if } \beta = -\frac{\pi}{2}, \\ \text{atan2}\left(-\frac{r_{12}}{\cos \beta}, \frac{r_{11}}{\cos \beta}\right) & \text{otherwise.} \end{cases} \quad (\text{A.10})$$

The function $\text{atan2}(y, x)$, known as the 2-argument arctangent or four-quadrant inverse tangent, is quite used in the field of control theory and returns a single angle in the closed interval $[-\pi, \pi]$ rad, based on the y and x values. One method to compute atan2 is presented in the equation A.11.

$$\text{atan2}(y, x) = \begin{cases} \arctan\left(\frac{y}{x}\right) & \text{if } x > 0, \\ \arctan\left(\frac{y}{x}\right) + \pi & \text{if } x < 0 \text{ and } y \geq 0, \\ \arctan\left(\frac{y}{x}\right) - \pi & \text{if } x < 0 \text{ and } y < 0, \\ +\frac{\pi}{2} & \text{if } x = 0 \text{ and } y > 0, \\ -\frac{\pi}{2} & \text{if } x = 0 \text{ and } y < 0, \\ \text{undefined} & \text{if } x = 0 \text{ and } y = 0. \end{cases} \quad (\text{A.11})$$

A.2.2. Inverse Kinematics

Contrary to forward kinematics, an inverse kinematics model finds the joints position required to perform a given end-effector pose, $\mathbf{q} = \mathbf{q}(\mathbf{p})$. Inverse kinematics models are clearly more appealing for industrial applications where an exact tool trajectory needs to be followed. However, since tool coordinates may not explicitly represent an unique robot configuration, multiple or even no admissible solutions increase the complexity of this model. No admissible solutions include valid mathematical solutions that are infeasible to the mechanical structure of the robot, for example requesting an end-effector pose that is out of reach or implies collision with the robot itself.

In the case of a six degrees-of-freedom robot manipulator, the inverse model can be solved by decoupling the end-effector position with its orientation. This assumption is only possible when three consecutive joint axes intersect at a common point [86], requiring to solve only one of those three joints position to obtain the other two. For the KUKA robot presented in Figure A.2, that condition is guaranteed by its spherical wrist, where joints angles $\theta_4, \theta_5, \theta_6$ are responsible for adjusting the orientation of the end-effector. The remaining three joints angles ($\theta_1, \theta_2, \theta_3$) are then used to modify the end-effector position.

The first step in this model is to find the position of the wrist, $\mathbf{p}_w = [p_{wx} \ p_{wy} \ p_{wz}]^T$. Since the end-effector pose is known, the homogeneous transformation matrix \mathbf{T}_6^0 can be assembled. For a robot with the

same Denavit-Hartenberg parameters as in Table A.1, where d_6 represents the distance from the wrist to the end-effector, the wrist position is given by the following transformation.

$$\begin{bmatrix} p_{wx} \\ p_{wy} \\ p_{wz} \\ 1 \end{bmatrix} = T_6^0 \begin{bmatrix} 0 \\ 0 \\ -d_6 \\ 1 \end{bmatrix} \Leftrightarrow \mathbf{p}_w = \mathbf{p}_{xyz} + \mathbf{R}_6^0 \begin{bmatrix} 0 \\ 0 \\ -d_6 \end{bmatrix}. \quad (\text{A.12})$$

Knowing the wrist position, it is now trivial to calculate the out-of-plane rotation θ_1 by simple trigonometry.

$$\theta_1 = \begin{cases} \text{atan2}(p_{wy}, p_{wx}), \\ \text{atan2}(-p_{wy}, -p_{wx}). \end{cases} \quad (\text{A.13})$$

From this point onwards, the robot can now be assessed as a 2D structure. With the first joint addressed, the model proceeds to evaluate the position of the wrist relative to the second joint. By considering l_x and l_z as the distance of the second joint to the wrist on the x and z -axes respectively, their values are calculated using:

$$l_x = \sqrt{p_{wx}^2 + p_{wy}^2} - a_1 \quad \text{and} \quad l_z = p_{wz} - d_1. \quad (\text{A.14})$$

In Figure A.3, a geometrical representation of these distances can be visualized. Another approach to calculate these values consists of making a geometrical analysis of the manipulator using θ_2 and θ_3 as variables. Assuming $a_x = \sqrt{d_4^2 + a_3^2}$ and $\theta_{3'} = \theta_3 + \arctan(a_3/d_4)$ as the distance from the third joint to the wrist and the respective angle that makes with its joint coordinate frame.

$$l_x = -a_2 \sin \theta_2 + a_x \cos(\theta_2 + \theta_{3'}) \quad \text{and} \quad l_z = a_2 \cos \theta_2 + a_x \sin(\theta_2 + \theta_{3'}). \quad (\text{A.15})$$

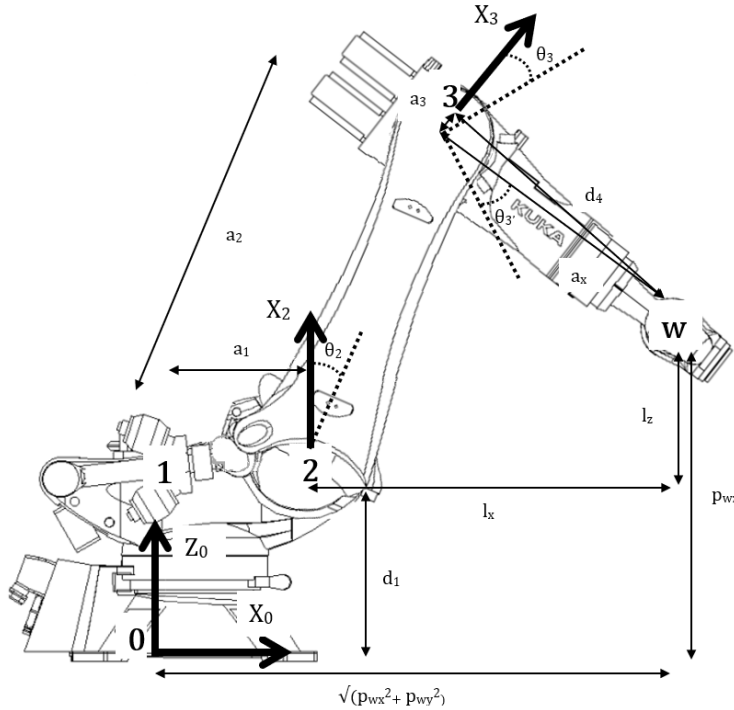


Figure A.3: Visual representation of the distance between the joint A2 and wrist.

By squaring and summing equations A.14 and A.15, an expression only dependent on $\theta_{3'}$ can be reached.

$$\sin \theta_{3'} = \frac{l_x^2 + l_z^2 - a_2^2 - a_x^2}{2a_2 a_x}. \quad (\text{A.16})$$

In order to get an angle θ_3 in the interval $[-\pi, \pi]$, equation A.16 can be rewritten as

$$\theta_3 = \text{atan2}(\sin \theta_{3'}, \pm \sqrt{1 - \sin^2 \theta_{3'}}) - \arctan \frac{a_3}{d_4}. \quad (\text{A.17})$$

By replacing the now known θ_3 in A.15, an expression for θ_2 is obtained.

$$\theta_2 = \text{atan2}(\sin \theta_2, \cos \theta_2), \quad \text{with} \quad (\text{A.18})$$

$$\sin \theta_2 = \frac{-(a_2 + a_x \sin \theta_{3'})l_x + a_x \cos \theta_{3'}l_z}{a_2^2 + a_x^2 + 2a_2 a_x \sin \theta_{3'}} \quad \text{and} \quad \cos \theta_2 = \frac{(a_2 + a_x \sin \theta_{3'})l_z + a_x \cos \theta_{3'}l_x}{a_2^2 + a_x^2 + 2a_2 a_x \sin \theta_{3'}}.$$

Since each θ_1 and θ_3 has two possible solutions, a total four robot configurations have the same position of the end-effector.

The first three joint variables are now known, yet it still remains to calculate the remaining joint angles associated to the tool orientation. The most common method to compute such angles consists of separating the rotation matrix \mathbf{R}_6^0 into two. The rotation matrix from the base to the wrist, \mathbf{R}_3^0 , can be calculated by the following matrix.

$$\mathbf{R}_3^0 = \begin{bmatrix} -\cos \theta_1 \sin(\theta_2 + \theta_3) & -\cos \theta_1 \cos(\theta_2 + \theta_3) & \sin \theta_1 \\ -\sin \theta_1 \sin(\theta_2 + \theta_3) & -\sin \theta_1 \cos(\theta_2 + \theta_3) & -\cos \theta_1 \\ \cos(\theta_2 + \theta_3) & -\sin(\theta_2 + \theta_3) & 0 \end{bmatrix}, \quad (\text{A.19})$$

By equation A.6, the rotation matrix from the wrist to the end-effector, \mathbf{R}_6^3 , is given by

$$\mathbf{R}_6^3 = (\mathbf{R}_3^0)^T \mathbf{R}_6^0 = \begin{bmatrix} b_{11} & b_{12} & b_{13} \\ b_{21} & b_{22} & b_{23} \\ b_{31} & b_{32} & b_{33} \end{bmatrix}, \quad (\text{A.20})$$

where b_{ij} represents \mathbf{R}_6^3 entry in the i th row and j th column. Assuming a spherical wrist (Figure 2.3), \mathbf{R}_6^3 is described by a ZYZ rotation matrix (or sometimes mentioned as roll-pitch-roll rotation matrix, in which first occurs a rotation about the forearm, then a pitch about the wrist center and lastly a rotation about the wrist center).

$$\mathbf{R}_6^3 = \begin{bmatrix} \cos \theta_4 \cos \theta_5 \cos \theta_6 - \sin \theta_4 \sin \theta_6 & -\cos \theta_4 \cos \theta_5 \sin \theta_6 - \sin \theta_4 \cos \theta_6 & \cos \theta_4 \sin \theta_5 \\ \sin \theta_5 \cos \theta_6 & -\sin \theta_5 \sin \theta_6 & -\cos \theta_5 \\ \sin \theta_4 \cos \theta_5 \cos \theta_6 + \cos \theta_4 \sin \theta_6 & -\sin \theta_4 \cos \theta_5 \sin \theta_6 + \cos \theta_4 \cos \theta_6 & \sin \theta_4 \sin \theta_5 \end{bmatrix}. \quad (\text{A.21})$$

Since the solutions for ZYZ rotation angles are well known [34], the values of θ_4 , θ_5 and θ_6 are then given by the following equations.

$$\theta_5 = \text{atan2}(\pm \sqrt{b_{13}^2 + b_{33}^2}, \mp b_{23}). \quad (\text{A.22})$$

$$\theta_4 = \begin{cases} \text{atan2}(b_{33}, b_{13}) & \text{if } \theta_5 \in [0, \pi], \\ \text{atan2}(-b_{33}, -b_{13}) & \text{if } \theta_5 \in [-\pi, 0]. \end{cases} \quad (\text{A.23})$$

$$\theta_6 = \begin{cases} \text{atan2}(-b_{22}, b_{21}), & \text{if } \theta_5 \in [0, \pi], \\ \text{atan2}(b_{22}, -b_{21}), & \text{if } \theta_5 \in [-\pi, 0]. \end{cases} \quad (\text{A.24})$$

With two possible solutions for θ_5 , but only one solution for θ_4 and θ_6 , the inverse orientation kinematics problem has two solutions. By multiplying with the four solutions previously obtained in the position problem, the inverse kinematics model for a robot kinematics will obtain a total of eight solutions. Therefore, inverse kinematics model is not ideal to use standalone, it always requires forward or backward knowledge in order to choose which is the most suitable solution.



Figure A.4: Eight possible robot configuration for the same end-effector pose. Cited in [10], p. 40.

A.2.3. Forward Differential Kinematics

Differential models (or also known as instantaneous models) establish a relationship between the velocity of the end-effector pose with the manipulator joints velocity. In particular, a forward differential kinematics model computes the velocity of the end-effector as a function of the velocity of all joints $\dot{\mathbf{p}} = \dot{\mathbf{p}}(\dot{\mathbf{q}})$. This mapping is described by a single matrix named Jacobian, $\mathbf{J}(\mathbf{q})$. Bear in mind that the velocity of the end-effector includes both linear and angular components of its frame relative to the world fixed frame. Also note that for a revolute joint, its velocity is made of a single angular velocity component.

By linearizing the forward kinematics expression $\mathbf{p} = \mathbf{p}(\mathbf{q})$ and neglecting higher-order derivatives, the relation between joints and end-effector velocities is given by the so-called analytical Jacobian matrix, $\mathbf{J}_A(\mathbf{q})$, of dimension $m \times n$, where m is the number of end-effector coordinates and n is the number of joints. This relation can be written as

$$\dot{\mathbf{p}} = \mathbf{J}_A(\mathbf{q})\dot{\mathbf{q}}, \quad \text{with} \quad \mathbf{J}_A(\mathbf{q}) = \begin{bmatrix} \frac{\partial \mathbf{p}_{xyz}}{\partial \mathbf{q}} \\ \frac{\partial \mathbf{p}_{a\beta\gamma}}{\partial \mathbf{q}} \end{bmatrix}, \quad (\text{A.25})$$

where $\dot{\mathbf{p}}$ is a m -dimensional vector composed of the spatial velocity of the end-effector and $\dot{\mathbf{q}}$ is a n -dimensional vector representing the joints velocity.

The analytical Jacobian matrix strongly depends on the selected coordinate systems for the end-effector pose. In fact, direct differentiation of forward kinematics expression may not be interpreted as the linear and angular velocities of the end-effector. For most cases, as its position is characterized by cartesian coordinates x , y , and z , the linear velocity of the end-effector can effectively be calculated by the analytical Jacobian. However, this property especially affects the interpretation of its angular velocity. For instance, in the minimal

representation, the Euler angles do not characterize a vector in a cartesian space, hence differentiation of expressions A.8, A.9 and A.10 does not represent the angular velocity of the tool in relation to the base frame. For this reason, the analytical Jacobian is not frequently adopted by robotic programmers, only in cases where orientation is subject to task-specific constraints.

An alternative method is to evaluate the geometric relationship between each joint coordinate frame relative to the base frame. Assuming V as the end-effector velocity vector containing the linear cartesian velocity $\dot{\mathbf{p}}_{xyz}$ and angular cartesian velocity \mathbf{w} , the relation between joints velocity and cartesian linear and angular velocities is given by the denominated geometric (or basic) Jacobian matrix, $J(\mathbf{q})$. Unlike the analytical Jacobian, this matrix has a fixed number of lines, since the end-effector velocity has now only six independent components, described by two 3D vector in a cartesian space. The geometric Jacobian is given by

$$\mathbf{V} = J(\mathbf{q})\dot{\mathbf{q}}, \quad \text{with} \quad \mathbf{V} = \begin{bmatrix} \dot{\mathbf{p}}_{xyz} \\ \mathbf{w} \end{bmatrix}, \quad (\text{A.26})$$

and can be computed by summing all velocity contributions of each joint to the end-effector linear and angular velocities, resulting in

$$J(\mathbf{q}) = \begin{bmatrix} \mathbf{t}_1 \times \mathbf{r}_{n+1}^1 & \dots & \mathbf{t}_n \times \mathbf{r}_{n+1}^n \\ \mathbf{t}_1 & \dots & \mathbf{t}_n \end{bmatrix}, \quad (\text{A.27})$$

where \mathbf{r}_j^i is the position of coordinate frame of i th link in relation to the coordinate frame of j th link and \mathbf{t}_i is the unit vector in which the i th link rotates in relation to the base frame.

Contrary to the previous methods presented, solving kinematics models with velocities exploits instantaneous linear equations rather than non-linear equations, which are easier to set up and model according to the task's priority. Analyzing singularities and adding redundant axes, which will be described later, increase the complexity of the model and are therefore solved mainly by means of differential kinematics. Yet, these methods assume that the robot configuration is completely known in advance, requiring to solve either a forward or inverse kinematics problem beforehand.

In the context of control systems, the equation A.26 is usually replaced by an approximation expression that describe the infinitesimal position and orientation of the end-effector with the infinitesimal joints position. This numerical solution can be written as:

$$\Delta \mathbf{p} = J(\mathbf{q})\Delta \mathbf{q}. \quad (\text{A.28})$$

As a remark, large steady-state errors are expected through this linear approximation. For this reason, proportional or other more sophisticated controllers are usually integrated with the feedback control algorithm, minimizing not only the steady-state error but also the response time and overshooting.

A.2.4. Inverse Differential Kinematics

The goal of an inverse differential kinematics model is to determine the joints velocity when the velocity of the end-effector is known, $\dot{\mathbf{q}} = \dot{\mathbf{q}}(\mathbf{p})$. With the introduction of a Jacobian matrix in forward differential kinematics models, a linear mapping is achieved between the and operational space vectors. The convenience of a linear relation guided robots programmers to utilize a similar approach to tackle inverse kinematics problems. By rewriting equation A.26, the model can be described by the following expression.

$$\dot{\mathbf{q}} = J^{-1}(\mathbf{q})\mathbf{V}. \quad (\text{A.29})$$

Unfortunately, equation A.29 requires the Jacobian matrix to be square and of full rank. In practice, this requirement to the Jacobian matrix demands the manipulator to have six degrees-of-freedom in order to completely determine the end-effector velocity. For complex systems with a higher or lower number of joints, the inverse of the Jacobian matrix cannot be used to solve inverse differential kinematics. In section A.4 will be analyzed how to solve the inverse kinematics if the robotic system contains more degrees-of-freedom.

In analogy to forward kinematics models using the differential equations, it is possible to make a numerical implementation of equation A.29. The six joint velocities are then computed by multiplying the inverse of the Jacobian with the joint variables at the previous time instant, as given in the equation A.30. Additionally, controllers must also be added to reduce the static-error.

$$\Delta \mathbf{q} = J^{-1}(\mathbf{q})\Delta \mathbf{p}. \quad (\text{A.30})$$

A.3. Kinematic singularities

The term kinematic singularities is employed in the context of robotic manipulators to describe configurations \mathbf{q} in which the Jacobian matrix is rank-deficient. This momentary loss of rank visibly impacts the robot's mobility, as the end-effector is locally incapacitated to move along or rotate about some direction in cartesian space. For this reason, in a singular configuration, an arbitrary end-effector motion cannot be imposed by traditional forward differential kinematics. Additionally, in the neighbourhood of a singularity, large joint velocities may be required to produce small changes in the velocity of the end-effector. On the other side, if inverse differential kinematics models are used, infinite solutions will be obtained for the joints velocity vector.

A distinction between two types of singularities are presented in [87] and consist of:

- Boundary singularities occur when the robot's structure is either completely stretched or retracted. They do not present a great challenge as can be easily avoided by imposing a condition in which the manipulator avoids its workspace limits.
- Internal singularities appear when two or more joint axes are coincident, making impossible to determine the actual locations of those joints. The whole workspace can be susceptible to internal singularities, rendering it necessary to impose several constraints to the motion planning in order to avoid these singularities.

Finding all internal singularities in advance contributes for a better planning of the robot motion. One approach consists of computing the determinant of the whole Jacobian matrix and equalize it to zero, $\det \mathbf{J}(\mathbf{q}) = 0$. However, calculating the Jacobian determinant whenever the robot configuration changes can be computationally expensive. In many cases, these singularities separate regions in the joint space with distinct inverse kinematics solutions. Hence, by decomposing the Jacobian matrix, the calculation of singularities can be decoupled into two or three individual problems. For example, KUKA industrial manipulators may face three different singularities, as represented in Figure A.5.

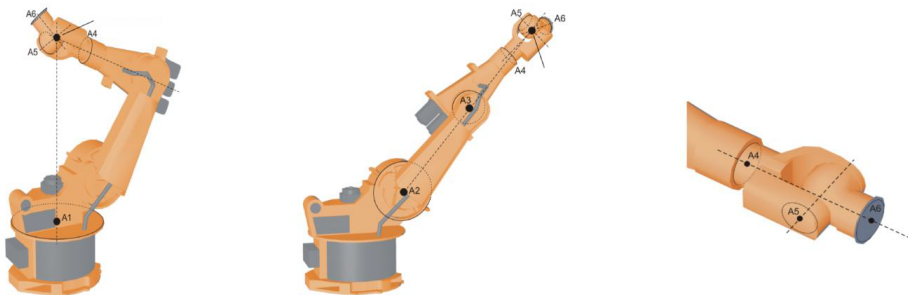


Figure A.5: Three types of singularities in a KUKA KR QUANTEC extra, from the left to the right, shoulder, elbow and wrist position singularity. Cited in [10], p.43.

To avoid chaotic movements of a manipulator near a singularity, industrial controllers often adopt strategies in which the robot velocity is reduced and a default or a previous position is assigned to the indeterminate joint. More advanced strategies can be integrated into motion planning algorithms generally by adding singularity constraints to their cost functions.

A.4. Kinematic redundancy

A robotic system is classified as kinematic redundant (or just redundant) when the number of degrees-of-freedom of its joint space is higher than the number of degrees-of-freedom necessary to accomplish its attributed task. Going back again to the forward differential expression $\mathbf{V} = \mathbf{J}(\mathbf{q})\dot{\mathbf{q}}$, from a mathematical point of view, what redundancy implies is that the robotic systems will always have more variables than equations (m lines $<$ n columns). Thus, this condition indicates the presence of a non-square Jacobian and, as a consequence, impossible to directly invert the matrix. Traditional inverse differential kinematics models cannot be applied in such a case.

By making use of redundant systems, a spectrum of advantages is provided that ultimately enhances robot performance, from increasing the reachable workspace, obstacle avoidance, to even minimizing execution

time and motion torques [90]. However, the potential benefits of redundancy should always be traded off against the complexity of the associated construction and the complicated algorithms required to compute the joint variables.

An important distinction between two types of redundant systems should be clarified. To facilitate their interpretation, it will be assumed that each joint adds a single degree-of-freedom to the system and that the operational space is fully characterized by six degrees-of-freedom (x, y, z and Euler angles).

- When the dimension of the joint space is higher than the dimension of the operational space, or using the above assumptions, when the number of joints is greater than six, the robot is said to be intrinsically redundant. For example, when adding a linear track or a mandrel to the system.
- When the dimension of the operational space is higher than the dimension of the task space, or for this case, when the number of degrees-of-freedom required for a task is smaller than six, the robot is denominated as functionally redundant. For example, for arc-welding, the rotation of the tool around its symmetry axis is irrelevant and can be removed.

In the literature, the resolution of redundant problems can be dated back to the 1970's [91] and methods have since been consistently improved, particularly given the interest of its applicability to industrial tasks. Most research has been focused on solving intrinsically redundant problems [91–94], with significant fewer dedicated to functional redundancy [95]. A brief explanation of the most standardized methods will be described in the following paragraphs, inspired by an exhaustive overview written by Siciliano [96].

In 1969, Whitney [91] proposed a method that instantaneously became a pillar framework for most of the subsequent developments. In his approach, the inverse of the Jacobian matrix was replaced by its Moore-Penrose pseudoinverse. The Whitney's inverse kinematics model was then governed by the equation

$$\dot{\mathbf{q}} = \mathbf{J}^+ \mathbf{V}, \quad \text{with} \quad \mathbf{J}^+ = \mathbf{J}^T (\mathbf{J} \mathbf{J}^T)^{-1}. \quad (\text{A.31})$$

Operating with the pseudoinverse made it possible to invert the Jacobian without sacrificing the relationship between joints and end-effector velocities. Another advantage is its least squares property, that is to say that the pseudoinverse generates joints velocity vectors with the minimum norm achievable. However, Whitney's method did not consider singularities and repeatability of the motion is not preserved. Though computationally simpler, these two disadvantages inhibit the application of his method to industrial robots.

Wampler [92] proposed the damped least square method, in which a damping factor λ was introduced to induce a robust manipulator behavior when crossing singularities. The expression for this model is presented in A.32, where \mathbf{I} is the $m \times m$ identity matrix.

$$\dot{\mathbf{q}} = \mathbf{J}^* \mathbf{V}, \quad \text{with} \quad \mathbf{J}^* = \mathbf{J}^T (\mathbf{J} \mathbf{J}^T + \lambda \mathbf{I})^{-1}. \quad (\text{A.32})$$

A clear trade-off between path precision against feasibility close to singular configurations can be visualized. When using large values of λ , a suitable robot motion is obtained, yet compromising the task accuracy. On the other hand, low values of λ result in unpredictable robot movements. Finding the right estimate for λ is a difficult task and for this reason repeatability of the motion is hardly guaranteed.

Introduced in [93] and further improved in [94], the gradient projection method embraced the concept of null space motion. When the Jacobian matrix is multiplied by a joints velocity and returns a zero end-effector velocity, this vector is considered as part of the null space of the Jacobian matrix. Going back to the inverse kinematics expression, what the latter sentence implies is that there are joints velocity vectors that only produce joint self-motion while the end-effector remains stationary. It is therefore possible to add to equation A.31 a homogeneous term that reconfigures the manipulator into a more dexterous posture without affecting the end-effector velocity, resulting in the following expression.

$$\dot{\mathbf{q}} = \mathbf{J}^+ \mathbf{V} + (\mathbf{I} - \mathbf{J}^+ \mathbf{J}) \dot{\mathbf{q}}_0, \quad (\text{A.33})$$

where \mathbf{I} is the $n \times n$ identity matrix. The homogeneous term $(\mathbf{I} - \mathbf{J}^+ \mathbf{J}) \dot{\mathbf{q}}_0$ is created by the projection of the desired joints velocity vector $\dot{\mathbf{q}}_0$ into the null space, where $\dot{\mathbf{q}}_0$ is computed as the gradient of a cost function.

All methods herein briefly outlined are currently well-established to deal with redundant problems. Recent efforts have been focusing on integrating redundancy into task-specific motion algorithms [10, 46, 95] and even developing what are known as hyper-redundant manipulators [90, 97]. In recent years, redundancy applied to complex paths was also analyzed using more advanced methods such as neural networks [98] or optimization based processes [10, 46, 99, 100].

B

Appendix B: Experimental Robot Motion

This appendix comprises the plots of all variables that characterize a KUKA manipulator motion when this is controlled externally with LayLa, for both Tow Course 1 and 2, with waypoints distributed at a rate of 5 Hz and a desired laydown speed of 0.1 m/s. Each of these plots contains three curves, consisting of the output of the time parameterization (joints request), the output of the position controller (joints command) and the actual robot states (joints state).

At the start, the six joint position trajectories and their corresponding velocities and accelerations are presented. The three first pages are divided in two columns (one column per joint) and three lines, with each line carrying the position, then the velocity and finally the acceleration of the respective joint. Afterwards, the end-effector pose is described using the KUKA nomenclature, three cartesian coordinates (X, Y, Z) and the Euler angles (A, B, C). As ROS operates with quaternions, the conversion from quaternions to Euler angles was applied to reach the KUKA variables. The figures are presented starting from the next page. Please note that, for a better interpretation, different scales were applied.

Observing all graphs, it can be concluded that the joint position trajectories were followed extremely closely, with the effects of quintic spline interpolation performed by the ROS controller being visible at their velocities and accelerations in the form of overshooting between the requested waypoints. The rounding errors created by the small number of decimal places stored by the RSI were clearly evident at joint A4 (initial joint of the wrist) for Tow Course 1, with the command specifying no motion yet a displacement of 0.0001° ($\approx 1.75 \mu\text{rad}$) detected. Given that joint A4 remained "constant" on both courses, practically no motion was detected at angles B and C. In these three variables, the scale in their plots was reduced to such an extent that the displacements between the requested and the actual state appear larger than they actually are, with differences being mostly created by numerical noises. As for acceleration, the appearance of white noise in all joints suggests an exacerbation of small errors. With the exception of joint A6, these perturbations let to higher accelerations than the actual acceleration requested. As a last note, discrepancies in the Z variable (vertical distance to the mold surface) were found with an order of magnitude of ± 0.01 mm, implying changes in the roller compression. Nonetheless, these differences should be dealt by the pneumatic cylinder usually included in the roller mechanism.

B.1. Tow Course 1

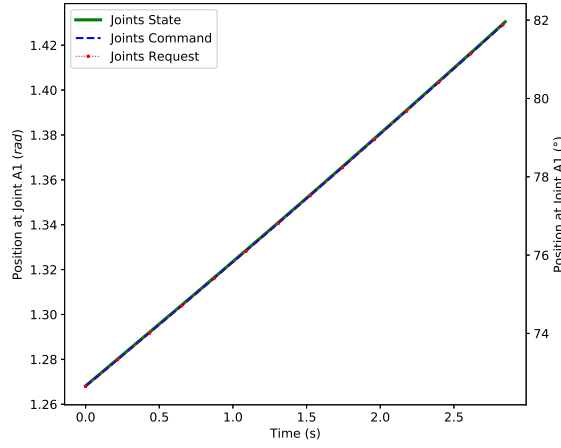


Figure B.1: Joint A1 trajectory for fitted Tow Course 1.

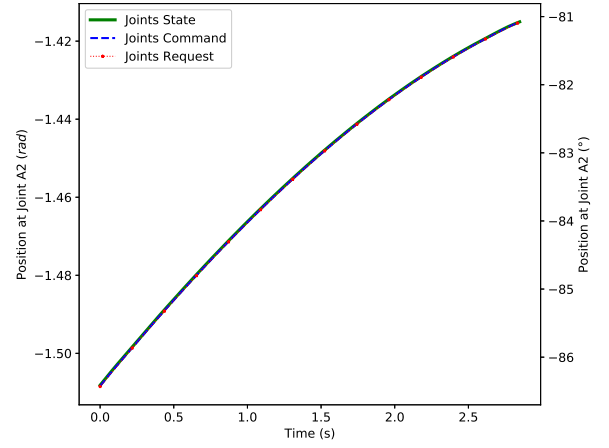


Figure B.2: Joint A2 trajectory for fitted Tow Course 1.

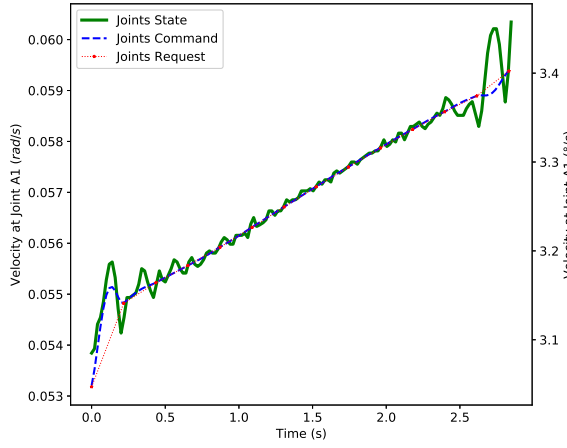


Figure B.3: Joint A1 velocity for fitted Tow Course 1.

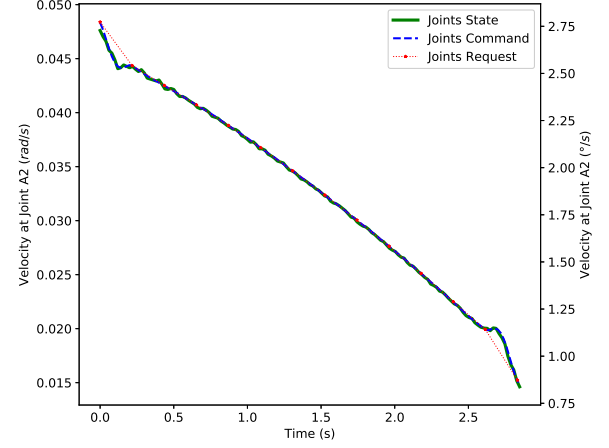


Figure B.4: Joint A2 velocity for fitted Tow Course 1.

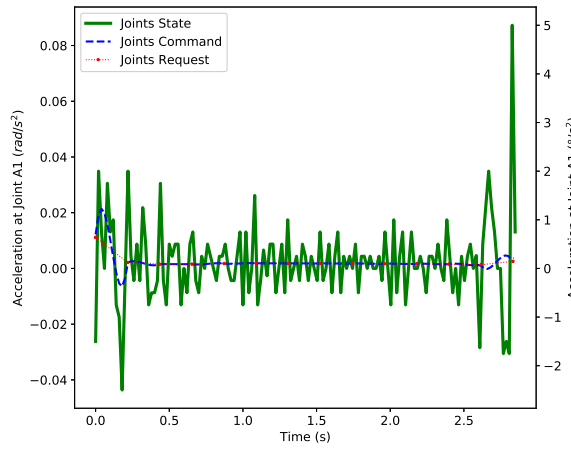


Figure B.5: Joint A1 acceleration for fitted Tow Course 1.

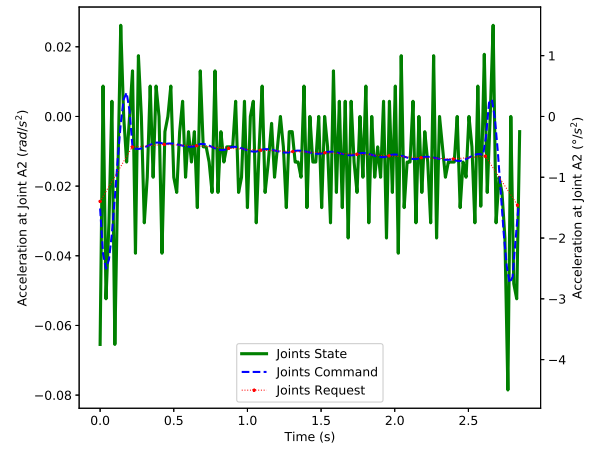


Figure B.6: Joint A2 acceleration for fitted Tow Course 1.

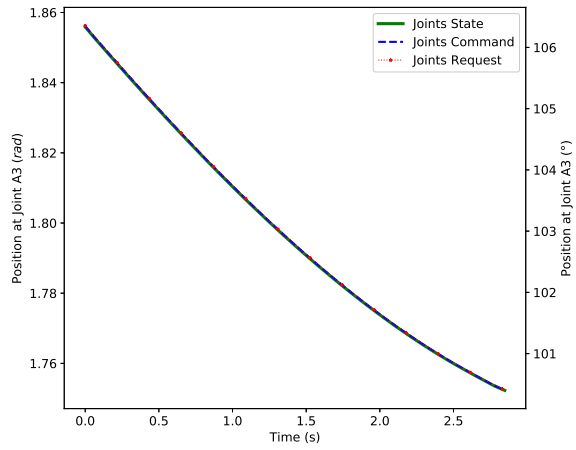


Figure B.7: Joint A3 trajectory for fitted Tow Course 1.

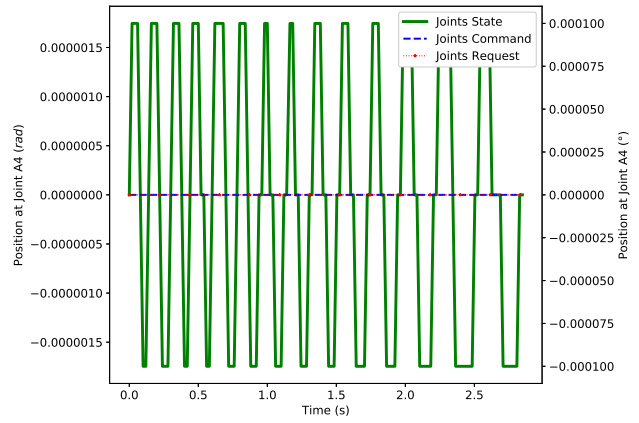


Figure B.8: Joint A4 trajectory for fitted Tow Course 1.

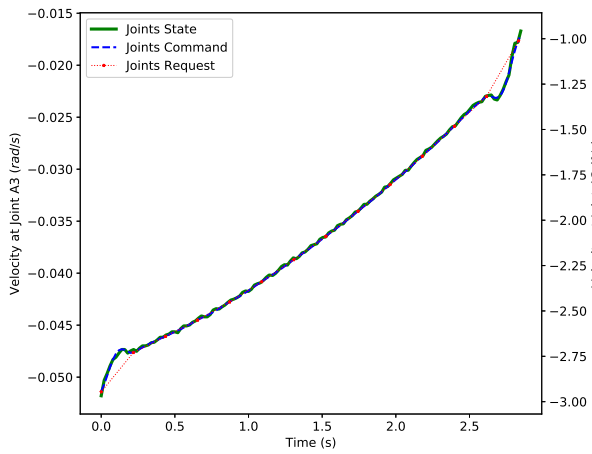


Figure B.9: Joint A3 velocity for fitted Tow Course 1.

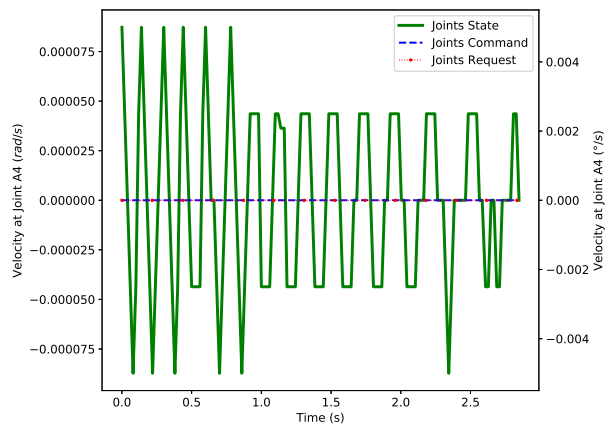


Figure B.10: Joint A4 velocity for fitted Tow Course 1.

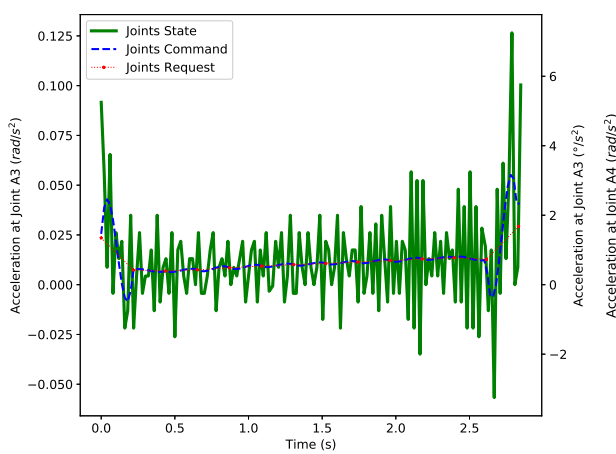


Figure B.11: Joint A3 acceleration for fitted Tow Course 1.

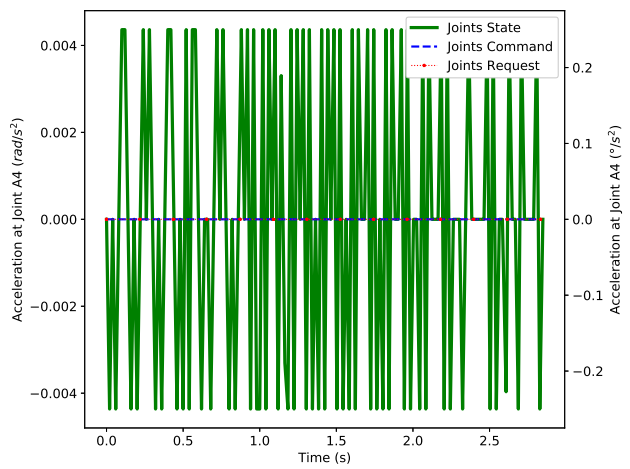


Figure B.12: Joint A4 acceleration for fitted Tow Course 1.

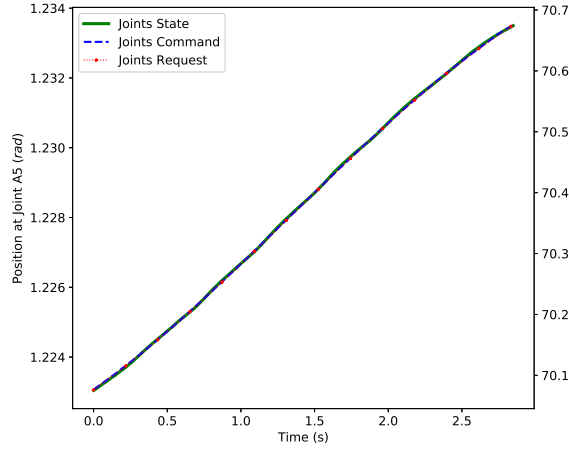


Figure B.13: Joint A5 trajectory for fitted Tow Course 1.

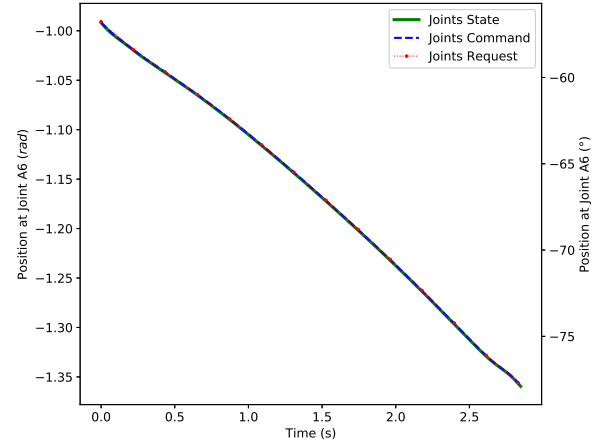


Figure B.14: Joint A6 trajectory for fitted Tow Course 1.

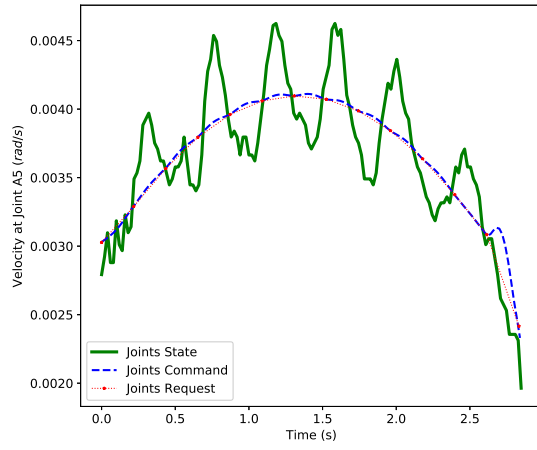


Figure B.15: Joint A5 velocity for fitted Tow Course 1.

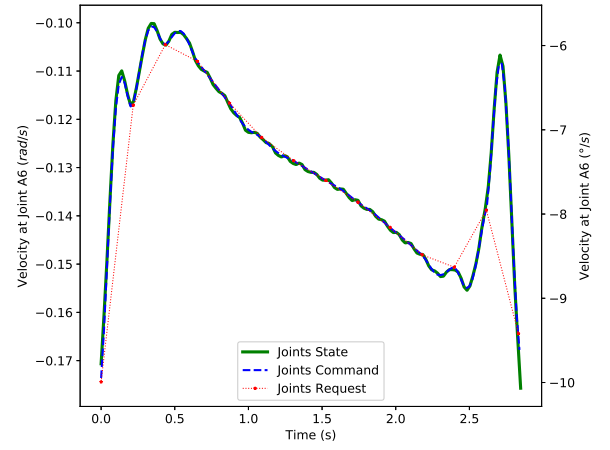


Figure B.16: Joint A6 velocity for fitted Tow Course 1.

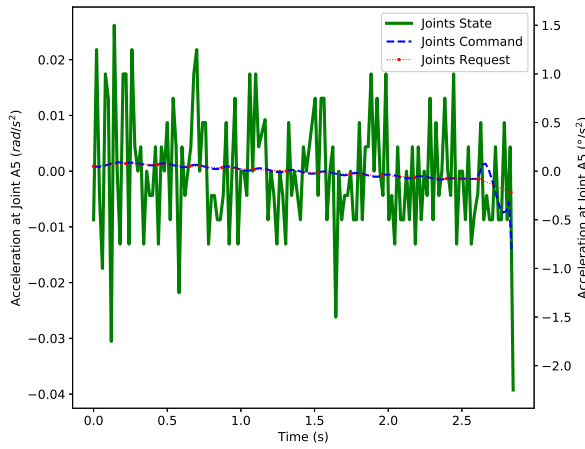


Figure B.17: Joint A5 acceleration for fitted Tow Course 1.

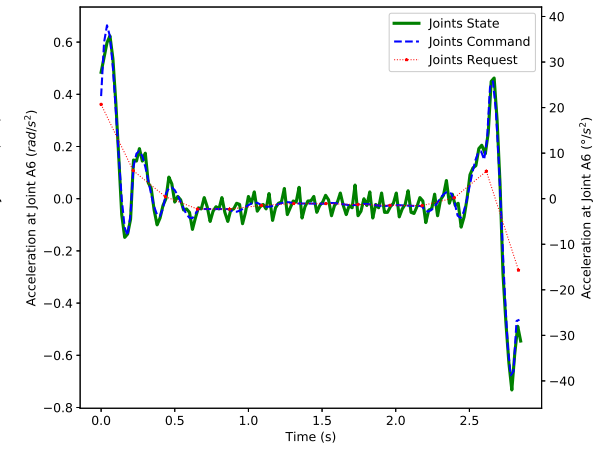


Figure B.18: Joint A6 acceleration for fitted Tow Course 1.

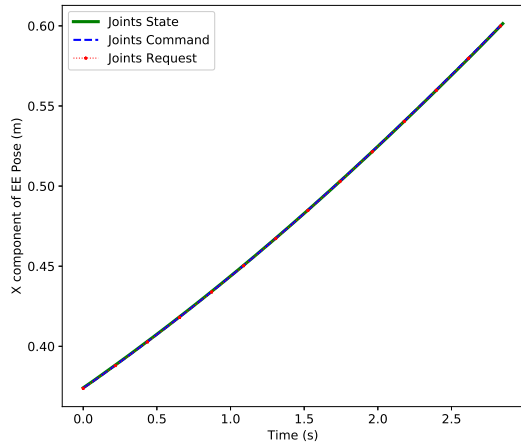


Figure B.19: X component of end-effector pose for fitted Tow Course 1.

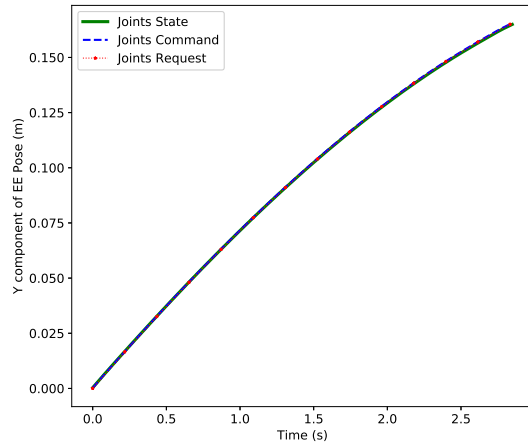


Figure B.20: Y component of end-effector pose for fitted Tow Course 1.

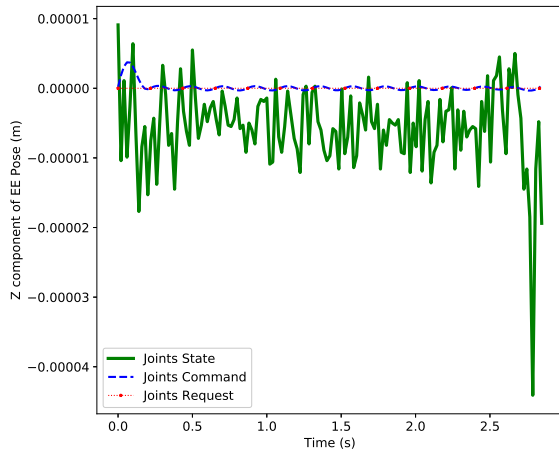


Figure B.21: Z component of end-effector pose for fitted Tow Course 1. The higher, the deeper it descends.

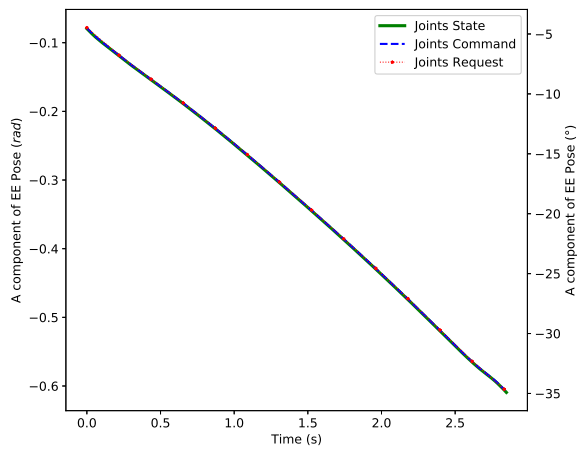


Figure B.22: A component (rotation about z-axis) for fitted Tow Course 1.

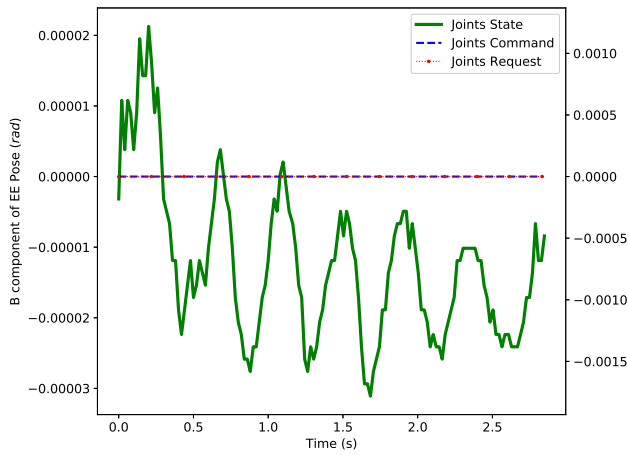


Figure B.23: B component (rotation about y-axis) for fitted Tow Course 1.

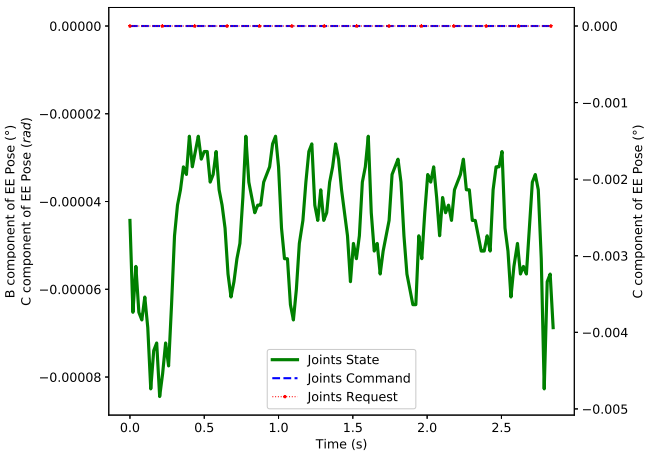


Figure B.24: C component (rotation about x-axis) for fitted Tow Course 1.

B.2. Tow Course 2

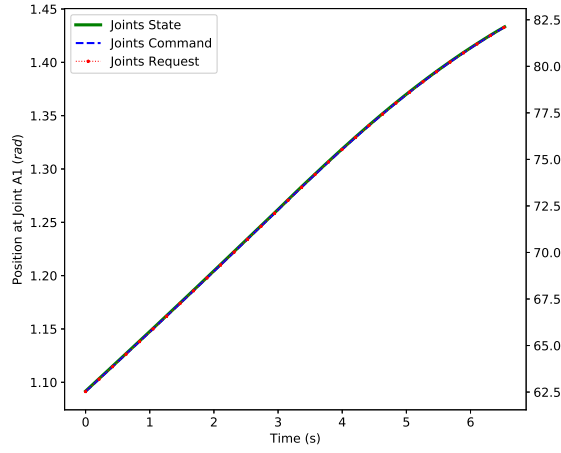


Figure B.25: Joint A1 trajectory for fitted Tow Course 2.

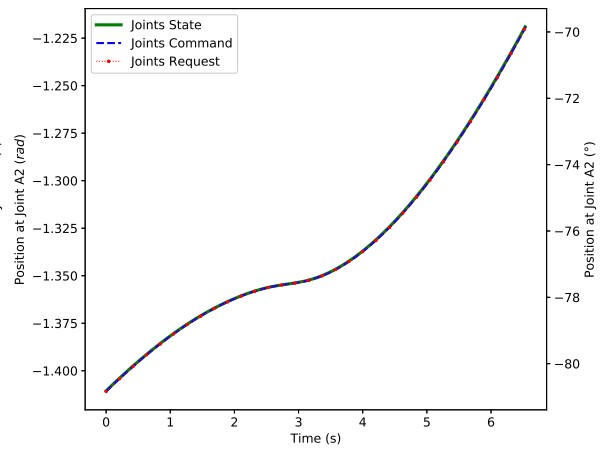


Figure B.26: Joint A2 trajectory for fitted Tow Course 2.

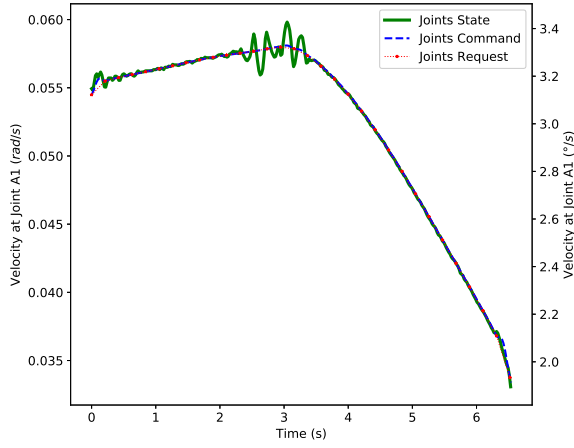


Figure B.27: Joint A1 velocity for fitted Tow Course 2.

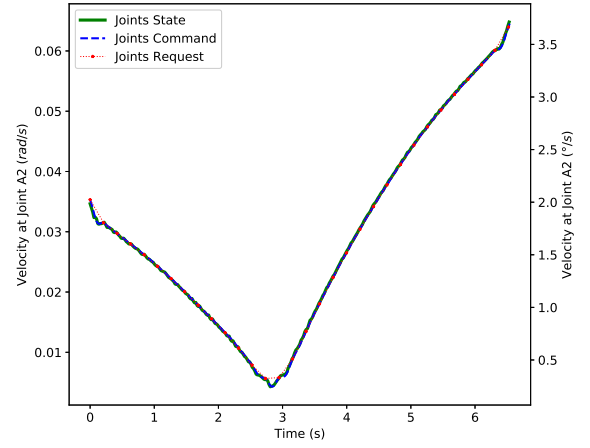


Figure B.28: Joint A2 velocity for fitted Tow Course 2.

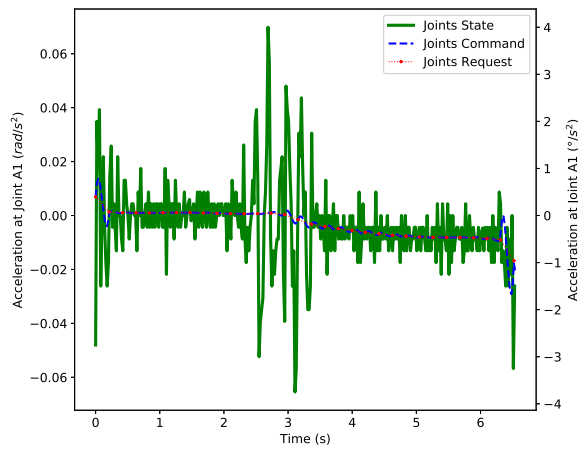


Figure B.29: Joint A1 acceleration for fitted Tow Course 2.

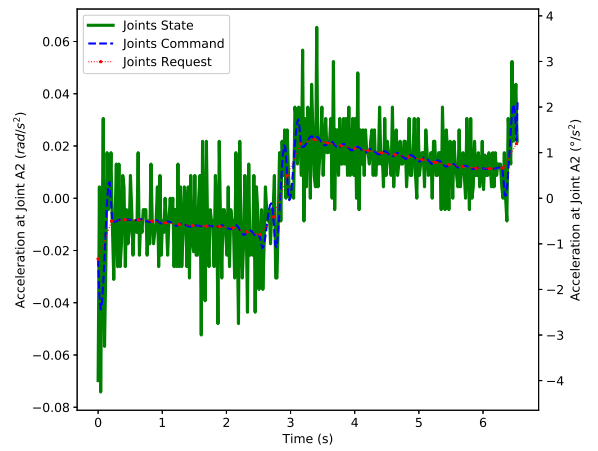


Figure B.30: Joint A2 acceleration for fitted Tow Course 2.

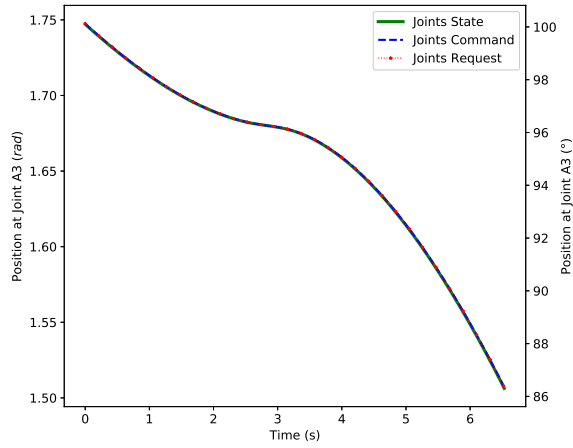


Figure B.31: Joint A3 trajectory for fitted Tow Course 2.

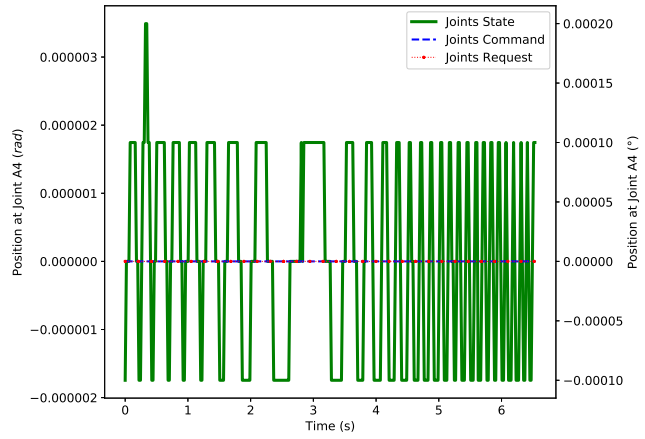


Figure B.32: Joint A4 trajectory for fitted Tow Course 2.

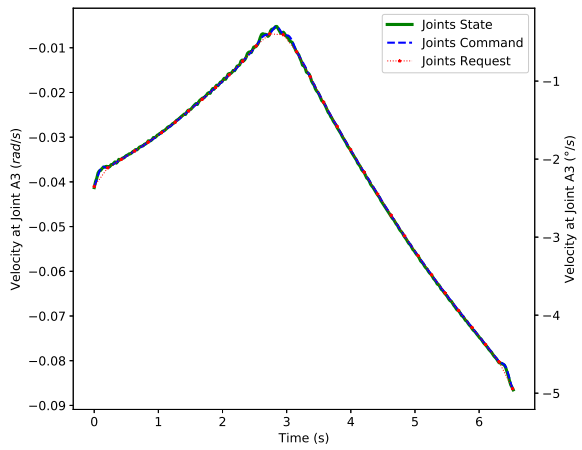


Figure B.33: Joint A3 velocity for fitted Tow Course 2.

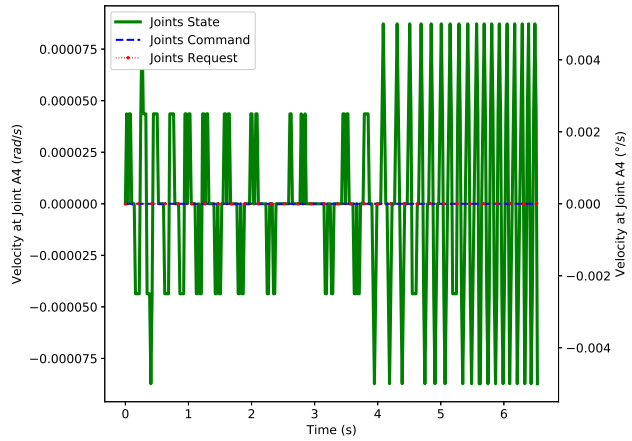


Figure B.34: Joint A4 velocity for fitted Tow Course 2.

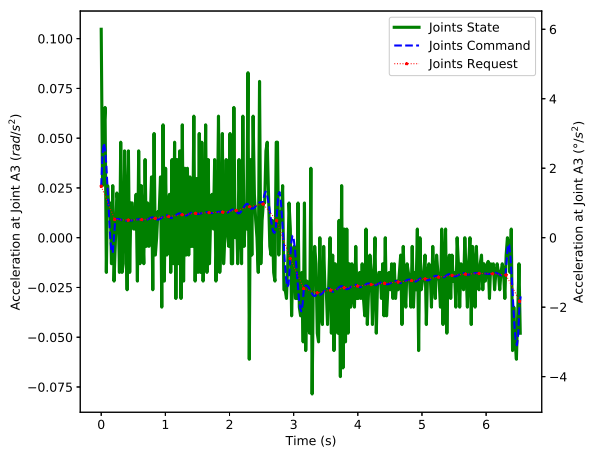


Figure B.35: Joint A3 acceleration for fitted Tow Course 2.

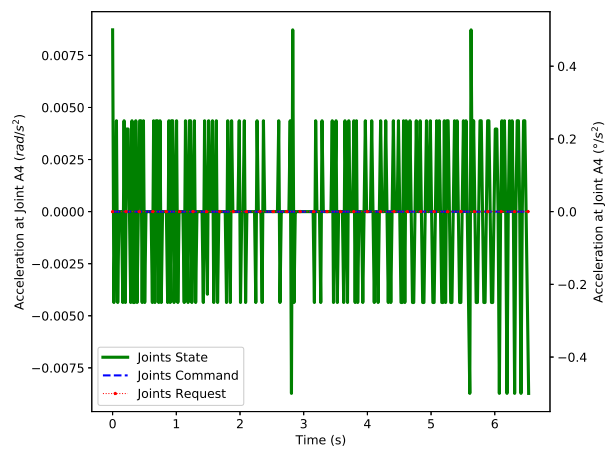


Figure B.36: Joint A4 acceleration for fitted Tow Course 2.

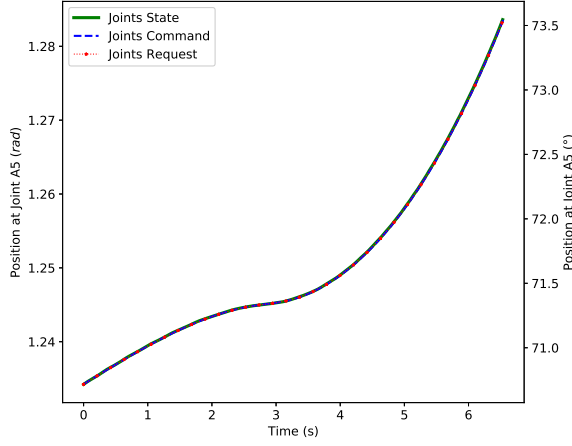


Figure B.37: Joint A5 trajectory for fitted Tow Course 2.

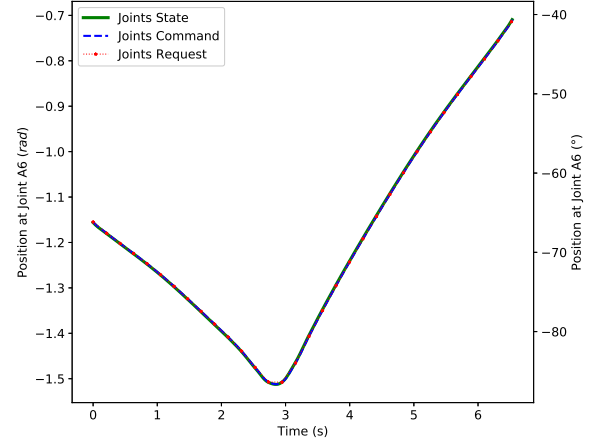


Figure B.38: Joint A6 trajectory for fitted Tow Course 2.

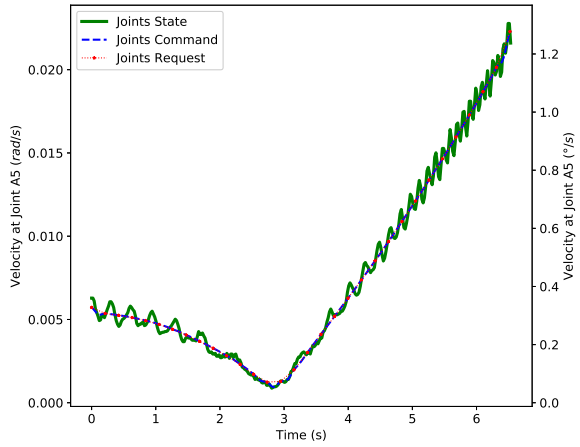


Figure B.39: Joint A5 velocity for fitted Tow Course 2.

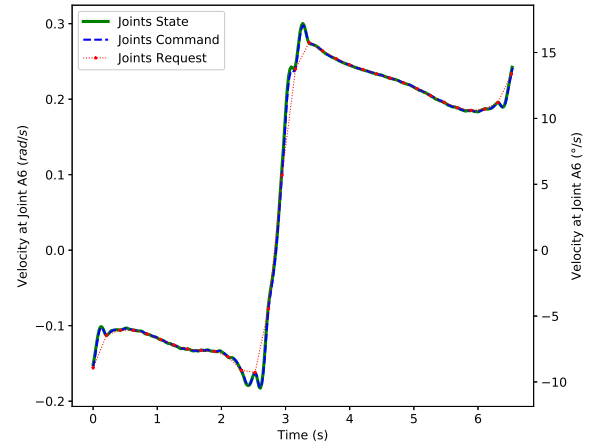


Figure B.40: Joint A6 velocity for fitted Tow Course 2.

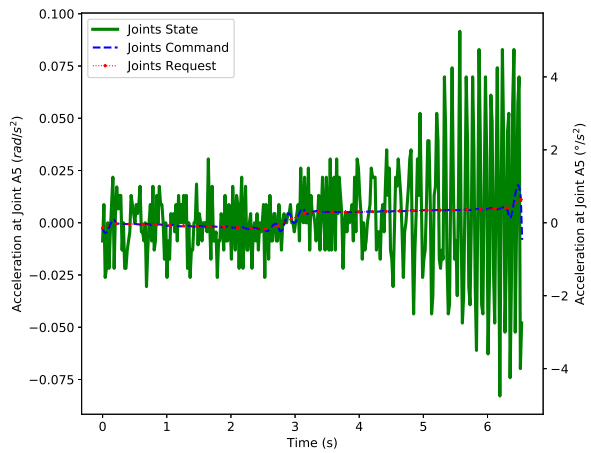


Figure B.41: Joint A5 acceleration for fitted Tow Course 2.

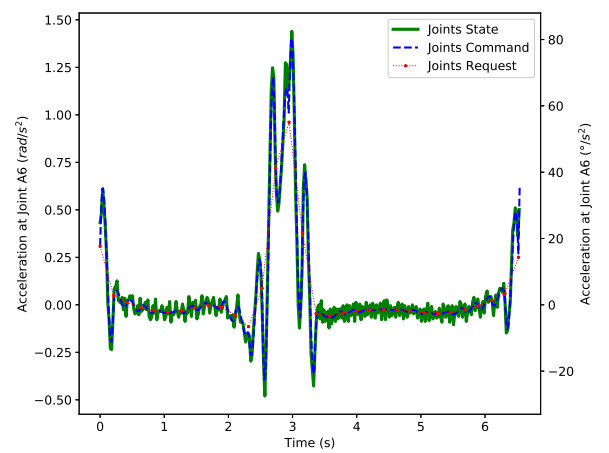


Figure B.42: Joint A6 acceleration for fitted Tow Course 2.

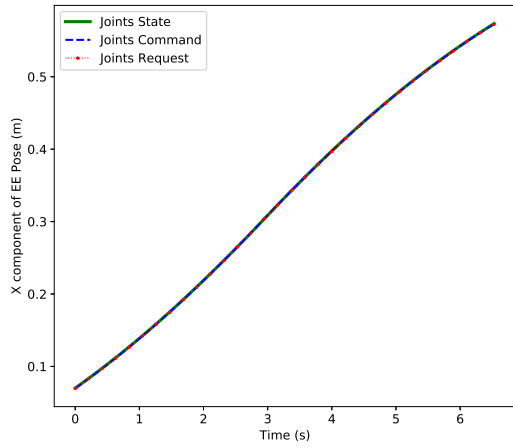


Figure B.43: X component of end-effector pose for fitted Tow Course 2.

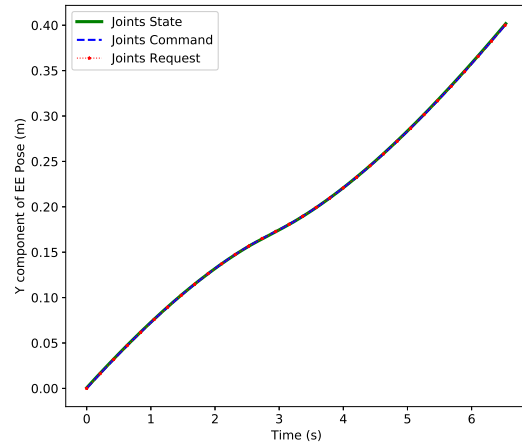


Figure B.44: Y component of end-effector pose for fitted Tow Course 2.

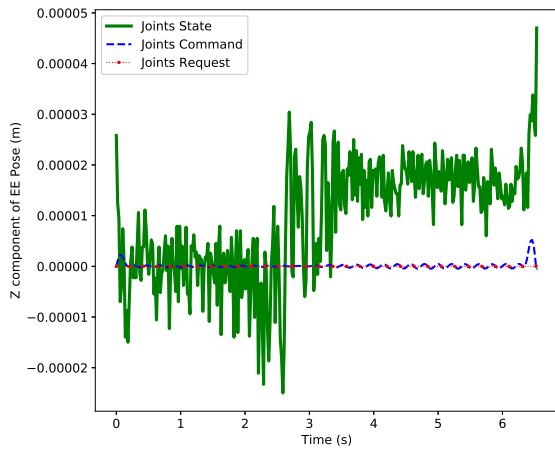


Figure B.45: Z component of end-effector pose for fitted Tow Course 2. The higher, the deeper it descends.

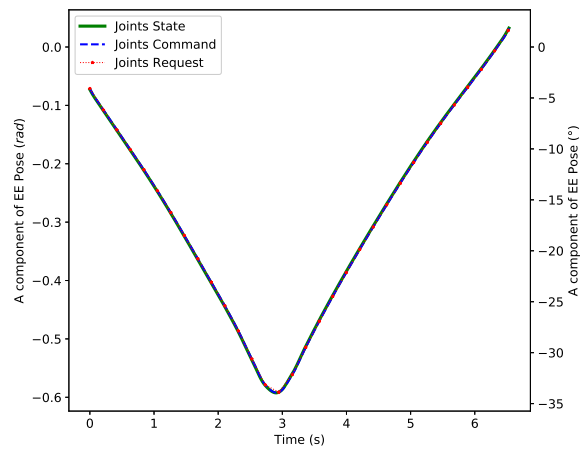


Figure B.46: A component (rotation about z -axis) for fitted Tow Course 2.

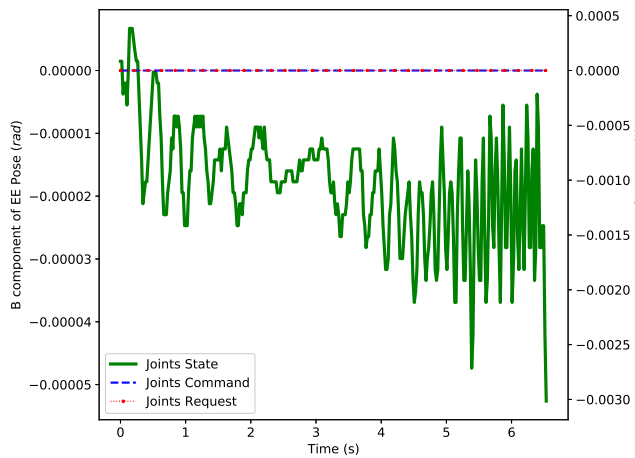


Figure B.47: B component (rotation about y -axis) for fitted Tow Course 2.

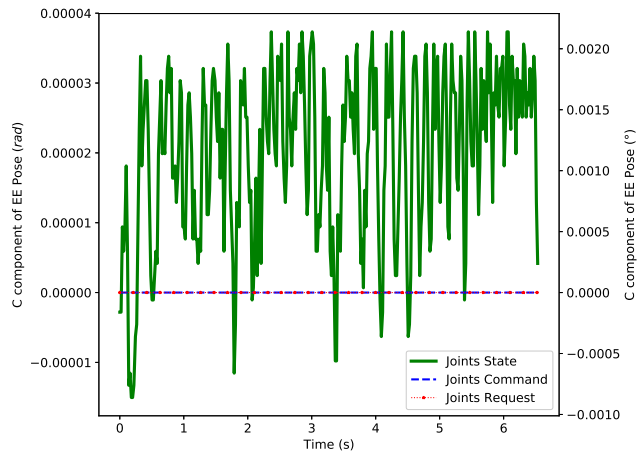


Figure B.48: C component (rotation about x -axis) for fitted Tow Course 2.

Bibliography

- [1] Brooks, T. R. and Martins, J. R. (2018) On manufacturing constraints for tow-steered composite design optimization. *Composite Structures*, **204**, 548–559.
- [2] Lukaszewicz, D. H. J. A., Ward, C., and Potter, K. (2012) The engineering aspects of automated prepreg layup: history, present and future. *Composites Part B: Engineering*, **43**(3), 997–1009.
- [3] Marsh, G. (2010) Airbus A350 XWB update. *Reinforced Plastics*, **54**(6), 20–24.
- [4] Lewis, A. (2014) Making composite repairs to the 787. *Boeing Edge AeroMagazine*, **56**(4), 5–14.
- [5] ROS-Industrial website accessed 17 February 2020, <https://rosindustrial.org/>.
- [6] Muxfeldt, A., Kubus, D., and Wahl, F. M. (2015) Developing new application fields for industrial robots - Four examples for academia-industry collaboration. In *IEEE International Conference on Emerging Technologies and Factory Automation*, pp. 1–7.
- [7] Harik, R., Saidy, C., Williams, S. J., Gurdal, Z., and Grimley, B. (2018) Automated fiber placement defect identity cards: cause, anticipation, existence, significance, and progression. In *SAMPE Conference 18*, pp. 1–16.
- [8] Clancy, G. J., Peeters, D. M. J., Oliveri, V., Jones, D., O’Higgins, R. M., and Weaver, P. M. (2019) A study of the influence of processing parameters on steering of carbon fibre/PEEK tapes using laser-assisted tape placement. *Composites Part B: Engineering*, **163**, 243–251.
- [9] Oliveri, V., Zucco, G., Peeters, D. M. J., Clancy, G. J., Telford, R., Rouhi, M., McHale, C., O’Higgins, R. M., Young, T. M., and Weaver, P. M. (2019) Design, manufacture and test of an in-situ consolidated thermoplastic variable-stiffness wingbox. *AIAA Journal*, **57**(4), 1671–1683.
- [10] Gao, J. (2018) Optimal motion planning in redundant robotic systems for automated composite lay-up process. *École centrale de Nantes*, pp. 1–134.
- [11] Gurdal, Z. and Olmedo, R. (1993) In-plane response of laminates with spatially varying fiber orientations - variable stiffness concept. *AIAA Journal*, **31**(4), 751–758.
- [12] Peeters, D. M. J., Lozano, G. G., and Abdalla, M. M. (2018) Effect of steering limit constraints on the performance of variable stiffness laminates. *Computers and Structures*, **196**, 94–111.
- [13] Lozano, G. G., Tiwari, A., Turner, C., and Astwood, S. (2016) A review on design for manufacture of variable stiffness composite laminates. *Proceedings of the Institution of Mechanical Engineers, Part B: Journal of Engineering Manufacture*, **230**(6), 981–992.
- [14] Heller, R. A. and Chiba, T. (1973) Alleviation of the stress concentration with analogue reinforcement. *Experimental Mechanics*, **13**(12), 519–525.
- [15] Hyer, M. W. and Charette, R. F. (1991) The use of curvilinear fiber format in composite structure design. *AIAA Journal*, **29**(6), 1011–1015.
- [16] Hyer, M. W. and Lee, H. H. (1991) The use of curvilinear fiber format to improve buckling resistance of composite plates with central circular holes. *Composite Structures*, **18**(3), 239–261.
- [17] Temmen, H., Degenhardt, R., and Raible, T. (2006) Tailored fibre placement optimisation tool. In *25th International Congress of Aeronautical Sciences*, pp. 2462–2471.
- [18] Peeters, D. M. J., Irisarri, F. X., Groenendijk, C., and Růžek, R. (2019) Optimal design, manufacturing and testing of non-conventional laminates. *Composite Structures*, **210**, 29–40.
- [19] Brooks, T. R., Martins, J. R., and Kennedy, G. J. (2019) High-fidelity aerostructural optimization of tow-steered composite wings. *Journal of Fluids and Structures*, **88**, 122–147.
- [20] Crothers, P., Drechsler, K., Felten, D., Herszberg, I., and Kruckenberg, T. (1997) Tailored fibre placement to minimise stress concentrations. *Composites Part A*, **28**, 619–625.
- [21] Ijsselmuiden, S. T. (2011) Optimal design of variable stiffness composite structures using lamination parameters. *Delft University of Technology*, pp. 1–210.
- [22] Lopes, C. S., Gürdal, Z., and Camanho, P. P. (2008) Variable-stiffness composite panels: buckling and first-ply failure improvements over straight-fibre laminates. *Computers and Structures*, **86**(9), 897–907.
- [23] Weaver, P. M., Potter, K., Hazra, K., Savarymuthapulle, M., and Hawthorne, M. (2009) Buckling of variable angle tow plates: from concept to experiment. In *50th AIAA/ASME/ASCE/AHS/ASC Structures, Structural Dynamics and Materials Conference*, pp. 2509–2519.
- [24] Wu, Z., Raju, G., and Weaver, P. M. (2018) Optimization of postbuckling behaviour of variable thickness composite panels with variable angle tows: towards “buckle-free” design concept. *International Journal of Solids and Structures*, **132**, 66–79.
- [25] Gurdal, Z., Tatting, B., and Wu, K. (2005) Tow-placement technology and fabrication issues for laminated composite structures. In *46th AIAA/ASME/ASCE/AHS/ASC Structures, Structural Dynamics and Materials Conference*, pp. 1–18.
- [26] Sabido, A., Bahamonde, L., Harik, R., and van Tooren, M. J. (2017) Maturity assessment of the laminate variable stiffness design process. *Composite Structures*, **160**, 804–812.
- [27] Blom, A. W. (2010) Structural performance of fiber-placed, variable-stiffness composite conical and cylindrical shells. *Delft University of Technology*, pp. 1–260.
- [28] Wu, K., Tatting, B., Smith, B., Stevens, R., Occhipinti, G., Swift, J., Achary, D., and Thornburgh, R. (2009) Design and manufacturing of tow-steered composite shells using fiber placement. In *50th AIAA/ASME/ASCE/AHS/ASC Structures, Structural Dynamics and Materials Conference*, pp. 1–18.

- [29] Kim, B. C., Hazra, K., Weaver, P. M., and Potter, K. (2011) Limitations of fibre placement techniques for variable angle tow composites and their process-induced defects. In *18th International Conference on Composite Materials Limitations*, pp. 1–6.
- [30] Waldhart, C., Gurdal, Z., and Ribbens, C. (1996) Analysis of tow placed, parallel fiber, variable stiffness laminates. In *37th AIAA/ASME/ASCE/AHS/ASC Structures, Structural Dynamics and Materials Conference*, pp. 2210–2220.
- [31] Kim, B. C., Potter, K., and Weaver, P. M. (2012) Continuous tow shearing for manufacturing variable angle tow composites. *Composites Part A: Applied Science and Manufacturing*, **43**(8), 1347–1356.
- [32] Kim, B. C., Weaver, P. M., and Potter, K. (2014) Manufacturing characteristics of the continuous tow shearing method for manufacturing of variable angle tow composites. *Composites Part A: Applied Science and Manufacturing*, **61**, 141–151.
- [33] Jutte, C. V., Wieseman, C. D., Lovejoy, A. E., and Stanford, B. (2020) Static loads testing of a high aspect ratio tow-steered wingbox. In *AIAA Scitech 2020* pp. 1–21.
- [34] Pires, J. N. (2007) Industrial robots programming: building applications for the factories of the future, Springer Science+Business Media, 1st edition.
- [35] KUKA Roboter GmbH (2015) Controller KR C4, KR C4 CK: Operating Instructions.
- [36] KUKA Roboter GmbH (2013) KUKA System Software 8.3: Operating and programming instructions for end-users.
- [37] Shirinzadeh, B., Wei Foong, C., and Hui Tan, B. (2000) Robotic fibre placement process planning and control. *Assembly Automation*, **20**(4), 313–320.
- [38] Shirinzadeh, B., Cassidy, G., Oetomo, D., Alici, G., and Ang, M. H. (2007) Trajectory generation for open-contoured structures in robotic fibre placement. *Robotics and Computer-Integrated Manufacturing*, **23**(4), 380–394.
- [39] Yan, L., Chen, Z. C., Shi, Y., and Mo, R. (2014) An accurate approach to roller path generation for robotic fibre placement of free-form surface composites. *Robotics and Computer-Integrated Manufacturing*, **30**(3), 277–286.
- [40] Hasenjaeger, B. (2013) Programming and simulating automated fiber placement CNC machines. *SAMPE Journal*, **49**(6), 7–13.
- [41] Sanfilippo, F., Hatledal, L. I., Zhang, H., Fago, M., and Petersen, K. Y. (2015) Controlling KUKA industrial robots: flexible communication interface JOpenShowVar. *IEEE Robotics & Automation Magazine*, **22**(4), 96–109.
- [42] Quigley, M., Conley, K., Gerkey, B., Faust, J., Foote, T., Leibs, J., Berger, E., Wheeler, R., and Ng, A. (2009) ROS: an open-source robot operating system. In *ICRA Workshop on Open Source Software*, pp. 1–6.
- [43] Michel, O. (2004) Webots: Professional mobile robot simulation. *International Journal of Advanced Robotic Systems*, **1**(1), 39–42.
- [44] Jackson, J. (2007) Microsoft robotics studio: A technical introduction. *IEEE Robotics and Automation Magazine*, **14**(4), 82–87.
- [45] Chitta, S., Sucan, I., and Cousins, S. (2012) Moveit! [ROS topics]. *IEEE Robotics and Automation Magazine*, **19**(1), 18–19.
- [46] Debout, P., Chanal, H., and Duc, E. (2011) Tool path smoothing of a redundant machine: application to automated fiber placement. *CAD Computer Aided Design*, **43**(2), 122–132.
- [47] Lukaszewicz, D. H. J. A. (2011) Optimisation of high-speed automated layup of thermoset carbon-fibre preimpregnates. *University of Bristol*, pp. 1–215.
- [48] Jeffries, K. A. (2013) Enhanced robotic automated fiber placement with accurate robot technology and modular fiber placement head. *SAE International Journal of Aerospace*, **6**(2), 774–779.
- [49] Chen, X., Zhang, Y., Xie, J., Du, P., and Chen, L. (2018) Robot needle-punching path planning for complex surface preforms. *Robotics and Computer-Integrated Manufacturing*, **52**, 24–34.
- [50] Martinec, T., Mlýnek, J., and Petrů, M. (2015) Calculation of the robot trajectory for the optimum directional orientation of fibre placement in the manufacture of composite profile frames. *Robotics and Computer-Integrated Manufacturing*, **35**, 42–54.
- [51] Kaloorazi, M. H. F., Bonev, I. A., and Birglen, L. (2018) Simultaneous path placement and trajectory planning optimization for a redundant coordinated robotic workcell. *Mechanism and Machine Theory*, **130**, 346–362.
- [52] Gasparetto, A., Boscariol, P., Lanzutti, A., and Vidoni, R. (2015) Path planning and trajectory planning algorithms: a general overview. In Carbone, G. and Gomez-Bravo, E. (eds.), *Motion and operation planning of robotic systems: background and practical approaches*, pp. 3–28, Springer Science+Business Media.
- [53] Shiller, Z. (2015) Off-line and on-line trajectory planning. In Carbone, G. and Gomez-Bravo, E. (eds.), *Motion and operation planning of robotic systems: background and practical approaches*, pp. 29–62, Springer Science+Business Media.
- [54] LaValle, S. M. (2006) Planning algorithms, Cambridge University Press, 1st edition.
- [55] Hauser, K. and Ng-Thow-Hing, V. (2010) Fast smoothing of manipulator trajectories using optimal bounded-acceleration shortcuts. In *IEEE International Conference on Robotics and Automation*, pp. 2493–2498.
- [56] Schulman, J., Duan, Y., Ho, J., Lee, A., Awwal, I., Bradlow, H., Pan, J., Patil, S., Goldberg, K., and Abbeel, P. (2014) Motion planning with sequential convex optimization and convex collision checking. *International Journal of Robotics Research*, **33**(9), 1251–1270.
- [57] Kuntz, A., Bowen, C., and Alterovitz, R. (2017) Fast anytime motion planning in point clouds by interleaving sampling and interior point optimization. In *International Symposium on Robotic Research*, pp. 1–16.
- [58] LaValle, S. M. (1998) Rapidly-exploring random trees: a new tool for path planning.
- [59] Kuffner Jr, J. J. and LaValle, S. M. (2000) RRT-connect: An efficient approach to single-query path planning. In *IEEE International Conference on Robotics and Automation*, pp. 995–1001.
- [60] Meijer, J., Lei, Q., and Wisse, M. (2017) Performance study of single-query motion planning for grasp execution using various manipulators. In *18th International Conference on Advanced Robotics*, pp. 450–457.

- [61] De Maeyer, J., Moyaers, B., and Demeester, E. (2017) Cartesian path planning for arc welding robots: Evaluation of the descartes algorithm. In *22nd IEEE International Conference on Emerging Technologies and Factory Automation*, pp. 1–8.
- [62] Shin, K. and Mckay, N. (1985) Minimum-time control of robotic manipulators with geometric path constraints. *IEEE Transactions on Automatic Control*, **30**(6), 531–541.
- [63] Shi, B. H. and He, J. (2016) The robot motion trajectory algorithm research based on B-spline and new velocity planning. In *28th Chinese Control and Decision Conference*, pp. 5968–5974.
- [64] Jiao, S.-x., Wang, H., Xia, L.-l., and Zhang, S. (2018) Research on trajectory planning of 6-dof cutting-robot in machining complex surface. *MATEC Web of Conferences*, **220**, 1–7.
- [65] Olabi, A., Béarée, R., Gibaru, O., and Damak, M. (2010) Feedrate planning for machining with industrial six-axis robots. *Control Engineering Practice*, **18**(5), 471–482.
- [66] Lee, A. C., Lin, M. T., Pan, Y. R., and Lin, W. Y. (2011) The feedrate scheduling of NURBS interpolator for CNC machine tools. *CAD Computer Aided Design*, **43**(6), 612–628.
- [67] Sun, Y., Zhao, Y., Bao, Y., and Guo, D. (2014) A novel adaptive-feedrate interpolation method for NURBS tool path with drive constraints. *International Journal of Machine Tools and Manufacture*, **77**, 74–81.
- [68] Bobrow, J. E., Dubowsky, S., and Gibson, J. S. (1985) Time-optimal control of robotic manipulators along specified paths. *The International Journal of Robotics Research*, **4**(3), 3–17.
- [69] Kunz, T. and Stilman, M. (2013) Time-optimal trajectory generation for path following with bounded acceleration and velocity. In Roy, N., Newman, P., and Srinivasa, S., (eds.), *Robotics: Science and Systems VIII*, pp. 209–216, The MIT Press.
- [70] Pham, Q. C. (2014) A general, fast, and robust implementation of the time-optimal path parameterization algorithm. *IEEE Transactions on Robotics*, **30**(6), 1533–1540.
- [71] Pham, H. and Pham, Q. C. (2018) A new approach to time-optimal path parameterization based on reachability analysis. *IEEE Transactions on Robotics*, **34**(3), 645–659.
- [72] Hauser, K. (2014) Fast interpolation and time-optimization with contact. *International Journal of Robotics Research*, **33**(9), 1231–1250.
- [73] Piazzzi, A. and Visioli, A. (2000) Global minimum-jerk trajectory planning of robot manipulators. *IEEE Transactions on Industrial Electronics*, **47**(1), 140–149.
- [74] Constantinescu, D. and Croft, E. A. (2000) Smooth and time-optimal trajectory planning for industrial manipulators along specified paths. *Journal of Robotics Systems*, **17**(5), 233–249.
- [75] Gasparetto, A. and Zanotto, V. (2008) A technique for time-jerk optimal planning of robot trajectories. *Robotics and Computer-Integrated Manufacturing*, **24**(3), 415–426.
- [76] Liu, H., Lai, X., and Wu, W. (2012) Time-optimal and jerk-continuous trajectory planning for robot manipulators with kinematic constraints. *Robotics and Computer-Integrated Manufacturing*, **29**(2), 309–317.
- [77] Biagiotti, L. and Melchiorri, C. (2008) Trajectory planning for automatic machines and robots, Springer Science+Business Media, 1st edition.
- [78] Lange, F. and Albu-Schäffer, A. (2016) Path-accurate online trajectory generation for jerk-limited industrial robots. *IEEE Robotics and Automation*, **1**(1), 82–89.
- [79] KUKA Roboter GmbH (2015) KR QUANTEC extra with F and C variants: Specification.
- [80] KUKA Roboter GmbH (2016) KUKA RobotSensorInterface 3.3.
- [81] Chitta, S., Marder-Eppstein, E., Meeussen, W., Pradeep, V., Tsouroukdissian, A. R., Bohren, J., Coleman, D., Magyar, B., Raiola, G., and Lüdtke, M. (2017) ros_control: A generic and simple control framework for ROS. *The Journal of Open Source Software*, **2**(20), 456.
- [82] Hanson, A. J. and Ma, H. (1995) Parallel transport approach to curve framing. pp. 1–20.
- [83] Morozov, M., Riise, J., Summan, R., Pierce, S., Mineo, C., MacLeod, C., and Brown, R. (2016) Assessing the accuracy of industrial robots through metrology for the enhancement of automated non-destructive testing. In *IEEE International Conference on Multisensor Fusion and Integration for Intelligent Systems (MFI)* pp. 335–340.
- [84] Kubela, T., Pochyly, A., and Singule, V. (2016) Assessment of industrial robots accuracy in relation to accuracy improvement in machining processes. In *IEEE International Power Electronics and Motion Control Conference (PEMC)* pp. 720–725.
- [85] Kubela, T., Pochyly, A., and Singule, V. (2019) High accurate robotic machining based on absolute part measuring and on-line path compensation. In *International Conference on Electrical Drives & Power Electronics (EDPE)* pp. 143–148.
- [86] Craig, J. J. (2005) Introduction to robotics: mechanics and control, Pearson Education, 3rd edition.
- [87] Siciliano, B., Sciavicco, L., Villani, L., and Oriolo, G. (2010) Robotics: modelling, planning and control, Springer Science+Business Media, 1st edition.
- [88] Denavit, J. and Hartenberg, R. S. (1955) A kinematic notation for lower-pair mechanisms based on matrices. *Journal of Applied Mechanics*, **77**, 215–221.
- [89] Horsch, T. and Jüttler, B. (1998) Cartesian spline interpolation for industrial robots. *Computer-Aided Design*, **30**(3), 217–224.
- [90] Chirikjian, G. S. and Burdick, J. W. (1994) A hyper-redundant manipulator. *IEEE Robotics Automation Magazine*, **1**(14), 22–29.
- [91] Whitney, D. E. (1969) Resolved motion rate control of manipulators and human prostheses. *IEEE Transactions on Man-Machine Systems*, **10**(2), 47–53.
- [92] Wampler, C. W. (1986) Manipulator inverse kinematic solutions based on vector formulations and damped least-squares methods. *IEEE Transactions on Systems, Man, and Cybernetics*, **16**(1), 93–101.
- [93] Liégeois, A. (1977) Automatic supervisory control of the configuration and behavior of multibody mechanisms. *IEEE Transactions on Systems, Man, and Cybernetics*, **7**(12), 868–871.

- [94] Dubey, R. V., Euler, J. A., and Babcock, S. M. (1988) An efficient gradient projection optimization scheme for a seven-degree-of-freedom redundant robot with spherical wrist. In *IEEE International Conference on Robotics and Automation*, pp. 28–36.
- [95] Huo, L. and Baron, L. (2008) The joint-limits and singularity avoidance in robotic welding. *Industrial Robot*, **35**(5), 456–464.
- [96] Siciliano, B. (1990) Kinematic control of redundant robot manipulators: a tutorial. *Journal of Intelligent and Robotic Systems*, **3**(3), 201–212.
- [97] Ueberle, M., Mock, N., and Buss, M. (2004) VISHARD10, a novel hyper-redundant haptic interface. In *12th International Symposium on Haptic Interfaces for Virtual Environment and Teleoperator Systems*, pp. 58–65.
- [98] Jin, L., Li, S., La, H. M., and Luo, X. (2017) Manipulability optimization of redundant manipulators using dynamic neural networks. *IEEE Transactions on Industrial Electronics*, **64**(6), 4710–4720.
- [99] Abo-Hammour, Z. S., Mirza, N. M., Mirza, S. M., and Arif, M. (2002) Cartesian path generation of robot manipulators using continuous genetic algorithms. *Robotics and Autonomous Systems*, **41**(4), 179–223.
- [100] da Graça Marcos, M., Tenreiro Machado, J. A., and Azevedo-Perdicoúlis, T. P. (2009) Trajectory planning of redundant manipulators using genetic algorithms. *Communications in Nonlinear Science and Numerical Simulation*, **14**(7), 2858–2869.