UNIVERSITY OF MISKOLC
FACULTY OF MECHANICAL ENGINEERING AND INFORMATICS

COLLABORATING ROBOT ARMS USING ARTIFICIAL INTELLIGENCE TECHNIQUES

PHD THESES

Prepared by

Rabab Benotsmane
Engineering of Automatic and Industrial informatics (BSc),
Engineering of Automation and Control of industrial systems (MSc)

ISTVÁN SÁLYI DOCTORAL SCHOOL OF MECHANICAL ENGINEERING SCIENCES
TOPIC FIELD OF DESIGN OF MACHINES AND STRUCTURES
TOPIC GROUP OF DESIGN OF MECHATRONIC SYSTEMS

Head of Doctoral School

Dr. Gabriella Bognár
DSc, Full Professor

Scientific Supervisors

Dr. László Dudás
PhD, Associate Professor
&
Dr. György Kovács
Dr. habil, Associate Professor

Miskolc
2021

“*That man can have nothing but what he strives for (39) That (the fruit of) his striving will soon come in sight (40) Then will he be rewarded with a reward complete (41) That to thy Lord is the final Goal (42)*”

Quran [39-42: An-Najm 53]

CONTENTS

CONTENTS ..... **I**

SUPERVISOR'S RECOMMENDATIONS ..... **III**

LIST OF SYMBOLS AND ABBREVIATIONS **4**

LIST OF FIGURES **8**

1. INTRODUCTION **10**
1.1. Motivation *11*
1.2. Aims and scope *12*
1.3. Applied research methodology *13*
1.4. Dissertation guide *15*

2. LITERATURE REVIEW **17**
2.1. Industry 4.0 conception *17*
2.2. Collaborating robots in Smart Factory *19*
2.3 Industrial robot arm *23*
2.4. Trajectory optimization of robot arms *25*
2.5. Artificial Intelligence and optimisation methods *26*

3. COOPERATIVE ROBOT ARMS INNOVATION **31**
3.1. Characteristics of RV-2AJ and RV-2SD Mitsubishi robot arms *32*
3.2. Task application: "Cards house" building *33*
3.3. "Cards house" building in real environment *35*
3.4. Result of task execution *37*

4. MODEL ANALYSIS FOR RV-2SD AND RV-2AJ ROBOT ARMS **41**
4.1. Forward and inverse geometry models *41*
4.2. Forward and inverse kinematics models *51*

5. STRUCTURAL DESIGN OF MANIPULATOR ARM USING SIMULATION TOOLS **56**
5.1. Design of industrial robot arms by application of SolidWorks software *56*
5.2. The CAD model constructed for RV-2AJ and RV-2SD robots arm *58*

6. "CARDS HOUSE" BUILDING IN VIRTUAL ENVIRONMENT **60**
6.1. Design of the "card house" model in CoppeliaSim *60*
6.2. Building of "card house" in CoppeliaSim *63*
6.3. Cooperating robots in CoppeliaSim *65*

7. MOTION CONTROL OF MANIPULATOR ARM USING SIMULATION TOOLS **66**
7.1. Dynamic analysis and controlling tool in MATLAB software *66*

8. CYCLE TIME REDUCTION OF ROBOT ARM **68**
8.1. Modelling the robot arm and the original and improved trajectories *68*
8.2. Newly elaborated "Whip-lashing" method *69*
8.3. Trajectory optimization's results *75*

9. NEWLY ELABORATED HYBRID ALGORITHM FOR OPTIMIZATION OF ROBOT ARMS' TRAJECTORY **81**
9.1. Tabu-search based optimisation method process *81*
9.2. Parameters of the optimisation task *84*
9.3. Case study of Hybrid algorithm for optimization of robot arms' trajectory *84*

10. THESES – NEW SCIENTIFIC RESULTS **90**

11. SUMMARY **91**

FUTURE WORK **93**

ACKNOWLEDGEMENTS **94**

REFERENCES **96**

LIST OF PUBLICATIONS RELATED TO THE TOPIC OF THE RESEARCH FIELD **101**

APPENDICES **102**

Appendix 1: Characterestics of an industrial arm *103*

Appendix 2: AI methods and their possible use for a robotic arm *105*

Appendix 3: Technical measurments of RV-2AJ and RV-2SD robot arms *107*

Appendix 4: Modeling analysis calculation *109*

Appendix 5: Script codes for motion control of the robot arm *123*

SUPERVISOR'S RECOMMENDATIONS

Rabab Benotsmane, is an Algerian PhD student with a degree in electrical engineering and mechatronics. During her M.Sc. studies, she dealt with the kinematic and dynamic tasks of robotic arms. She started her studies at the University of Miskolc in the autumn of 2017 as a Stipendium Hungaricum PhD student.

She has successfully passed her exams and carried out scientific research actively in field of cooperating robots (e.g. in solving the "card house building" task by cooperating robots) until the restrictive measures intervened taken in connection with the Covid-19 pandemic. The building of the Institute of Informatics were not open to students in 2020; therefore, we had to focus on virtual modelling and simulation of collaborative robots instead of real implementation of the cooperative activity of robots.

Rabab Benotsmane is a very enthusiastic researcher. She has written 16 scientific publications (7 high quality articles in Scopus indexed journals; among these 1 article in an IF/Q1 international journal and 6 articles in Q3 international journals) with her supervisors. Furthermore, she has taken presentations frequently at national and international scientific conferences.

Main significant added value of her research is that she elaborated a novel "Whip-lashing" method and a new "Hybrid Tabu-search" optimization algorithm for trajectory optimization of robot arms resulting in cycle time saving. These methods are introduced in the dissertation in detailed.

Based on the before mentioned facts, we as supervisors of Rabab Benotsmane (Dr. László Dudás and Dr. habil. György Kovács associate professors) consider her PhD studies to be successful.

LIST OF SYMBOLS AND ABBREVIATIONS

GREEK LETTERS

$\mathbf{\Gamma}$	: Vector of joint torques or forces	[N·m]
α_j	: The angle between $\mathbf{z}_{j-1}$ and $\mathbf{z}_j$ about $\mathbf{x}_{j-1}$	[rad]
θ_j	: The angle between $\mathbf{x}_{j-1}$ and $\mathbf{x}_j$ about $\mathbf{z}_j$	[rad]
ω_n	: Angular velocity	[rad/s]
ω_j	: Angular velocity of link j	[rad/s]
$\dot{\omega}_j$	: Angular acceleration of link j	[rad/s]

LATIN LETTERS

A	: The $(\mathbf{n}\times\mathbf{n})$ symmetric and positive definite inertia matrix of the robot	
C	: Vector of Coriolis and centrifugal torques	[N·m]
g	: Gravitational acceleration	[m/s²]
j	: The index number of the joint	
q	: Vector of joint positions	[rad]
x	: The state of the end effector in closed-loop scheme	
n	: Number of joints in a robot arm	
I	: Identity matrix	
X	: The position of the end effector	[m]
J	: Jacobian matrix	
Fe	: Vector of forces and moments exerted by the robot on the environment	[N]
L	: Lagrangian of the system equal to E – U	[kg·m²/s²]
E	: Total kinetic energy of the system	[kg·m²/s²]
U	: Total potential energy of the system	[kg·m²/s²]
Q	: Vector of gravity	[m/s²]
p	: Momentum	[kg·m/s]
m	: Mass of an object	[kg]
v	: Velocity of an object	[m/s]
ct	: Cycle time	[s]
sct	: Starting cycle time	[s]
ts	: Time step	[s]
sts	: Starting time step	[s]
ets	: Ending time step	[s]
sct	: Searched cycle time	[s]
I	: Number of trajectory points	
TP	: Trajectory points	

T	: Torque vector	[N·m]
mT	: Maximum torques vector	[N·m]
aT	: Allowable torques vector	[N·m]
TS	: Tabu-search algorithm	
PI	: Point insertion	
GR	: Grid point	
gs	: Grid step	
N	: Number of trajectory points in Tabu-search algorithm	

SUBSCRIPTS

$\dot{q}$	: Vector of joint velocities	[rad/s]
$\ddot{q}$	: Vector of joint accelerations	[rad/s²]
$\hat{x}$	: The measured state by the sensing device	
x_d	: The desired state	
R_j	: The frame assign to joint j	
O_j	: The origin of the frame R_j	
d_j	: The distance between $\mathbf{z}_{j-1}$ and $\mathbf{z}_j$ along $\mathbf{x}_{j-1}$	[m]
r_j	: The distance between $\mathbf{x}_{j-1}$ and $\mathbf{x}_j$ along $\mathbf{z}_j$	[m]
H_j^{j-1}	: Homogeneous transformation matrix defining frame R_{j-1} into frame R_j	
$\mathbf{A}_j^{j-1}$	: Rotation matrix defining frame R_{j-1} into frame R_j	
P_j^{j-1}	: Cartesian position vector of the frame R_j	
$\mathbf{s}_j^{j-1}$	: The components of the unit vectors along the $\mathbf{x}_j$ expressed in frame R_j	
$\mathbf{n}_j^{j-1}$	: The components of the unit vectors along the $\mathbf{y}_j$ expressed in frame R_j	
$\mathbf{a}_j^{j-1}$	: The components of the unit vectors along the $\mathbf{z}_j$ expressed in frame R_j	
U_j	: The desired position according to the homogeneous transformation matrix	
$\dot{\mathbf{X}}$	: The velocity of the end effector	[m/s]
$\mathbf{V}_n$	: Linear velocity	[m/s]
$L_{k,n}$	: The position vector connecting O_k to O_n	[m]
$\mathbf{J}^+$	: The pseudoinverse of the Jacobian matrix $\mathbf{J}$	
$c_{i,jk}$	: Christoffell symbols	
$\mathbf{V}_j$	: Linear velocity of O_j	[m/s]

$\dot{\mathbf{V}}_j$	: Linear acceleration of O_j	[m/s²]
$\mathbf{V}_{Gj}$	: Velocity of the center of gravity of the link j	[m/s]
$\dot{\mathbf{V}}_{Gj}$	: Acceleration of the center of gravity of the link j	[m/s²]
$\mathbf{G}_j$	: Center-of-mass of link j	
$\mathbf{I}_{\mathbf{Gj}}$	: Inertia tensor of link j about $\mathbf{G}_j$	[kg·m²]
$\mathbf{I}_{\mathbf{aj}}$	: Moment of inertia of the rotor and the transmission system of actuator joint j	[kg·m²]
${}^{\mathrm{j}}J_{\mathrm{j}}$	: Inertia tensor of link j with respect to frame R_j	[kg·m²]
J_{j}	: Spatial inertia matrix of link j	[kg·m²]
L_{j}	: Position vector between $O_{\mathrm{j-1}}$ and O_{j}	[m]
L_{Cj}	: Vector of the center of mass coordinates of link j. It is equal to $O_{\mathrm{j}}\,\mathbf{G}_{\mathrm{j}}$	[m]
M_{j}	: Mass of link j	[kg]
$\mathbf{M}_{\mathrm{j}}$	: Moment of the external forces exerted on the link j around	
$\mathbf{M}_{Gj}$	: Moment of the external forces exerted on the link j around G_j	
$\mathbf{F}_j$	: External forces on link j	[N]
$\mathbf{f}_j$	: Force exerted on link j by link $j-1$	[N]
$\mathbf{fe}_j$	: Force exerted by link j on the environment	[N]
$\mathbf{m}_{\mathrm{j}}$	: Moment about O_{j} exerted on link j by link $j-1$	[kg·m²]
$\mathbf{m}_{\mathrm{ej}}$	: Moment about O_{j} exerted by link j on the environment	[kg·m²]
F_{sj}	: Dry friction parameter of joint j	
F_{vj}	: Viscous friction parameter of joint j	

ABBREVIATIONS

AI	: Artificial Intelligence
MAS	: Multi-Agent-System
Dof	: Degree of freedom
IoT	: Internet of things
HCR	: Human-Robot communication
Cobot	: Robot-Robot communication

CTM	: Computed torque method
PID	: Proportional-Integral-Derivative
ML	: Machine Learning
ANN	: Artificial Neural Networks
RL	: Reinforcement Learning
TS	: Tabu-search
DTS	: Decision Tree Search
CAD	: Computer Aided Design
Atan2	: Arc tangent function
DGM	: Direct Geometry Model
IGM	: Inverse Geometry Model
FK	: Forward Kinematic
IKM	: Inverse Kinematic Model
DDM	: Direct Dynamic Model
IDM	: Inverse Dynamic Model
URDF	: Unified Robot Description Format
XML	: Extensible Markup Language
STL	: Standard Triangle Language
API	: Application Programming Interface
GUI	: Graphic User Interface
DLS	: Damped Least Square
RBT	: Rigid Body Tree
Dt	: Desired trajectory

LIST OF FIGURES

Figure 1. The four industrial revolutions 17
Figure 2. Main pillars of Industry 4.0 concept 18
Figure 3. Human-robot collaboration in CoppeliaSim software 20
Figure 4. Cooperative industrial robots working together in a production line 20
Figure 5. Design of cooperating multi industrial robot systems 22
Figure 6. General structure of a manipulator arm 23
Figure 7. Closed-loop control of an industrial arm 24
Figure 8. Tabu-search flow chart 28
Figure 9. Decision Tree Search scheme 29
Figure 10. Cooperating industrial robots in real environment [96, 97] 31
Figure 11. Axes of RV-2AJ arm [96] 32
Figure 12. Industrial design of RV-2AJ arm [96] 32
Figure 13. Axes of RV-2SD arm [97] 33
Figure 14. Industrial design of RV-2SD arm [97] 33
Figure 15. Elements needed for card house building 34
Figure 16. Changing the batteries of RV-2AJ arm 35
Figure 17. Changing the batteries of controller 35
Figure 18. Connection between compressor and the gripper 35
Figure 19. Measurement and design of “card” element 36
Figure 20. Measurement of support element 36
Figure 21. CNC machine and its software 36
Figure 22. “House of cards” prototype 36
Figure 23. Pick and place scenario by RV-2AJ arm 36
Figure 24. The script code writing in MELFA-BASIC IV Language in COSIROP software 37
Figure 25. Imprecise contact between two “cards” 37
Figure 26. The mechanical holder prototype geometry development 38
Figure 27. “Card house” building by RV-2AJ robot arm in the real and virtual environment 39
Figure 28. The final view of holder for “card” elements including 16 elements 40
Figure 29. The new “card” basement 40
Figure 30. Frames assignments for RV-2AJ and RV-2SD robot arms 42
Figure 31. Simplification of frames assignments for RV-2AJ arm 42
Figure 32. Simplification of frames assignments for RV-2SD arm 44
Figure 33. RV-2AJ robot arm in SolidWorks software and the reality 58
Figure 34. RV-2SD robot arm in SolidWorks software and the reality 58
Figure 35. Generating URDF file in SolidWorks 59
Figure 36. Exporting URDF file in SolidWorks 59
Figure 37. Exporting XML file in SolidWorks 59
Figure 38. The pieces of equipment after the mashing process 60
Figure 39. Photo of configuration + angles + hierarchy 61
Figure 40. Code script of Forward kinematics for RV-2AJ arm 62
Figure 41. Inverse kinematic configuration for RV-2AJ arm 63
Figure 42. The configuration motion of RV-2AJ gripper 63
Figure 43. The final assembly model for “card house” building in CoppeliaSim 64
Figure 44. The cooperating process between RV-2AJ and RV-2SD robots in CoppeliaSim environment. 65
Figure 45. RV-2AJ arm is located in its home position and in a random position 66
Figure 46. Internal blocks in Simulink of RV-2AJ arm 67
Figure 47. The trajectories of the RV-2AJ arm in three views 69
Figure 48. Basic elements of a whip 70
Figure 49. The motion of a whip 70
Figure 50. The variation of velocity, mass, kinetic energy, and torque as a function of time 71
Figure 51. The motion of RV-2AJ arm 72

Figure 52. Cycle time minimization algorithm 74
Figure 53. Technical measuring of allowable torques........ 75
Figure 54. The original and the improved scenarios of RV-2AJ robot arm 75
Figure 55. The block system scheme of RV-2AJ arm of the original trajectory........ 76
Figure 56. The block system scheme of RV-2AJ arm of the improved trajectory........ 76
Figure 57. Variation of joint torques according to the two paths, ct = 5 [s]........ 77
Figure 58. Variation of joint torques—original path—for different cycles........ 78
Figure 59. Different iterations of cycle time minimization algorithm for the original path around the searched cycle time sct = 2 [s]........ 78
Figure 60. Variation of joints torques according to Improved path 79
Figure 61. Different iterations of the cycle time minimization algorithm........ 79
Figure 62. Optimal cycle time values for the original and improved paths 80
Figure 63. Tabu-search based optimisation method interpolation in 2D........ 83
Figure 64. Tabu-search flow chart for refining trajectory 83
Figure 65. Position insertion cycle flow chart........ 85
Figure 66. Hybrid algorithm execution for all iterations........ 86
Figure 67. The best candidates trajectories executed by the hybrid algorithm........ 86
Figure 68. The block system scheme of hybrid algorithm 88
Figure 69. Hybrid algorithm results for best trajectories in the mechanics explorer from different views 88
Figure 70. The joint torques variation for initial and final paths with ct=6[s] 89
Figure 71. The joint torques variation for initial and final paths with their minimum........ 89
Figure A.1. Prismatic joint (a) and revolute joint (b)........ 103
Figure A.2. Human arm behind an industrial arm........ 103
Figure A.3. Control scheme of industrial robot........ 104
Figure A.4. RV-2AJ arm 107
Figure A.5. Controller 107
Figure A.6. Teaching box 107
Figure A.7. COSIROP software 107
Figure A.8. Air compressor........ 107
Figure A.9. RV-2SD arm........ 107
Figure A.10. Controller CR1DA-700 series........ 107
Figure A.11. R56 Teaching box 107
Figure A.12. The geometric parameters in the case of a simple open structure........ 109
Figure A.13. Types of equations encountered in the Paul method........ 111
Figure A.14. The possible solutions for joint angles of RV-2AJ robot arm........ 111
Figure A.15. The possible solutions for joint angles of RV-2SD robot arm........ 112
Figure A.16. Singular positions of RV-2SD robot........ 112
Figure A.17. Script code for Forward and Inverse kinematics of RV-2AJ robot in MATLAB………...123
Figure A.18. Inputs arguments of hybrid algorithm for optimization of robot arm's trajectory 124

1. INTRODUCTION

Nowadays, the modern industry in all sectors is facing a new revolution known as Industry 4.0 [1,2], where many challenges and requirements are taken into consideration with the aim of building smart factories that combine flexibility and ability concepts [3,4] by developing a new paradigm based on the latest technologies, where automation and network systems present the efficient keys for realizing the new industrial revolution [5,6]. The essential target of Industry 4.0 is all about shifting to another paradigm transforming the factories into smart environments using Internet of things and endowing Artificial Intelligence (AI) algorithms on all industry departments, where machines and robots can monitor themselves with decentralizing the control system used for all equipments, cooperating and working in parallel to achieve the task required, optimizing the lead time and finding significant solutions in case of malfunction, the whole process is supervised and the data are collected in every moment using Big data technology, these data should be protected using Cybersecurity concept [7,8].

Recently, industrial robotics has become an important solution used in different sectors due to the advantages guaranteed by industrial robots [9], as manipulator arms and parallel robots are represented with higher precision and higher productivity. This optimizes the lead time of the production process [10, 11]. Especially with technological developments, the manipulator's arm, for example, presents the most often used tool in the production sector [12], where it can cooperate with its environment and work safely in the area of robotics, the trajectory planning of manipulator arms represents an essential field for focus. The execution of the well-defined task of a robot arm with optimized trajectory can guarantee many benefits such as a reduced cycle time and energy consumption, as well as increased productivity. The main objective in the trajectory planning field is to compute the desired points, that represent the reference input data for the controller of a robot using mathematical techniques.

A multi-agent system based on the cooperating concept of robots with themselves and with the products takes a large focus in the production process where the manufacturing economical benefits are in increasing with the development of these technologies.

Typically, multi-agent systems integrate autonomous systems which are endowed by Artificial Intelligence (AI) techniques, this concept aims to apply intelligent machines collaborating together to build a flexible environment, multiple agents that interact with each other in a common environment, some of which may be machines, computer programs, etc. In the industry, these agents represent robots, sensors, controllers that can apply a common language that structures the rules of cohabitation and collective work. To design a multi-agent system, we must know the model of each agent that will come into action and define their environment and their interactions and their essential objectives to achieve [13]. A classic opposition has been drawn between reactive and cognitive agents: the reactive agents are those that have just reflexes while the cognitive agents are those that can form plans for their behaviors.

In the industry, avoiding errors and estimating the cycle time and malfunction prediction are highly recommended for the production process that includes a set of machines with different type and execution tasks, where 3D simulation of processes is a way for providing real-time data to observe the physical world in a virtual surrounding that will include machines, products, and humans. It is a way to make scenarios for system operation and based on it to optimize the production and maximize the resources (human, equipment, etc.), thereby increasing productivity and reducing wastes, and improving quality.

All the concepts presented above describe efficient tools that should be used in the automation process, each concept is a deep subject that should be studied carefully, taking into consideration the newest research found on it.

In this dissertation, I present the concept of Cooperating industrial robots using Artificial Intelligence techniques, where the dissertation starts from a global view and goes deeply to trait the following subjects: Industry 4.0 which lead to the cooperating concept - Industrial robots - trajectory optimization for such robots - the use of optimization cases with such robots - Simulation process in the virtual environment. With the aim to clarify each point, a task application is executed to prove the combination necessity of them in the development of industrial processes.

1.1. Motivation

Cooperating industrial robots in the factory is not a new concept where, it is a well-known application since last decades, especially with the development of the automotive industry, the automation systems have replaced the human resources, by installing industrial robots in the production chains as parallel robots, manipulator arms and/or hybrid robots to accomplish a concrete task [14]: pick and place, packaging, welding [15] or painting. With the appearance of AI and Multi-Agent System (MAS) concepts the idea was developed to use intelligent industrial robots [16] instead of controlled ones collaborating together and take a decision in a smart way to achieve their tasks. Usually, industrial manipulator arms are the most used in the automotive industry because of their advantages to using them in different situations, also their ability to carry heavy products. Cooperating of multi manipulator arms guarantee the achievement of tasks planned more easily than single manipulator arm, it is known that the control of a single robot arm was always a trivial task comparing to the control of multi-robot arms which presents a real deal for scientist, the development of controller design for such structure was proposed in several articles [17].

Nowadays, automotive corporations as Audi, BMW, Mercedes have already emerged the concept of cooperating robots in their production chains trivially, the meaning of different industrial robots working together in the same environment and executing different tasks, if some malfunction or an error occurs the process, where one robot fails and became out of order to execute its task, therefore the whole process will stop till the maintenance will be done. Stopping all the robots and checking the error in the process takes a large time which causes the distribution of the lead time decreasing productivity.

From this insight, I started to build the basis of the dissertation, wherefrom my sight the process of cooperation robots can be developed in a smart way using AI tools, the idea is based on creating an assembly line that includes a few robot manipulators, these robots as members of a multi-agent system can help each other and cooperate to finalize the appropriate line tasks using efficient algorithms. If some malfunction or another problem occurs in the production line, the robots can reconfigure themselves and reorganize the steps of the same task.

1.2. Aims and scope

Collaborative robots using AI techniques is a large topic founded on many subjects that deal with different fields. With the aim to achieve the real goal of the dissertation the working path is divided into several aims presented as follow:

- First of all, as a reminder, the main goal is creating an assembly line using industrial robots, these robots working parallelly. If a problem occurs in the chain process, the robots will continue the work by using a quasi-optimal solution created by us or applying AI tools, this aim is explained theoretically by giving the hypothesis needed.

- The first aim in this large area is determining a real task that needs the cooperative concept, the task should be explained very precisely and taking into consideration all the equipments used: *The task is building a house of cards using two manipulator arms*, where from our childhood we knew this task by using two hands and building together with the house of cards, in our insight the two hands are presented as two manipulator arms.

- The second aim is about dealing with the industrial arm, *studying the theoretical part of the robot arm structure and modeling it in the virtual environment,* also being familiar with *programming a robot arm in the real-life*, respecting the workspace and the obstacles that can be existent.

- The third aim of the dissertation is to deal with the *trajectory improvement of robot motion* by studying the torques and velocity effects in each joint and develop better scenario to perfectionate the process.

- The fourth aim is *dealing with the virtual environment* since realizing the cooperative concept in the reality is a challenging task, therefore, simulation in the virtual robotic software gives us more flexibility *to check and control all the hypothesis* where boost us to develop new techniques.

- As a final aim, modeling the task in the virtual environment allows us to execute the process trying different AI algorithms that can be used for industrial robots, especially for finding the quasi-optimal robot trajectory developing an algorithm for such task.

1.3. Applied research methodology

To accomplish my research work I applied different scientific methodologies. They can be classified in two main groups:

- quantitative methodologies,
- qualitative methodologies.

These basic methodologies can be refined in the application areas:

- methodologies for literature review,
- methodologies for research,

where new methodology types also may appear.

1.3.1. Methodologies for literature review

Because of the explosion in the scientific areas covered by this dissertation there was a large enticement to collect literature without thoroughness and conduct ad hoc. Instead, I divided the research area in more sections for the discussion of the state of the art:

- Industry 4.0 oriented publications.
- Cooperative robots area documents.
- Robot arm trajectory optimisation-oriented papers.

- Publications concerning AI and optimisation methods used regarding the above three scientific areas.

The used documents served as fundament and background for the research but sometimes, like in case of the Whip lashing model or the "House of cards" building task they served as tools checking against the absolute novelty of the method or task.

The next characteristics of the used publications were evaluated:

- closeness to my research topic,
- novelty,
- scientific level,
- accessibility.

The closeness was checked through title, keywords and abstract of the publications, then the selected candidates were read and reselected. For the choosing of the papers for literature review the references section of the highly classified papers provided the main help beside te directed search.

The novelty was checked using the publishing date of the document and the publication date of the referenced by the document papers.

The scientific level was ensured by checking of the quality and publicity of the publisher and the journal series.

Finally, the accessibility of the full text of the document was an essential point because of time and financial limits, so the use of open libraries and documents downloadable from web were preferred.

a) Narrative methodology for literature review

I used widely this form of literature analysis because this form is suggested for PhD students and I saw many examples of this method in others thesis. Though I studied more professional literature analysis methods, I chose this because of its simplicity and wide range of its usage. I will introduce the studied other literature analysis methods shortly in the following.

b) Quantitative methodology

This method gives a reliable result. It needs collecting many references of selected topic then build up a table of scientific parameters of these references. The process needs fairly large time but is advantageous to find the best references and explore scientific gaps that need additional literature research. Though this method needs more work than the usual narrative literature analysis, provides a more comprehensive overview and deeper understanding of the cohesion of the selected research area so I will consider it in the future.

c) Qualitative methodology

This methodology needs systematic reading of the documents to find and identify novel thoughts, clustering and identifying the same or similar ideas into one group then the review of the documents is made emphasising the appearing novelties in the taken by time order documents. This method was applied only such a manner that the arising novelties were associated to the document and mentioned in the literature review.

1.3.2. Methodologies applied in the research process

a) Quantitative research methodology

From the quantitative research methodology, the next methods were applied for conducting the research by topic areas:

- Industry 4.0 topic: surveys, case studies.
- Robot arm trajectory optimisation: model generation, modelling, measurements, results analysis, parameter manipulation.
- Cooperative robots area: prototype creation, model generation, modelling, experiments, measurements, parameter manipulation.

b) Qualitative research methodology

From the qualitative research methodology, the next methods were applied for conducting the research by topic areas:

- Industry 4.0 topic: interpreting events, describing actions, observations, document analysis.
- Robot arm trajectory optimisation: model innovation, observations.
- Cooperative robots area: task innovation, observations.

1.4. Dissertation guide

In the second part of the dissertation, I present the literature review relating to the most relevant topics that will be discussed: Industry 4.0 - Cooperative robots - Robot arm control motion - Robot arm trajectory optimization – AI and optimization methods.

In the third part, I define the core of the dissertation by describing the research and innovations done during the PhD study period, where:

- An algorithm was created to reduce and optimize the robot arm cycle time, the method is based on Whip-lashing theory, where it is supposed that the robot arm motion can act as a whip, consequently it results in achieving the improved trajectory of the robot arms, in order to increase the velocity of the robot arm’s parts, thereby minimizing motion cycle times and to utilize the torque of the joints more effectively.

- I continued the improvement of trajectory optimization by using this time Tabu-search algorithm as AI tool, the method is presented in 3D grid with interpolated point (wayPoint) insertion, plus grid step halving. The method can determine a quasi-optimal trajectory where its application in the real environment is useful to minimize cycle time when avoids obstacles in the workspace of the robot.
- I realized the cooperative concept, by using two Mitsubishi robot arms, RV-2AJ robot arm with five (Dof) degrees of freedom and RV-2SD robot arm with six degrees of freedom, the robots are cooperating together in order to build a "house of cards" using card elements with a special design to maintain the stability of the house. The process is presented in two scenarios: - Normal scenario where both robots are executing their tasks correctly; - Abnormal scenario where one of the robots fails to execute its task, in this case, the second robot should complete the process by using a smart solution. This thesis is realized in real and virtual environments using different software: SolidWorks – CoppeliaSim.

In the final part, I declare the achieved results in theses, give the short summary of the work with emphasizing the projected research tasks in the future, list the used references and my publications that introduced my results.

2. LITERATURE REVIEW

This chapter gives a literature review that investigates different topics related to the dissertation's main goal "cooperative industrial robots" which has gained much importance in recent years, where the topic has been extensively explored from the academic community and the industrial one due to its economic benefits. The topics are exposed in logical hierarchy starting from Industry 4.0 concept and going deep to the main topics of the dissertation.

2.1. Industry 4.0 conception

The Industry 4.0 production philosophy, as a whole complex concept – known as the Industry of the Future – was created in Germany at the Hannover Fair in 2011. This concept has prevailed worldwide due to its strategy, which is based on a new way of organizing operations at the industrial level using new technologies [18], and represents the upcoming 4th industrial revolution. According to the concept, the 1st industrial revolution was the age of mechanization, the application of the steam engine. The 2nd industrial revolution was the age of mass production and electricity, while the 3rd industrial revolution was the age of automated production through the application of electronic and information systems as seen in **Figure 1**.

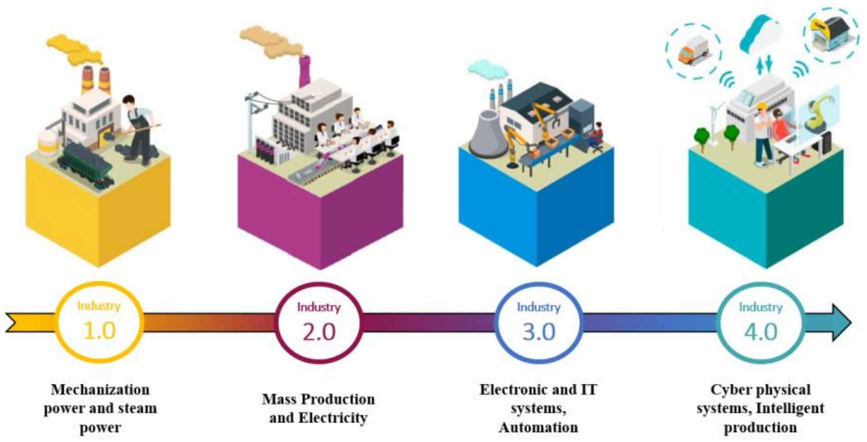

Industry 1.0
Industry 2.0
Industry 3.0
Industry 4.0
Mechanization power and steam power
Mass Production and Electricity
Electronic and IT systems, Automation
Cyber physical systems, Intelligent production

Figure 1. The four industrial revolutions

The technologies applied in the recent Industry 4.0 concept to create a Smart Factory are more interconnected, more communicative, and more intelligent than traditional manufacturing [19]. Instead of traditional supply chains, a digital global supply chain network is needed in the Industry 4.0 concept, one that can adapt flexibly to the changing unique customers' demands, to the activity of the supply chain members, and to the changing market environment.

Industry 4.0 can be defined as a digital transformation making autonomous decentralized decisions in all cyber-physical systems, where each element works in interaction. Products and machines communicate with each other, and the transfer of information is implemented by sensors, linked in a global network, which is itself connected to the whole supply chain that guarantees the individual needs of customers [20].

2.1.1. Pillars of Industry 4.0

Interoperability, virtualization, decentralization, real-time capability, and modularity must be present in the production systems in the Future Industry. These features are based on nine pillars, according to the Boston Consulting Group, and these are the newest technologies known all over the world [21]. **Figure 2** describes these nine main pillars of the Industry 4.0 concept.

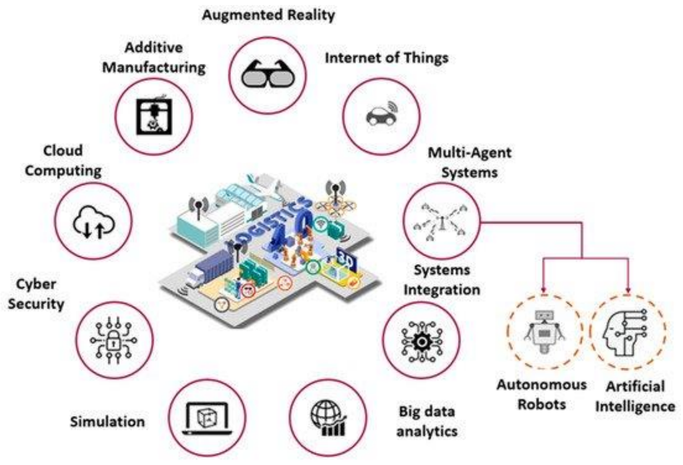


Figure 2. Main pillars of Industry 4.0 concept

1. **Multi-Agent Systems** (MAS) [22] can be intelligent smart machines, collaborating robots, sensors, controllers, etc., that are communicating with the production control system and the smart workpieces so that machines coordinate, control, and optimize themselves and the whole production process.
   - **Autonomous Robots** are intelligent industrial robots that can cooperate and collaborate with each other during manufacturing in order to perform more complex tasks with higher efficiency.
   - **Artificial Intelligence** is the ability of robots to learn and think logically and autonomously, not only depending on programs written by people.
2. **System Integration** means the optimal reconfiguration of the connecting Cyber-Physical Systems, which can be sensors, actuators, etc. (vertical integration) [23]; and the optimal operation of the whole supply chain including suppliers, manufacturers, service providers, etc. (horizontal integration).
3. **Big data** means the huge amount of information required for the optimal operation of intelligent network-like systems. The collection, analysis, and evaluation of this data set are essential for real-time decision making [24].

4. **Simulation** tools are used for optimization of production processes and maximal utilization of resources. Simulation is an efficient method for modeling deterministic and stochastic processes and supporting decision making.
5. **Cyber Security** includes technologies that are developed to protect systems, networks and data from cyber-attacks [25].
6. **Cloud Computing** provides unlimited computing power to transfer, store and analyze the huge amount of data required for the optimal operation of systems.
7. **Additive Manufacturing** is an innovative technology to build three-dimensional objects by adding layer upon layer of a given material. This 3D printing technology provides the possibility of manufacturing more complex and unique components and products, which is needed for more flexible and unique production [26].
8. **Augmented Reality** provides the possibility of visualization of manufacturing processes by transforming the real environment to a virtual environment.
9. **Internet of Things** (IoT) technology allows objects (e.g., machines, vehicles, products or other devices) to communicate and interact with each other. IoT provides the network connection and data exchange of objects [27].

Smart Factory is a complex system that integrates these main elements (e.g., autonomous robots, IoT, Big data, Cloud Computing, and simulation) of the Industry 4.0 production philosophy. The essence of the Smart Factory concept is that the traditional centrally controlled production processes will be replaced by decentralized control, in which the intelligent machines, robots, tools, and intelligent workpieces communicate and collaborate with each other continuously. Smart Factories are self-organizing, self-optimizing, and more competitive. Factories can self-optimize their own performance, self-adapting to new situations and conditions [26].

2.2. Collaborating robots in Smart Factory

Multi-Agent Systems include the collaborating robots using AI, which is an essential element of Industry 4.0 concept, are autonomous robots that become more autonomous and cooperative. Intelligent robots can interact with people, machines, equipment, products and other robots in order to improve productivity and product quality. These robots can perform more complex tasks and manage unexpected problems [28, 29].

A collaborating robot is able to interact with another robot in order to solve a problem in a complex situation. Application of the Industry 4.0 concept creates Smart Factories based on physical smart devices, thereby generally minimizing the number of workers in the workplace.

However, a highly skilled labor force is needed for the programming and operation of intelligent devices [30, 31]

The collaboration between agents may mean two well-known concepts:

1. **Collaborating human – robot (HCR)** in the modern automotive industry, this concept is already realized where human operators and robots working together to manufacture a car product [32, 33], the concept requires to study several aspects as collision, safety, and respect the robot workspace aiming to enable versatile automation steps and increase productivity. It is an additional element that combines human capabilities with the efficiency and precision of machines. **Figure 3** illustrates HCR scenario in CoppeliaSim software.

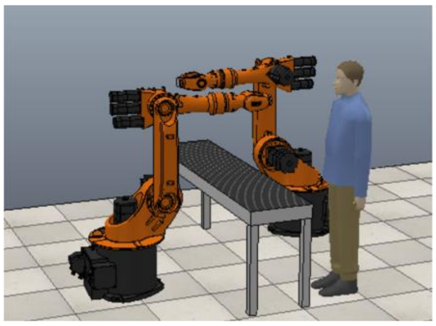


Figure 3. Human-robot collaboration in CoppeliaSim software

2. **Collaborating robot – robot (Cobot)** in future industry is highly recommended to transform the assembly chain to a flexible and structured line without any human interaction where multiple robots working cooperatively in a redundant way that will offer new possibilities in the execution of complex tasks in dynamic workspaces if some malfunction or an error occurs in the production line [34, 35]. The robotic manufacturing or assembly cell can react and make a decision easily to continue the task planned for. **Figure 4** presents cooperative industrial robots working together on the part of a production line.

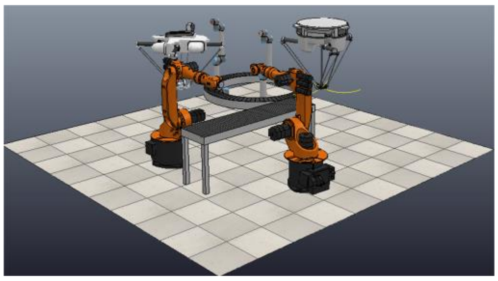


Figure 4. Cooperative industrial robots working together in a production line

2.2.1. Operational strategies in a Smart Factory with collaborating robots

There are many strategies and technologies aim to operate and control a manufacturing process in Industry 4.0, that depend on the production situations. We can define basically two scenarios in the Industry 4.0 concept [36, 37]:

1. **Normal Scenario:** When all of the manufacturing elements work correctly, agents (e.g., robots) collaborate and communicate together to achieve tasks according to a production plan. In this scenario, each agent controls its activity.
2. **Abnormal Scenario:** When a problem occurs in a production line that is caused by any of the agents, this results in an operational problem in the manufacturing system i.e., malfunction, lost data, or defective products. This abnormal situation can create many further economic problems; the avoidance of abnormalities can be achieved by realizing the following keys solution:

- **Simulation**: Is an efficient tool in the industry, which provides the modeling and visualization of the manufacturing processes of the Smart Factory in order to avoid problems and achieve more flexible and more efficient production [38].
- **Artificial Intelligence**: Is the ability of robots to learn and think logically and autonomously, not only depending on programs written by people, but also independently. AI helps to create a smart manufacturing environment where agents adapt to critical situations faster and make optimal decisions.

2.2.2. Design of cooperating industrial multi robot systems

The interest on developing cooperative systems has increased due to the advantages they offer against the single robot manipulators, since the cooperative systems can perform tasks which with a single robot would be impossible to achieve. In fact, in the industry, many tasks are difficult or impossible to be executed by a single robot manipulator, making it was necessary to use two or more manipulators in a cooperative way. Such tasks include handling heavy or large payloads, the assembly or disassembly of big or small pieces, and manipulating rigid or flexible objects [39, 40].

A cooperative system consists of multiple robot manipulators who aim to hold an object, so that position on the end effector of each is limited geometrically. These constraints model object and caused a reduction in degrees of freedom of the cooperative system because the end effector of each of the manipulators must maintain contact with the object, so it cannot be moved in all directions. The motion degrees of freedom lost become in contact forces, so they should be included in the dynamics of each of the robots to form the cooperative system. The dynamic model for each manipulator with restricted movement is obtained by using the Lagrangian formulation [41]:

$$A_i(q_i)\ddot{q}_i + C_i(q_i,\dot{q}_i)\dot{q}_i + g_i(q_i) = \Gamma_i$$

where $\boldsymbol{q}_i, \dot{\boldsymbol{q}}_i, \ddot{\boldsymbol{q}}_i \in \boldsymbol{R}^n$ are the position, velocity and acceleration joint space, respectively; $\boldsymbol{A}_i(\boldsymbol{q}_i) \in \boldsymbol{R}^{n_i \times n_i}$ is a symmetric positive-definite inertia matrix; $\boldsymbol{C}_i(\boldsymbol{q}_i, \dot{\boldsymbol{q}}_i) \in \boldsymbol{R}^{n_i \times n_i}$ is a matrix containing the Coriolis and centripetal torques effects; $\boldsymbol{g}_i(\boldsymbol{q}_i) \in \boldsymbol{R}^{n_i}$ is a vector of gravity torque obtained as a gradient result on the potential energy; and $\boldsymbol{\Gamma}_i \in \boldsymbol{R}^{n_i}$ is the generalized torque.

The manipulation of a rigid object using a cooperative system imposes some kinematic and dynamic constraints. The constraint is a characteristic which limits the geometry and system movement as presented in **Figure 5**.

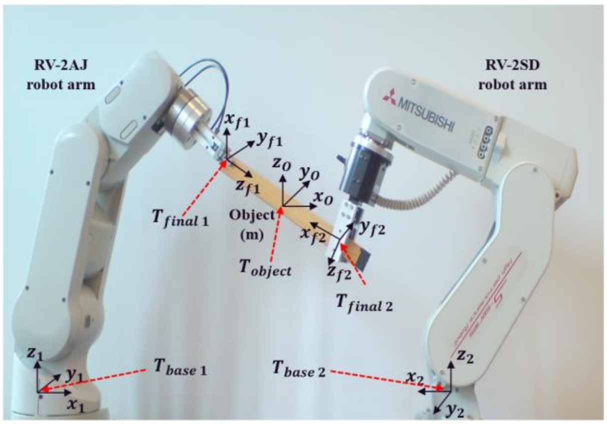


Figure 5. Design of cooperating multi industrial robot systems

It also must be applied to achieve a strong grip, quick control, proper load distribution, compliance with the planned trajectory, stable transition to perform the task and efficient closed-loop energy balance. Such requirements provide greater dexterity, flexibility in handling and assembly tasks. Consequently, in order to manipulate a body using a cooperative system, the goals will simultaneously control the position and velocity of each robot’s end-effectors under the internal forces applied over the body; ensure the grip; and control the external forces which move the object. In other words, the target in this task is to hold a body on a support, which let you do the grip with enough force to prevent the body from turning, slipping, falling or uncontrolled movements, allowing movement within your workspace. To perform these tasks there exist some possibilities, but the most-used control structure is the position/force control [42].

2.2.3. Assumptions on cooperative robots system

1. **Assumptions on manipulators** [42]

Assumption 1. The robots are formed by rotational joints.

Assumption 2. The links of the cooperative manipulators are rigid.

Assumption 3. The robot arms do not enter at singular configurations throughout the task.

Assumption 4. Cooperative robots are non-redundant.

2. **Assumptions on manipulated object** [42]

Assumption 5. The robot manipulators rigidly holding the object, therefore, no relative movement between the end-effectors and the object (stable grip condition).

Assumption 6. As it doesn't exist a relative movement between the end-effector and the object, the effects due to the tangential friction between the effectors of robotic manipulators, and object are null.

Assumption 7. The manipulated object is rigid and does not undergo deformation when it is gripped.

Assumption 8. We know the forward kinematic of the rigid object.

2.3 Industrial robot arm

A robot manipulator is an electronically controlled mechanism, consisting of multiple segments, that performs tasks by interacting with its environment. They are also commonly referred to as robotic arms. Robot manipulators are extensively used in the industrial manufacturing sector and also have many other specialized applications. The study of robot manipulators involves dealing with the positions and orientations of the several segments that make up the manipulators [43, 44]. This module introduces the basic concepts that are required to describe these positions and orientations of rigid bodies in space and perform coordinate transformations. Manipulators are composed of an assembly of links and joints. Links are defined as the rigid sections that make up the mechanism and joints are defined as the connection between two links. The device attached to the manipulator which interacts with its environment to perform tasks is called the end-effector. **Figure 6** presents general structure of a manipulator arm, the link number six is the end effector, which can be a gripper, a welding torch, electromagnet, or any other tool/device that is required to perform the intended task [45]. In **Appendix 1** I describe characteristics and joints attached for an industrial arm [46, 47].

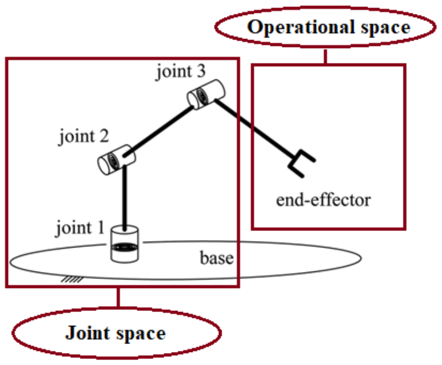


Figure 6. General structure of a manipulator arm

2.3.2. Controlling and programming of industrial robot arms

The robot controller is the module that determines the robot movement that is the pose, velocity and acceleration of the individual joints or the end-effector. This is performed through motion planning and motion control, see **Figure 7**. Motion planning includes the trajectory definition considering its requirements and restrictions, as well as the manipulator dynamic features. The robot operates in the joint space but the motion is normally programmed in the Cartesian space due to easier comprehension for humans. Therefore, the controller has to solve the inverse geometric problem (the calculation of the joint states from the end-effector state) to achieve a desired state in the Cartesian space for the end-effector, then the controller must calculate the proper current values to drive the joint motors in order to produce the desired torques in the sampling moments [48]. The industrial robot controllers are designed to provide good solutions to this problem within certain subspaces. In closed architectures the user is able to program the robot movement by giving the target states and eventually by defining the movement type, e.g., linear, circular etc. The controller then decides how to reach these states by dividing the trajectory into smaller parts defined by interpolation points. The time to perform the movement through a pair of interpolation points is defined by the interpolation frequency, in order to reach the interpolation points closed-loop control at the joint level is performed with much higher frequency. Generally, the control at the joint level is not open to the programmer of industrial robots [49, 50].

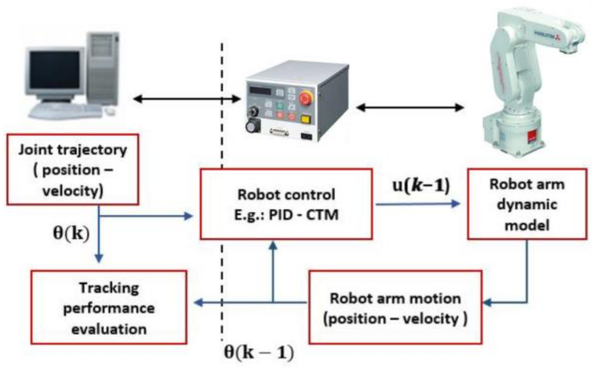

Joint trajectory (position – velocity)
θ(k)
Robot control E.g.: PID - CTM
u(k−1)
Robot arm dynamic model
Tracking performance evaluation
Robot arm motion (position – velocity)
θ(k − 1)

Figure 7. Closed-loop control of an industrial arm

Control algorithm reads the desired joint position/velocity from the reference data file. Also reads actual joint position/velocity of each joint form built-in joint encoders. Then, required joint torques to reduce the error in position and velocity are calculated using the dynamic model (computed torque method) CTM of the manipulator or using any other control law such as PID controller [51].

- User specifies the movements of the tool by a set of via points, and speeds at various path segments. The trajectory generator plans the corresponding joint angle profiles.

- Using sensor feedback, changes can be adapted to manipulator's motion on-line.
- Causes of error: actuator saturation, backlash, gravity, friction. In **Appendix 1** I describe more deeply the closed-loop control scheme at the end-effector level of robot arm [46, 52, 53].

2.4. Trajectory optimization of robot arms

In the area of robotics, the trajectory planning of manipulator arms represents an essential field for focus. The execution of a robot arm's defined task optimizes its trajectory, which can guarantee many benefits such as a reduced cycle time and energy consumption, as well as increased productivity. Basically, the main objective in the trajectory planning field is to compute the desired points that represent the reference input data for the controller of a robot using mathematical techniques [54, 10]. The motion executed from the reference inputs always represents two categories known as forward and inverse kinematics: (1) in free space based on the joint angles, where the motion is limited by the structure constraints, i.e., velocity, torque, and workspace limits; or (2) in task space based on the position and the orientation of the end-effector, where it depends on precision and avoiding obstacles [55, 56]. The approaches generally used are polynomial interpolation function, the bang-bang law, the trapezoid law, etc. [57, 58].

Regarding of pure displacement, we can distinguish the following classes of motion [9, 59]:

In joint space:

- The movement between two points with free trajectory among the points.
- The movement between two points via intermediate points. Specified to avoid obstacles, with free trajectory along the intermediate points.

In operational space:

- The movement between two points with constrained trajectory amongst points "Rectilinear trajectory".
- The movement between two points via intermediate points with trajectory constraint betwixt the intermediate points.

Over the years, researchers studied this field deeply by proposing many methods and solutions to solve trajectory problems for industrial robots [60, 61]. The definition of the optimality concept is divided in many directions [62, 63]. Some scientists focus on a time-optimal trajectory to increase productivity [64, 65], while others work on the smoothness of trajectories [66–68], taking into account reducing cycle time by implementing fast trajectories combined with optimal jerk values in order to reduce the excitation of the resonant frequencies and limit the vibrations of the mechanical system [69, 70]. From the literature, a basic approach is known for generating a trajectory using splines [71, 72], where the virtual points are required to ensure the continuity of the trajectory from the starting point to the endpoint. The development of this approach aims to apply an improved technique in the aspect of motion optimality using B-spline interpolation, based on the calculation of inverse of Jacobian matrix. Regarding time optimality, an approach was proposed for a hyper-redundant robot taking into account the obstacles located in a 3D workspace [73, 74]. It aims to minimize the cycle time during the execution of required tasks, regarding trajectory optimization for robots in terms of energy consumption and minimizing joint torque. Other researchers described a new scheme to determine the trajectory of a redundant robot arm with the purpose of minimizing the total energy consumption [63]. In order to optimize both the energy consumption and the time required for executing a trajectory, many researchers have elaborated new methods based on a fuzzy logic algorithm, a genetic algorithm, or an ant colony algorithm [75, 76]. By using a genetic algorithm, a contribution was proposed to optimize the torque applied at the joints of the robot [77, 78]. We can also cite the second contribution for the same target, which uses a unified quadratic-programming-based dynamic system [79], as well as the role of neural networks for the optimized dynamics of redundant robots [80]. Most of the literature in motion planning features deals with point-to-point applications in free space without any obstacles, where the starting and ending points of the end-effector are predefined. The main purpose of this literature analysis was to guarantee a deeper understanding of the path planning field so that researchers could find an optimal solution without any constraints. Further, time and energy consumption presents the most important factors for evaluation [81].

2.5. Artificial Intelligence and optimisation methods

Since 2016 many leading robot manufacturers as Fanuc and KUKA have succeeded to apply AI and Internet of Things (IoT) technology in different automation systems [82]. Application of AI and Machine Learning (ML) in collaborating robots has an essential target resides in the capability of many robots not only working together but also working safely alongside human operators, where they can easily reprogram themselves for new tasks unlike traditional robots, that depend always on the human decision, where increasing the lead time and achieving the higher productivity are the greatest benefits that can be guaranteed with the use of AI by today's manufacturers. AI science is based on theories of mathematics and optimization techniques used to solve complex problems in different sectors. ML is a subset of AI which allows the machine to automatically learn from historic data without programming, where the main purpose of ML is to allow robots to learn from data to accommodate to the needs of environment. Otherwise the goal of AI is to make a smart computer system enabling stimulation as humans to solve complex problems [83]. The emergence of AI and ML sciences in robotics field has created a new concept known as robotic learning [84] based on the concept of cooperation between machines and industrial robots, by making decision as operators and learn from previous experiences.

2.5.1. Artificial Intelligence algorithms specified for industrial robots

Thanks to the sensors installed in the robotic cell systems, AI makes it possible to capture the energy consumption of individual machines, to analyze the maintenance cycles and then to optimize them during the next step. It can also indicate when operating data are faulty. As the amount of data increases, the system optimizes its efficiency and allows more accurate predictions. Using AI techniques, industrial robots can for instance identify objects on conveyor chains with image recognition, sort them automatically, and identify product defects in terms of precision and quality.

In this part, I would like to overview the main algorithms and models of AI that can be used in the research to developing the functioning of an industrial robot in controlling task and optimization of trajectories. In **Appendix 2**, I describe AI methods and their possible use for a robotic arm [85, 86, 87, 88, 89].

2.5.2. Search algorithms

Search algorithms are universal problem-solving methods for robots to solve a specific problem and provide the best result, is represented as a tree; the root of the search tree is the root node which is corresponding to the initial state or the starting step. Search algorithms are divided in two main categories [90]: 1) - Blind search algorithms where the search algorithm starts to examine each node of the tree without any information about the search space until achieve the goal node. 2) - Informed search algorithms where a problem information is known that helps the search to find a solution more efficiently than a blind search strategy, it is called also a Heuristic search. A heuristic is a strategy presented with a cost function that do not guarantee best solution but guarantees to find a good solution in reasonable time. It is used as an additive technique to different algorithms like genetic algorithm to find the optimal solution.

The efficiency of these algorithms approved by comparing the four essential properties:

- **Completeness:** A search algorithm is complete if at least one solution exists for any random input.
- **Optimality:** Optimal solution where the solution found is guaranteed to be the best solution (lowest path cost).

- **Time Complexity:** Time complexity is a measure of algorithmic steps for an algorithm to complete its task.
- **Space Complexity:** It is the maximum storage space required at any point during the search, as the complexity of the problem.

In industrial robotics the application of search algorithms can be a good strategy to find the optimal solution by scheduling the steps of any process. From literature review, I can cite different search algorithms that deal with the control and the optimization of a robotic arm.

1. Tabu-search algorithm

Tabu-search (TS) algorithm was first proposed by Fred Glover in 1986 [91]. This approach is a combination of optimization and search methods based on converging to the best available neighborhood solution point [92]. In other words, the principle idea is to move to the best point of local search space in term of cost function, even though it is worse than the current best solution point. TS method is a local search approach that includes two memories, short and long-term memory. The short-term memory prevents the reversal of the recent moves and steps of a robotic arm, otherwise the long-term memory reinforces attractive components, forcing the algorithm to move towards more preferable solutions, the tabu list helps the algorithm to move out from local optimum and to reach the global one [93].

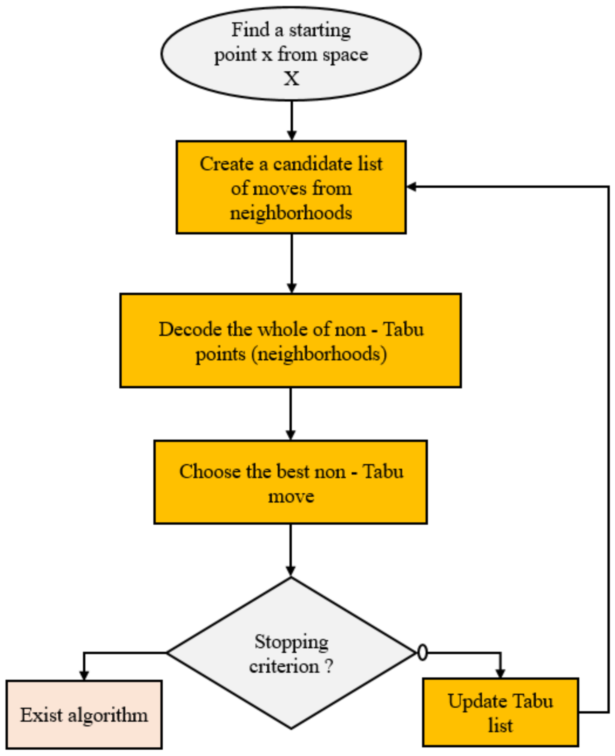

Find a starting point x from space X
Create a candidate list of moves from neighborhoods
Decode the whole of non - Tabu points (neighborhoods)
Choose the best non - Tabu move
Stopping criterion ?
Exist algorithm
Update Tabu list

Figure 8. Tabu-search flow chart

The flowchart in **Figure 8** describes TS algorithm in five steps. Assume that *X* is a total search space and *x* is a solution point sample, and *f(x)* is cost function. First, we choose *x* from space *X* to start the process, then a candidate list of non-Tabu moves in neighborhood is created *N*. All solution points are decoded and then we find the best local *xwinner* from *N*. In the last step, the algorithm exits if the stopping criterion is satisfied. If not, *x* = *xwinner*, the Tabu List will be updated, and it returns to the first step of algorithm.

Usually it is used for task scheduling application for pick and place or disassembling processes [92], as well as to find the optimal trajectory for the manipulator arm.

2. Decision Tree Search algorithm

Decision Tree Search (DTS) is a flowchart structure in which each internal node represents a “test” on an attribute, each branch represents the output of the test, a class label which describes the decision taken after testing is known as a leaf node, where classification rules are presented as the paths from the root to leaf [94, 95]. Mostly used in operations research, specifically in decision analysis, to help identify a strategy, most likely to reach a goal and to execute a trajectory or schedule a steps to pick and place, also in the cooperative robotic systems, where DTS can be used to determine which robot is suitable for a specified action. **Figure 9** illustrates the global scheme of DTS including rules and actions of the process where each test has a gain and using this gain the cost function can be calculated to determine which path from the tree is optimal.

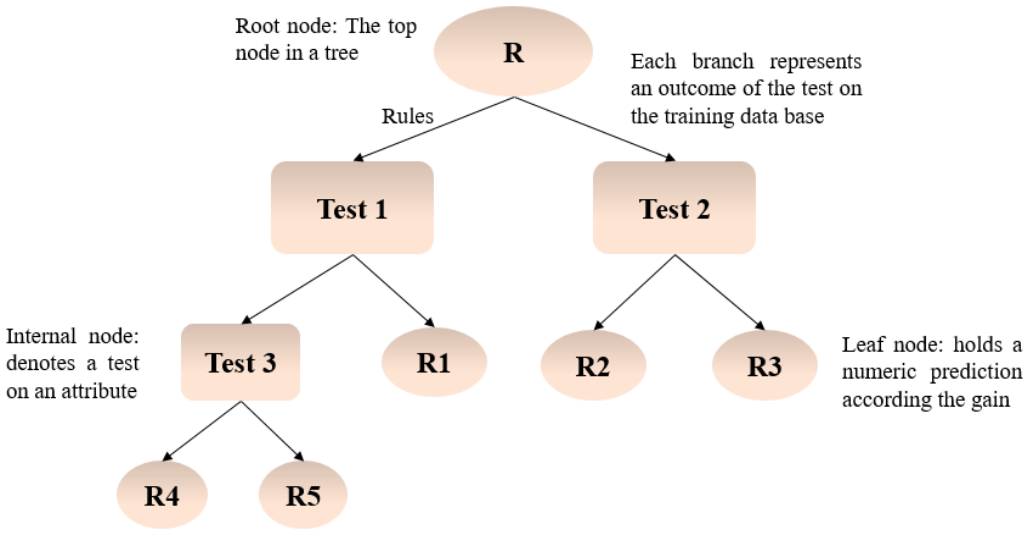
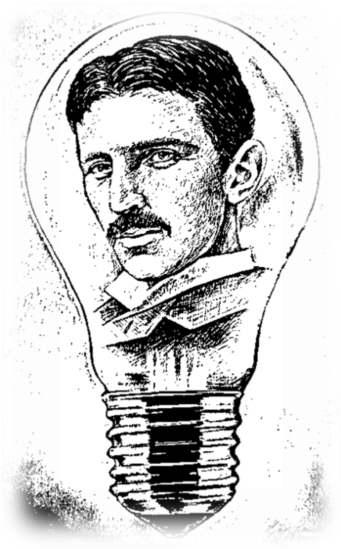

Root node: The top node in a tree
R
Each branch represents an outcome of the test on the training data base
Rules
Test 1
Test 2
Internal node: denotes a test on an attribute
Test 3
R1
R2
R3
Leaf node: holds a numeric prediction according the gain
R4
R5

Figure 9. Decision Tree Search scheme



"A new idea must not be judged by its immediate results"

Nikola Tesla

Research and Innovation

This section demonstrates my research and results related to cooperating industrial robots topic, where the working path is divided into several theses. In each thesis, I applied different tools and methodologies which were already highlighted in the research methodologies and literature review background sections.

3. COOPERATIVE ROBOT ARMS INNOVATION

The essential target of the dissertation presents in the development of cooperating robots concept creating an assembly line that includes a few robot manipulators, these robots as members of a multi-agent system can help each other and cooperate to finalize the appropriate line tasks using efficient algorithms. If some malfunction or another problem occurs in the production line, the robots can reconfigure themselves and reorganize the steps of the same task.

The process is presented in two scenarios:

I. Scenario: Presents the normal process or the static mode including tasks, processes that take place as were planned. In this scenario the planning methods of AI are used for creating the optimal scheduling for the tasks taking into the parallel work of the robots. These AI methods mainly mean search algorithms and first order logic.

II. Scenario: Presents the disturbed process including some problems, random happenings, errors, etc., when intelligent reordering, rescheduling of steps needed. In this scenario the AI is intended for finding quasi-optimal strategies for continuing the work after the random happening. In this case the mentioned AI methods are completed with probability algorithms.

Thanks to Robotic simulation software, the model of the assembly line can be executed in virtual environment by designing all the equipment's needed in the process using CAD software, robotic simulation software allow us to control the steps of each task executed by the robots precisely using one of programming language (Lua, Python, C++, ..).

As we can create larger systems virtually only using simulation, smaller tasks were created to demonstrate the robot cooperation in reality, where in the real task we intended to use two Mitsubishi robots RV-2AJ and RV-2SD from our Institutes see **Figure 10**.

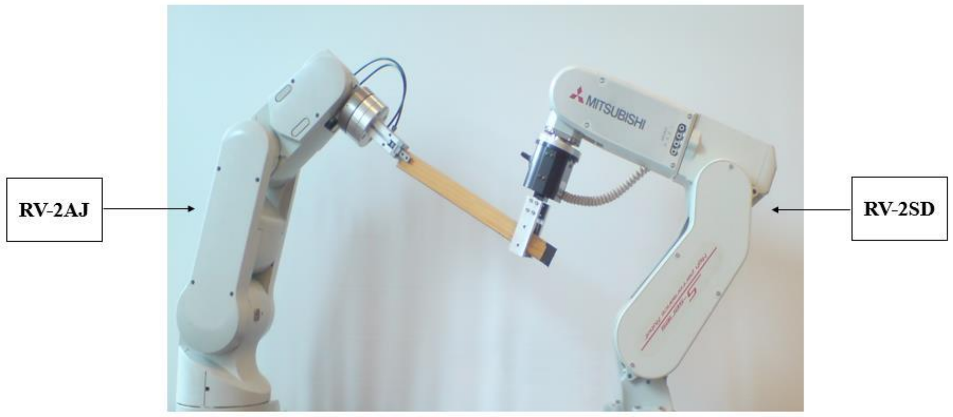


Figure 10. Cooperating industrial robots in real environment [96, 97]

3.1. Characteristics of RV-2AJ and RV-2SD Mitsubishi robot arms

RV-2AJ robot arm. The Mitsubishi RV-2AJ arm assembly from Institute of Information Science and Technology includes the following equipment: robot arm, controller, teaching box to programming the robot manually, power/signal cables, programming instructions documentation, COSIROP software to programming automatically, a compressor to command the gripper as shown in **Figure A.4 - A.8** and **Tables A.2 - A.4** from **Appendix 3** and **Figures 11 - 12** describe the technical measurements of RV-2AJ robot arm [96].

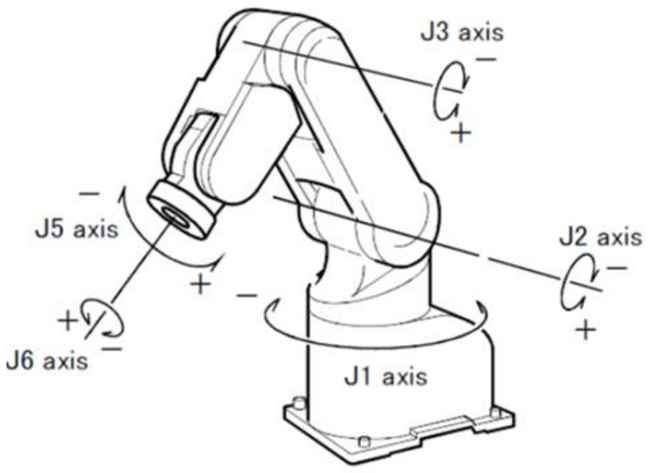


Figure 11. Axes of RV-2AJ arm [96]

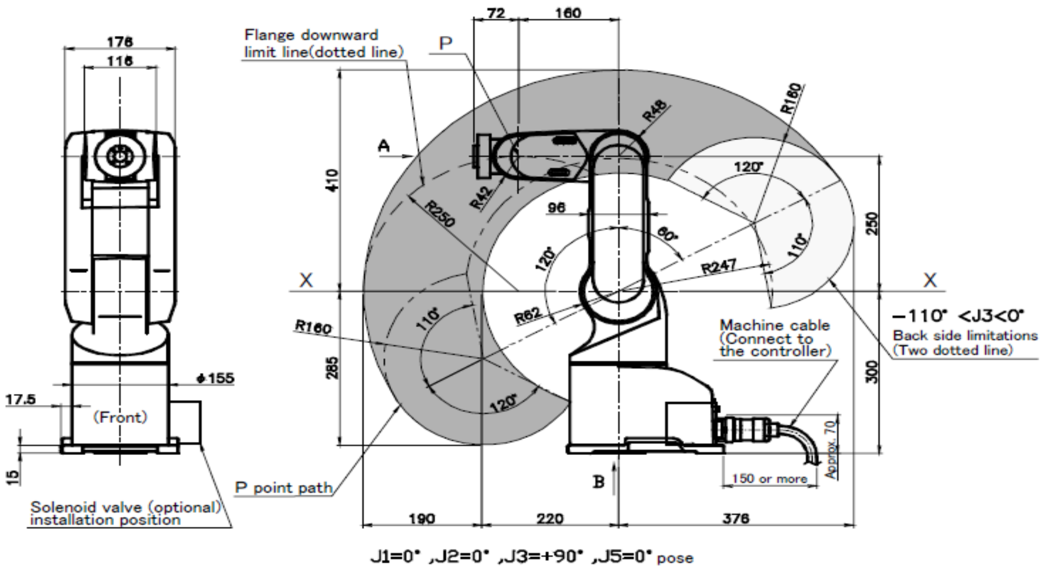

Flange downward limit line(dotted line)
P
Solenoid valve (optional) installation position
P point path
Machine cable (Connect to the controller)
−110° <J3<0°
Back side limitations (Two dotted line)
(Front)
150 or more
Approx. 70
J1=0° ,J2=0° ,J3=+90° ,J5=0° pose

Figure 12. Industrial design of RV-2AJ arm [96]

RV-2SD robot arm. As the previous robot, RV-2SD arm assembly from Institute of Logistics has the same equipment with some improved characteristics as presented in **Figures 13 – 14** and **Figures A.9 – A.12** and **Tables A.5 – A.7** [97] in **Appendix A3.**

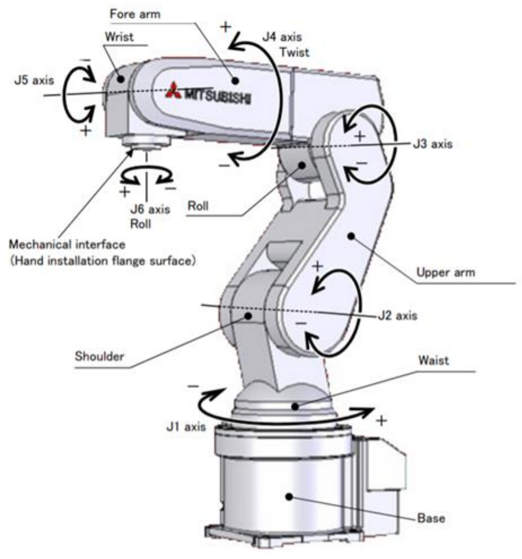


Figure 13. Axes of RV-2SD arm [97]

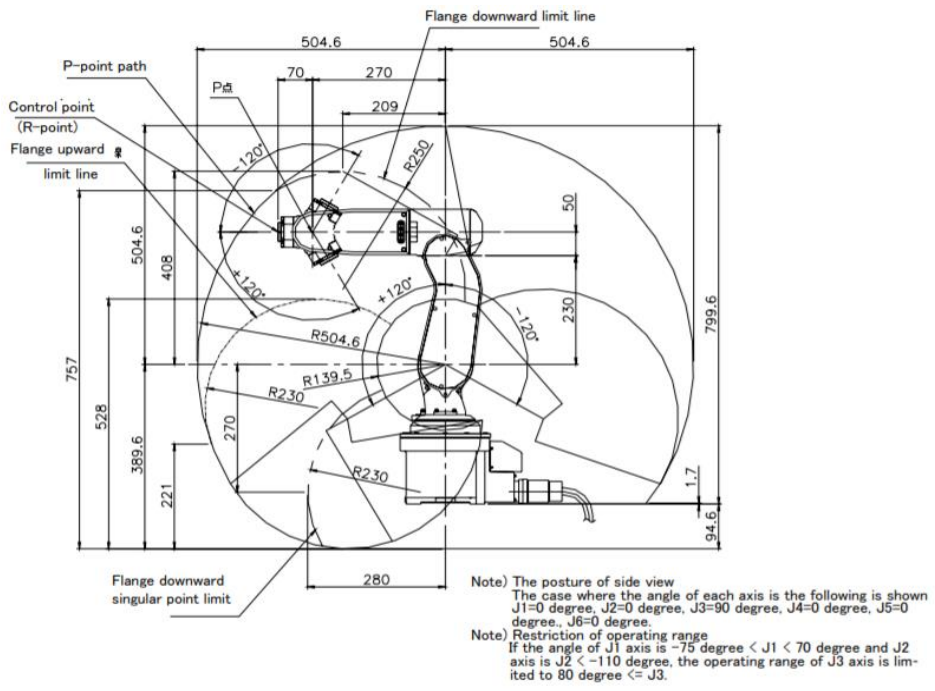

Flange downward limit line
504.6
504.6
P-point path
70
270
P点
Control point
(R-point)
209
Flange upward
limit line
R250
120°
50
504.6
408
+120°
+120°
-120°
230
799.6
757
R504.6
R139.5
R230
528
270
389.6
R230
221
1.7
94.6
Flange downward
singular point limit
280
Note) The posture of side view
The case where the angle of each axis is the following is shown
J1=0 degree, J2=0 degree, J3=90 degree, J4=0 degree, J5=0
degree., J6=0 degree.
Note) Restriction of operating range
If the angle of J1 axis is −75 degree < J1 < 70 degree and J2
axis is J2 < −110 degree, the operating range of J3 axis is lim-
ited to 80 degree <= J3.

Figure 14. Industrial design of RV-2SD arm [97]

3.2. Task application: "Cards house" building

The task aims to create a "house of cards" providing some building elements ("cards") where the "cards" are presented as wooden sheets. The task needs cooperation trivially if there is no some sort of supporting element while to build an A shape from two cards needs two hands, in our task it needs two manipulator arms RV-2AJ arm and RV-2SD arm.

The process is presented in two scenarios:

Normal Scenario: Presents the trivial application where RV-2AJ arm and RV-2SD arm pick two "cards" and place them in slanted way together as an A shape.

Abnormal Scenario: Presents the abnormal process when one of the robot fails to execute its target, therefore the other robot should think in smart way and find a new solution to accomplish the building process. The solution could be using the functioning robot arm Mitsubishi RV-2AJ or RV-2SD by applying an innovative technique using a simple support element.

Using the support element and removing it later needs extra time opposite to cooperating robots, which can prove the advance of robot cooperation in optimization of lead time in any process.

We started the realization first by manufacturing all the elements needed for this operation presented in **Figure 15**.

The components of the "card house" are the following:

- 12 "cards" (slanted "cards") having a special form **Figure 15.b** to create A shape and guarantee the stability of the structure,
- 3 "cards" in horizontal position having rectangular shape, see **Figure 15.c**, located between the levels of the "card house",
- one "card" which is a support element presented in **Figure 15.d** to help fixing the single slanted "cards".

The "cards" were manufactured from wood material using CNC machine.

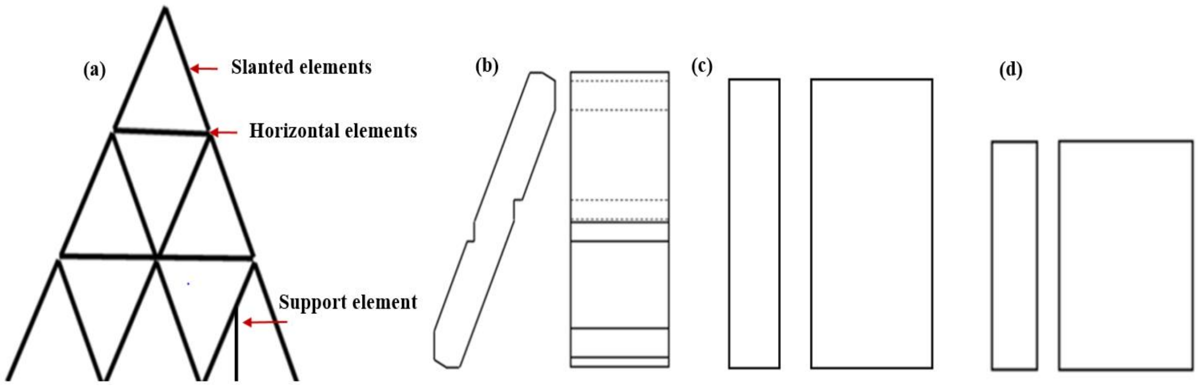


Figure 15. Elements needed for card house building

3.2.1. Task application goals

The task of "card house" building is described in real and virtual environments in order to reach the following points:

- Prove the advance of robots cooperation in optimizing the lead time of task.
- Modeling and simulating the behavior of each robot which require us to perform the kinetic and the dynamic analyses to define the workspace of robots and their singularities, and use different virtual simulation tools.
- Be familiar with robot programming by learning the generating of motion plans of single robot and define the optimal trajectory.

3.3. “Cards house” building in real environment

I started my work by applying the simplest scenario: building a “card house” using just one robot and support element, the robot used in the achievement of task is RV-2AJ arm according to its availability in our laboratory. Unfortunately, the RV-2AJ arm was turned off since many years. To reconfigure it again, I executed the following steps:

- Changing the batteries of RV-2AJ arm and controller seen **Figure 16** and **Figure 17**.

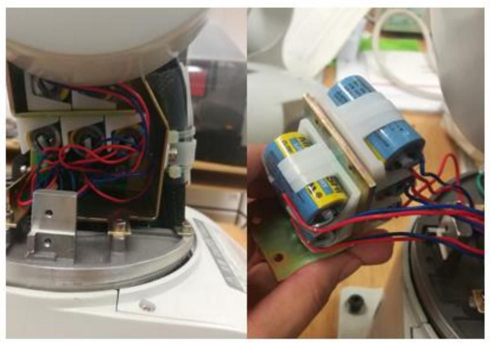


Figure 16. Changing the batteries of RV-2AJ arm

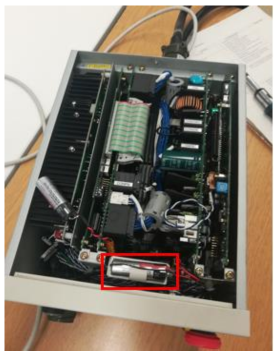


Figure 17. Changing the batteries of controller

- Set the reference points in RV-2AJ arm: Inputting origin data from the packaging of the robot in order to create reference points after that we applied **Mechanical stopper method** which require to release the breaks to define the limits of each joint.
- Connect the compressor to the pneumatic gripper seen **Figure 18**.

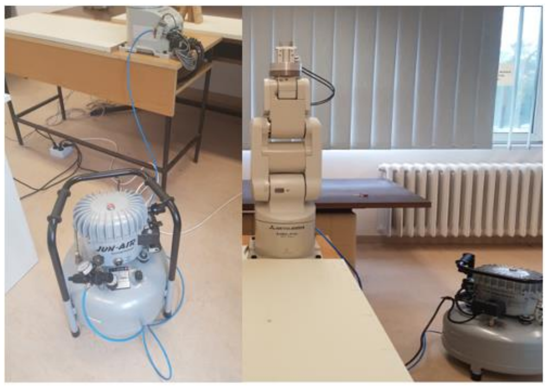


Figure 18. Connection between compressor and the gripper

After realizing all this steps, the robot was capable to read any position and execute many motions types in its workspace that allowed me to start programming and write short programs on computer using its own software COSIROP.

3.3.1. Process of “card house” building

Building house of cards is known task since we were young, this task requires a high precision to make the house stable, the following figures describe the measurement and the design of wooden “cards” and the support element. Cutting and forming the shape of this elements was realized using EMCO CNC machine and wood material as presented in **Figures 19 - 21.**

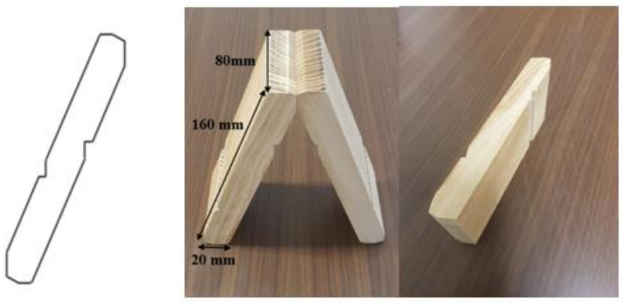


Figure 19. Measurement and design of "card" element

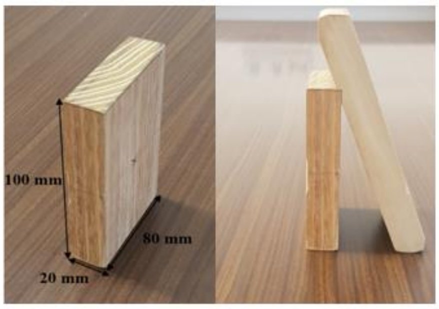


Figure 20. Measurement of support element

Figure 21. CNC machine and its software

Figure 22 illustrates the "house of cards" to be built and **Figure 23** shows Pick and place scenario by RV-2AJ.

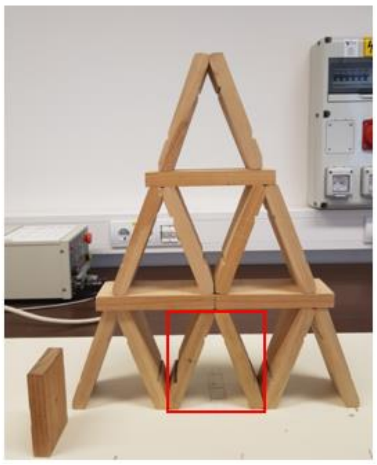


Figure 22. "House of cards" prototype

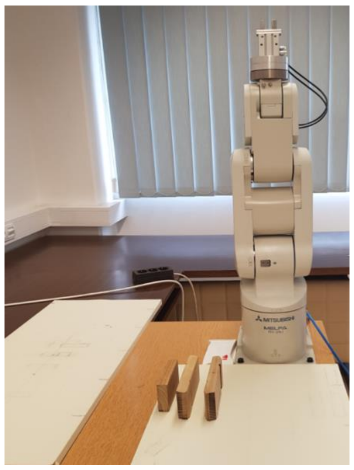


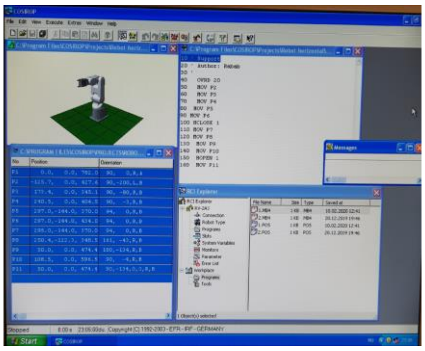
Figure 23. Pick and place scenario by RV-2AJ arm

The building experience task was divided into subtasks starting by realizing the first small "A" shape composed of two "cards" where RV-2AJ arm executes the following scenario to achieve the final process:

- RV-2AJ arm picks the support element and places it in a known position.
- RV-2AJ arm picks the slanted "cards" and places it according to the support element.
- RV-2AJ arm picks a new slanted "cards" and places it according to the previous "cards" forming an A shape.

3.3.2. Motion control of RV-2AJ robot arm

Motion control of an industrial robot arm is executed precisely by adjusting the position and storing its coordinate system in the interface of the controller, the position can be defined – in Joint space where the controller receives joint angles vector containing the angle of each joint and by using Forward kinematics, the position vector is calculated in Cartesian space where the position is defined regarding the coordinate system of the gripper on (X, Y, Z) plus the rotations A, B, C around X, Y and Z respectively. Using Inverse kinematics, the controller can calculate the joint angles needed for each articulation. The motion from point P1 to another point P2 can take different paths, for example linear motion or circular one. **Figure 24** presents the script code writing in MELFA-BASIC IV Language in COSIROP software to control the executed motion automatically respecting the Cartesian positions.



```
10 ' Support
20 ' Author: Rabab
30 '
40   OVRD 20
50   MOV P2
60   MOV P3
70   MOV P4
80  MOV P5
90 MOV P6
100 HCLOSE 1
110 MOV P7
120 MOV P8
130  MOV P9
140  MOV P10
150  HOPEN 1
160  MOV P11
```

```
No   Position                Orientation
P1      0.0,    0.0, 782.0   90,    0,R,A
P2   -115.7,    0.0, 627.6   90,-200,L,R
P3    177.4,    0.0, 145.1   90,  -80,R,A
P4    240.5,    0.0, 404.5   90,   -3,R,A
P5    297.0, -144.0, 370.0   94,    0,R,A
P6    297.0, -144.0, 434.0   94,    0,R,A
P7    295.0, -144.0, 370.0   94,    0,R,A
P8    250.4, -112.3, 348.5   161,  -43,R,A
P9     30.0,    0.0, 474.4   180,-134,R,A
P10   108.5,    0.0, 596.5   90,   -4,R,A
P11    30.0,    0.0, 474.4   90,-134,0,0,R,A
```

Figure 24. The script code writing in MELFA-BASIC IV Language in COSIROP software

3.4. Result of task execution

The results of the previous scenario are presented in **Figure 25**, where we can see that RV-2AJ arm was able to place the three elements in the positions expected but the imprecise contact between two "cards" appeared, the experience was repeated many times even with the correct calibration of RV-2AJ and setting the precise zero position "reference point", this problem arises from the limited freedom of the robot arm and creates instability where we were unable to build up the house of "cards".

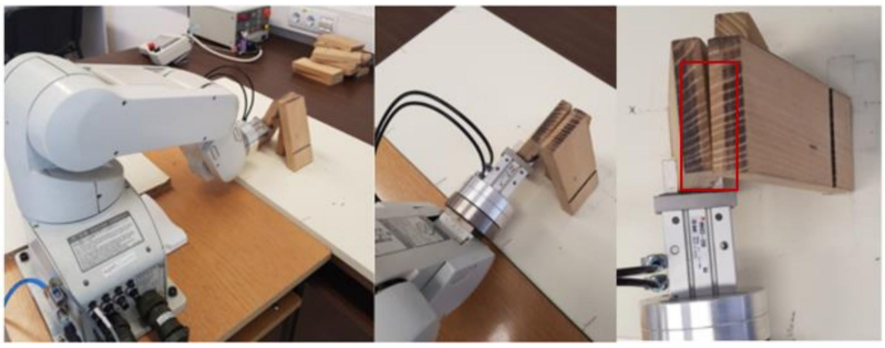


Figure 25. Imprecise contact between two "cards"

3.4.1. Positioning solution for “card house” building task

The imprecise contact between the two slanted “cards” is caused due to the angle problem where RV-2AJ arm has a five-degree of freedom (5Dof; RRR-RR), the missing of the 6th joint makes RV-2AJ robot incapable to execute such positioning and coincide the two slanted “cards” in a good fixation. In order to solve the problem of parallelism precision the rethinking of the robot kinematical possibilities has made. The virtual modelling of the possible orientations of the robot arm revealed an arrangement where the first and the fifth joints axes are parallel. The first joint rotation axis represents the axis of a cylindrical coordinate system, where the chain of the two robot arm segments between the second and fourth parallel axis joints form a triangle structure that is capable to reach any point in a limited region of the cylindrical coordinate system implicit realizing the axis direction Z and radius direction R of the cylindrical coordinate system, where the third axis, the rotation around Z is provided by the first joint of the robot arm. The fourth joint of the robot arm which has axis parallel to second and third joints axes makes possible to set the axis of the fifth joint to parallel to the first joint axis. Due to the parallelism of these axes the gripper and the positioned “cards” angle can be set in any angle with the fifth joint and this angle appears around the first joint axis too, so if the RV-2AJ robot is moved from vertical position to horizontal, the necessary slanted orientation for any “cards” can be provided. This not trivial solution solves the “cards” orientation problem and needs not acquisition of a new robot having 6 Dof.

Trials were made, the solution based on creating a new placement for RV-2AJ where the robot will be changed from normal vertical into horizontal placement. **Figure 26** illustrates the design of the mechanical holder prototype where RV-2AJ arm will be fixed on it.

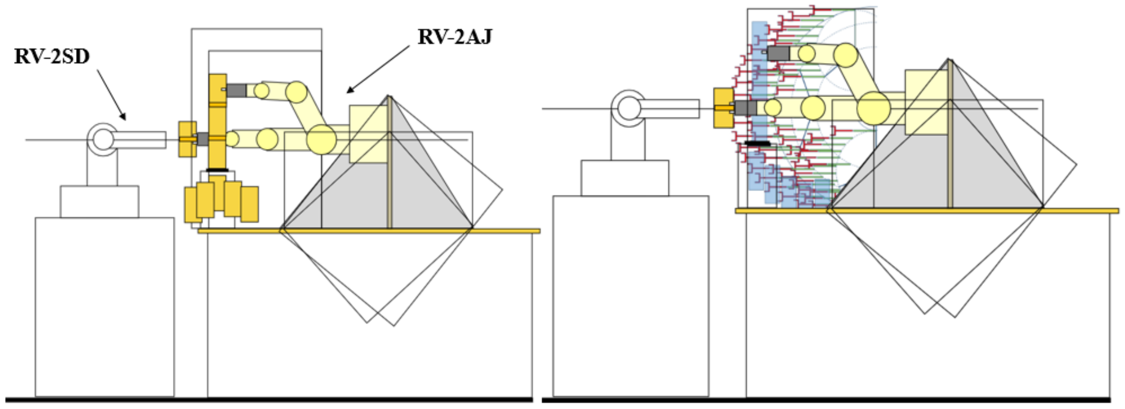


Figure 26. The mechanical holder prototype geometry development

From **Figure 26** is clear that RV-2AJ arm in the new horizontal placement could achieve all the position required for the “house of cards” building, the solution for this problem was based on the optimal formation of the workspace where the arm in the horizontal placement can move and rotate the end effector more precisely to eliminate the orientation error providing stability of the “cards house”. To prove the success of the new placement a simulation test has been performed using SolidWorks software by modeling the whole process and testing the accessing of RV-2AJ gripper to all the “cards”. **Figure 27** shows the “Card house” building by RV-2AJ robot arm in the real environment and in the virtual environment of SolidWorks software [98]. Where it describes the mechanical holder with the new placement of RV-2AJ robot arm, it is clear that the holder is strong enough to carry on such a body.

After finalizing the process of assembling RV-2AJ arm on the mechanical holder, we started thinking on an optimal basement for “card” elements, where robot can pick and place “cards” easily using the predetermined locations, as seen in **Figure 28**. The idea of creating a “card house” basement origin from the following scenario: RV-2AJ will pick a “card” from the holder position and place it in the house of “cards”. The “cards” will be placed vertically in a **holder element** located below **the basement**.

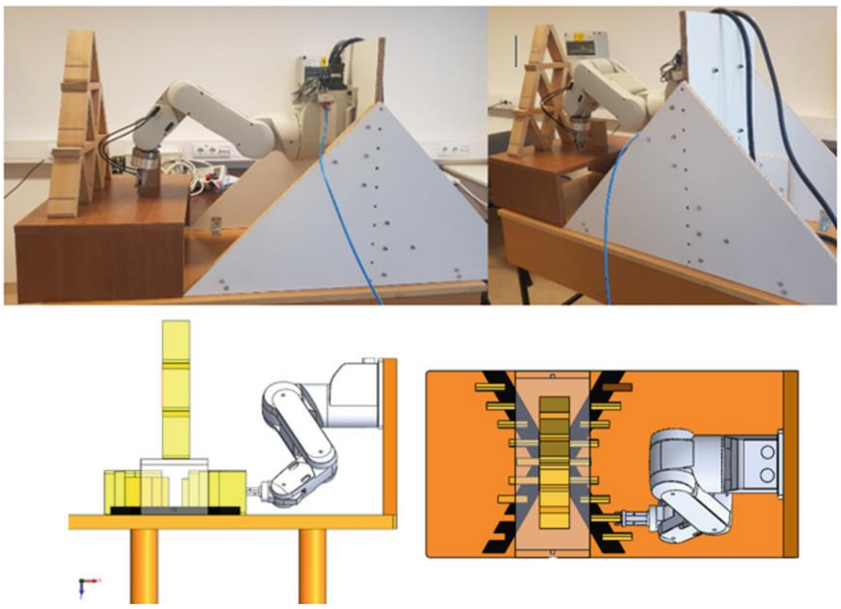


Figure 27. “Card house” building by RV-2AJ robot arm in the real and virtual environment

The holder is realized in two phases:

The first phase: I produced flexible cuprum plate elements with “U” shape for holding the “cards”. This material is perfect due to its enough flexibility to maintain the “card” and to its needed plasticity to form it on “U” shape.

The second phase: In this step a wood surface was produced then the “U” shape holders were fixed on it using bolts, I tried to form an “X” form in order to guarantee the symmetry and also for the future work which involves the cooperation of the second robot RV-2SD arm.

Figure 28 shows the final view of the holder for 16 building elements (one supporting element – 12 slanted “cards” to create the 6 A shapes – 3 horizontals “cards” to create the levels).

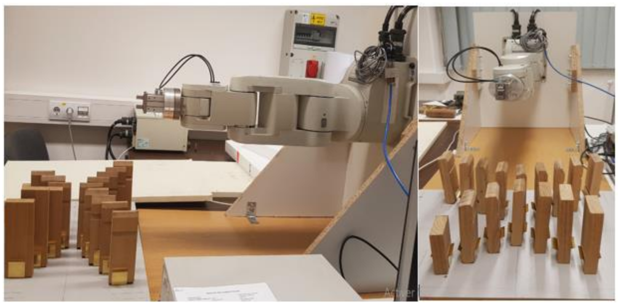


Figure 28. The final view of holder for “card” elements including 16 elements

Figure 29 presents the new “card” basement where the surface of the material was smooth, therefore has been changed to a rough surface using a simple coarse paper on it, this solution is trivial and simple but it guarantees a good stability resulting in friction between the “card” and the surface of basement.

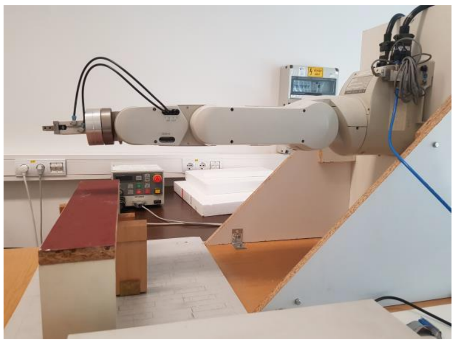


Figure 29. The new “card” basement

The test of pick and place task in the new arrangement was clear after the modification realized, but the realization of the necessary position was always difficult using Trial and error approach thus I decided modelling the new process theoretically and calculate the inverse kinematic model for RV-2AJ in the new environment, getting the model analysis of RV-2AJ arm to facilitate the simulation of the behaviour in the virtual environment using different Robotic simulation tools.

4. MODEL ANALYSIS FOR RV-2SD AND RV-2AJ ROBOT ARMS

The motion control of the RV-2AJ and RV-2SD robots arm in the real environment requires studying the dynamic analysis, in order to model very precisely the process, where the trial and error method took a large time to get familiar with the robot workspace. I am talking about some mathematical representations that allow the modelling in the following points [44]:

- The transformation matrixes between the operational space and the joint space.
- Forward and inverse geometry and kinematics models that express the situation of the terminal organ according to the articular variables of the mechanism and vice versa.
- The Jacobian matrixes which express the velocity of the terminal organ as a function of joint velocities and vice versa.
- Dynamic models defining the equations of motion of the robot, which make it possible to establish relationships between the torques or the forces exerted by the actuators and the positions, velocities, and accelerations of the joints.

4.1. Forward and inverse geometry models

In general, a serial robot is composed of a sequence of $\boldsymbol{n+1}$ links and $\boldsymbol{n}$ joints. The links are assumed to be perfectly rigid. The joints are either revolute or prismatic and are assumed to be ideal (no backlash, no elasticity) [99–101]. Using Denavit-Hartenberg method we can calculate the forward geometry model by calculating the homogenous transformation matrix (See **Appendix 4**).

The homogeneous transformation matrix defining frame R_j relative to the frame R_{j-1} is given:

$$\boldsymbol{H_j^{j-1} = Rot(\mathbf{x},\alpha_j)Trans(\mathbf{x},d_j)Rot(\mathbf{z},\theta_j)Trans(\mathbf{z},r_j)} \qquad \textbf{(1)}$$

$$\boldsymbol{H_j^{j-1}} = \begin{pmatrix} C\theta_j & -S\theta_j & 0 & d_j \\ C\alpha_j S\theta_j & C\alpha_j C\theta_j & -S\alpha_j & -r_j S\alpha_j \\ S\alpha_j C\theta_j & S\alpha_j C\theta_j & C\alpha_j & r_j C\alpha_j \\ 0 & 0 & 0 & 1 \end{pmatrix} \qquad \textbf{(2)}$$

I note that the (3x3) rotation matrix $\mathbf{A_j^{j-1}}$ can be obtained as:

$$\mathbf{A_j^{j-1}} = Rot(\mathbf{x},\alpha_j)Rot(\mathbf{z},\theta_j)$$

4.1.1. Forward geometry for RV-2AJ and RV-2SD robot arms

I present in this part the optimal methods that I have used to get the forward geometric model for our robots. **Figure 30** presents the morphology structure of RV-2SD and RV-2AJ robot arms positioned horizontally, with the frame assignments of rotational joints.

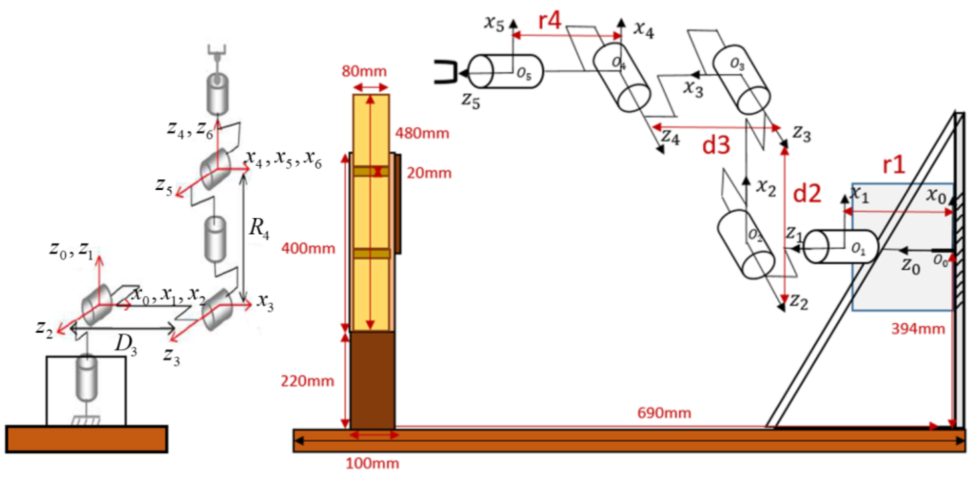


Figure 30. Frames assignments for RV-2AJ and RV-2SD robot arms

a) RV-2AJ arm forward geometry

Modeling robots requires a methodical way to facilitate the description of their morphology. There are several methods. The most common is that of Denavit-Hartenberg [102]. where this method is applicable just for serial structures, RV-2AJ arm is a serial type robot with an open-chain, the structure is composed of six links and 5 rotational joints.

In order to guarantee easy calculation in the process of forward and inverse geometric, it is always better to coincide (R0/R1/R2) and (R4/R5) as presented in **Figure 31**, where $r_{01} = r_{45} = 0$.

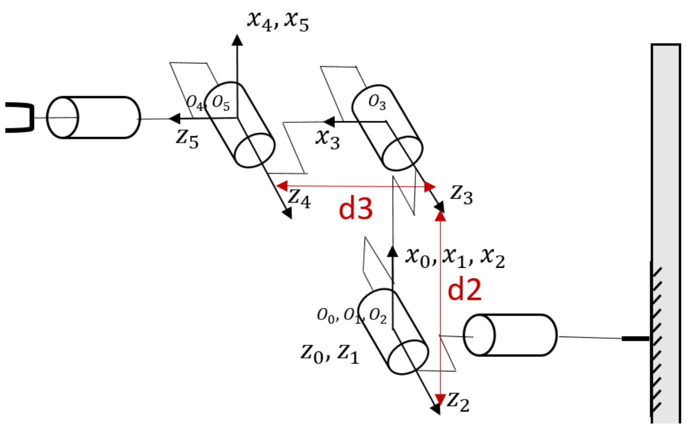


Figure 31. Simplification of frames assignments for RV-2AJ arm

Table 1. Denavit-Hartenberg parameters for RV-2AJ robot arm

Joints *j*	**Frame**	θ_j	d_j	α_j	r_j
1	$O_0 - O_1$	θ_1	0	0	r_{01}
2	$O_1 - O_2$	θ_2	0	$-\pi/2$	0
3	$O_2 - O_3$	θ_3	d_{23}	0	0
4	$O_3 - O_4$	θ_4	d_{34}	0	0
5	$O_4 - O_5$	θ_5	0	$\pi/2$	r_{45}

According to **Table 1** which presents Denavit-Hartenberg parameters for RV-2AJ robot arm, I calculate the Homogeneous Transformation Matrix between two frames $\boldsymbol{j-1}$ to $\boldsymbol{j}$ in (**Appendix 4**). The homogeneous transformation matrix from the base to the end effector is the following:

$$\boldsymbol{H_5^0 = H_1^0 \times H_2^1 \times H_3^2 \times H_4^3 \times H_5^4} \qquad \textbf{(3)}$$

Where I got, as a result, a matrix of (4×4), the fourth column represents the position vector which presents the position of the end effector when I execute defined joint rotations.

Robot data: $d_{23} = 250\,\text{mm}$ and $d_{34} = 160\,\text{mm}$.

I get the following matrix:

$$H_5^0 = \begin{pmatrix} s_x & n_x & a_x & P_x \\ s_y & n_y & a_y & P_y \\ s_z & n_z & a_z & P_z \\ 0 & 0 & 0 & 1 \end{pmatrix}$$

The position vector is the following:

$$P_x = 250\text{C1C2} + 160\text{C1C2C3} - 160\text{C1S2S3}$$
$$P_y = 250\text{C2S1} + 160\text{C2C3S1} - 160\text{S1S2S3}$$
$$P_z = 250\text{S2} + 160\text{C2S3} + 160\text{C3S2}$$

The orientation matrix is the following:

$$\mathbf{s_X} = \text{C5(C4(C1C2C3 - C1S2S3) - S4(C1C2S3 + C1C3S2)) - S1S5}$$
$$\mathbf{s_Y} = \text{C5(C4(C2C3S1 - S1S2S3) - S4(C2S1S3 + C3S1S2)) + C1S5}$$
$$\mathbf{s_Z} = \text{C5(C4(C2S3 + C3S2) - S4(S2S3 - C2C3))}$$
$$\mathbf{n_X} = -\text{S5(C4(C1C2C3 - C1S2S3) - S4(C1C2S3 + C1C3S2)) - C5S1}$$
$$\mathbf{n_Y} = \text{C1C5 - S5(C4(C2C3S1 - S1S2S3) - S4(C2S1S3 + C3S1S2))}$$
$$\mathbf{n_Z} = -\text{S5(C4(C2S3 + C3S2) - S4(S2S3 - C2C3))}$$
$$\mathbf{a_X} = -\text{C4(C1C2S3 + C1C3S2) - S4(C1C2C3 - C1S2S3)}$$
$$\mathbf{a_Y} = -\text{C4(C2S1S3 + C3S1S2) - S4(C2C3S1 - S1S2S3)}$$
$$\mathbf{a_Z} = -\text{C4(S2S3 - C2C3) - S4(C2S3 + C3S2)}$$

b) RV-2SD arm Forward geometry

RV-2SD robot arm is a serial type robot with an open-chain, the structure is composed of seven links and 6 rotational joints. With the application of the same method presented earlier, we coincide (R0/R1/R2) and (R4/R5/R6) as presented in **Figure 32**, where $\boldsymbol{r_{01}} = \boldsymbol{r_{45}} = 0$, then we determine Denavit-Hartenberg parameters for RV-2SD robot arm according to **Table 2**, where we calculate the Homogeneous Transformation Matrix between two frames $\boldsymbol{j-1}$ to $\boldsymbol{j}$.

$$\boldsymbol{H_6^0 = H_1^0 \times H_2^1 \times H_3^2 \times H_4^3 \times H_5^4 \times H_6^5}$$

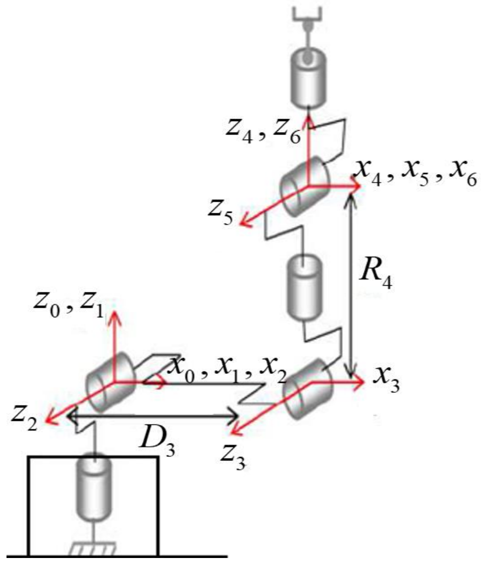


Figure 32. Simplification of frames assignments for RV-2SD arm

Table 2. Denavit-Hartenberg parameters for RV-2SD robot arm

Joints j	**Frame**	$\boldsymbol{\theta_j}$	$\boldsymbol{d_j}$	$\boldsymbol{\alpha_j}$	$\boldsymbol{r_j}$
1	$O_0 - O_1$	$\boldsymbol{\theta_1}$	**0**	**0**	**0**
2	$O_1 - O_2$	$\boldsymbol{\theta_2}$	**0**	$\pi/2$	**0**
3	$O_2 - O_3$	$\boldsymbol{\theta_3}$	**D3=** d_{23}	**0**	**0**
4	$O_3 - O_4$	$\boldsymbol{\theta_4}$	**0**	$-\pi/2$	**R4=** $\boldsymbol{r}_{34}$
5	$O_4 - O_5$	$\boldsymbol{\theta_5}$	**0**	$\pi/2$	**0**
6	$O_5 - O_6$	θ_6	**0**	$-\pi/2$	**0**

we get the following matrix:

$$H_6^0 = \begin{pmatrix} s_x & n_x & a_x & P_x \\ s_y & n_y & a_y & P_y \\ s_z & n_z & a_z & P_z \\ 0 & 0 & 0 & 1 \end{pmatrix}$$

The position vector is the following:

$$\boldsymbol{P}_x = -C1(S23R4 - C2D3)$$
$$\boldsymbol{P}_y = -S1(S23R4 - C2D3)$$
$$\boldsymbol{P}_y = C23R4 + S2D3$$

The orientation matrix is the following:

$$\mathbf{s_x} = C1(C23(C4C5C6 - S4S6) - S23S5C6) - S1(S4C5C6 + C4S6)$$
$$\mathbf{s_y} = \mathrm{S1(C23(C4C5C6 - S4S6) - S23S5C6) + C1(S4C5C6 + C4S6)}$$
$$\mathbf{s_z} = \mathrm{S23(C4C5C6 - S4S6) + C23S5C6}$$
$$\mathbf{n_x} = C1(-C23(C4C5S6 + S4C6) + S23S5S6) + S1(S4C5S6 - C4C6)$$

$\mathbf{n_y} = S1(-C23(C4C5S6 + S4C6) + S23S5S6) - C1(S4C5S6 - C4C6)$

$\mathbf{n_z} = -S23(C4C5S6 + S4C6) - C23S5S6$

$\mathbf{a_x} = -C1(C23C4S5 + S23C5) + S1S4S5$

$\mathbf{a_y} = -S1(C23C4S5 + S23C5) - C1S4S5$

$\mathbf{a_z} = -\mathrm{S23C4S5+C23C5}$

where: C23 = $\cos(\theta_2 + \theta_3)$ and S23 = $\sin(\theta_2 + \theta_3)$, R4= 270mm, D3=230mm

4.1.2. Inverse geometry for RV-2AJ and RV-2SD robot arms

We have already seen that the forward geometry model of a robot allows us to give the position of the end effector, and this is done from the calculation of operational coordinates based on joint coordinates. The problem inverse consists of calculating the joint coordinates corresponding to the situation data of the end effector. When it exists, the explicit form which gives all possible solutions called the inverse geometry model. Calculating the inverse geometry model of RV-2AJ and RV-2SD robot arms, I have used Paul's method [103] because it allows us to treat each particular case separately and suitable for most industrial robots including our robots.

a) Inverse geometry for RV-2AJ robot arm

As seen previously, the homogeneous transformation matrix is as follow:

$$H_5^0 = H_1^0(\theta_1) \times H_2^1(\theta_2) \times H_3^2(\theta_3) \times H_4^3(\theta_4) \times H_5^4(\theta_5) \qquad \textbf{(4)}$$

We suppose that we have the desired position U_0 where:

$$H_5^0 = \begin{pmatrix} s_x & n_x & a_x & P_x \\ s_y & n_y & a_y & P_y \\ s_z & n_z & a_z & P_z \\ 0 & 0 & 0 & 1 \end{pmatrix} = U_0$$

We search to solve the following system of equations:

$$U_0 = H_1^0(\theta_1) \times H_2^1(\theta_2) \times H_3^2(\theta_3) \times H_4^3(\theta_4) \times H_5^4(\theta_5) \qquad \textbf{(5)}$$

➢ **Calculation of ($\theta_1, \theta_2, \theta_3$)**

To find the solutions of **eq 5**, Paul proposed a method which consists in successively pre-multiplying the two members of **eq 4** by the matrices H_{j-1}^{j} for ***j*** varying from 1 to 4, operations which make it possible to isolate and to identify one after the other the joint variables we are looking for. For our robot with five degrees of freedom, we proceed as follows:

We multiply **eq 4** by H_0^1 we get:

$$\mathbf{H_0^1(\theta_1) \times U_0 = H_2^1(\theta_2) \times H_3^2(\theta_3) \times H_4^3(\theta_4) \times H_5^4(\theta_5)} \qquad \textbf{(6)}$$

where:

$$\mathbf{H_0^1(\theta_1) \times H_1^0(\theta_1) = I}$$

From both sides we have:

$$-C1 \times P_x - S1 \times P_y = 250 \times C2 - 160 \times S2 \times S3 + 160 \times C2 \times C3 \qquad \textbf{(7)}$$

$$S1 \times P_x - C1 \times P_y = 0 \qquad \textbf{(8)}$$

$$-P_z = 250 \times S2 - 160 \times C2 \times S3 + 160 \times C3 \times S2 \qquad \textbf{(9)}$$

From **eq 8** we can calculate $\boldsymbol{\theta_1}$, where:

$$C1 \times P_y = S1 \times P_x$$

$$\theta_1 = atan2(P_y, P_x)$$

$$\theta_1 = 0 \text{ or } \theta_1^{'} = \theta_1 + \pi$$

We complete the same process for $\boldsymbol{\theta}_2$ by multiplying **eq 6** by H_1^2 from both sides:

$$\mathbf{H_1^2(\theta_2) \times H_0^1(\theta_1) \times U_0 = H_3^2(\theta_3) \times H_4^3(\theta_4) \times H_5^4(\theta_5)} \qquad \textbf{(10)}$$

We got:

$$\theta_2 = atan2(S2, C2)$$

where:

$$C2 = \frac{2 \times B3 \times B2 - B1 \times \xi \times \sqrt{B1^2 + B2^2 - B3^2}}{B1^2 + B2^2}$$

$$S2 = \frac{B3 \times B1 - B2 \times \xi \times \sqrt{B1^2 + B2^2 - B3^2}}{B1^2 + B2^2} \qquad \xi = \pm 1$$

and:

$$B1 = 2(250P_z)$$

$$B2 = 2(250)(C1 \times P_x + S1 \times P_y)$$

$$B3 = (160^2) - P_z{}^2 - (-C1 \times P_x - S1 \times P_y)^2 - (250)^2$$

Knowing $\boldsymbol{\theta}_2$, we can calculate $\boldsymbol{\theta}_3$ where:

$$\theta_3 = atan2(S3, C3)$$

where:

$$C2 = \frac{S2 \times P_{\mathbf{z}} - C2 \times (C1 \times P_x + S1 \times P_y) - 250}{160}$$

$$S2 = \frac{C2 \times P_{\mathbf{z}} - S2 \times (C1 \times P_x - S1 \times P_y)}{160}$$

- **Calculation of (θ_4, θ_5)**

For calculating θ_4 and θ_5 which represent orientation angles, we use the orientation matrixes.

$$\begin{bmatrix} i & j & k \end{bmatrix} = A_5^0 \qquad \textbf{(11)}$$

where A_5^0 is the orientation matrix which relies the base and the end effector which can be taken from the DGM H_5^0, by multiplying both sides of **eq 11** by A_0^3, we get:

$$A_0^3 \times \begin{bmatrix} i & j & k \end{bmatrix} = A_5^3$$

We have:

$$A_5^3 = \begin{pmatrix} C4C5 & -C4S5 & -S4 \\ C5S4 & -S4S5 & C4 \\ -S5 & -C5 & 0 \end{pmatrix}$$

After the implementation and simplifying we get the following:

$$\theta_4 = atan2(S4, C4)$$

where:

$$C4 = \mathrm{Cos}(\theta_2 + \theta_3)H_{\mathbf{z}} + \mathrm{Sin}(\theta_2 + \theta_3)(-C1 \times H_{\mathbf{x}} - S1 \times H_{\mathbf{y}})$$
$$S4 = \mathrm{Cos}(\theta_2 + \theta_3)(-C1 \times H_{\mathbf{x}} - S1 \times H_{\mathbf{y}}) - \mathrm{Sin}(\theta_2 + \theta_3)H_{\mathbf{z}}$$

and we get:

$$\theta_5 = atan2(S5, C5)$$

where:

$$C5 = -S1 \times G_{\mathbf{x}} + C1 \times G_{\mathbf{y}}$$
$$S5 = -S1 \times F_{\mathbf{x}} + C1 \times F_{\mathbf{y}}$$

where the desired orientation is: $\begin{bmatrix} F & G & H \end{bmatrix}$

b) Inverse geometry for RV-2SD robot arm

With the same concept, we applied Paul's method for calculating the inverse geometric of RV-2SD, we search to solve the following system of equations:

$$U_0 = H_1^0(\theta_1) \times H_2^1(\theta_2) \times H_3^2(\theta_3) \times H_4^3(\theta_4) \times H_5^4(\theta_5) \times H_6^5(\theta_6)$$

by successively pre-multiplying the two members of **eq 12** by the matrices H_{j-1}^{j} for $\boldsymbol{j}$ varying from 1 to 5. The succession of equations allowing the calculation of all $\boldsymbol{q}_j$ as follows:

$$U_0 = H_1^0(\theta_1) \times H_2^1(\theta_2) \times H_3^2(\theta_3) \times H_4^3(\theta_4) \times H_5^4(\theta_5) \times H_6^5(\theta_6) \quad \textbf{(12)}$$
$$H_0^1(\theta_1) \times U_0 = H_2^1(\theta_2) \times H_3^2(\theta_3) \times H_4^3(\theta_4) \times H_5^4(\theta_5) \times H_6^5(\theta_6) \quad \textbf{(13)}$$
$$H_1^2(\theta_2) \times U_1 = H_3^2(\theta_3) \times H_4^3(\theta_4) \times H_5^4(\theta_5) \times H_6^5(\theta_6) \quad \textbf{(14)}$$
$$H_2^3(\theta_3) \times U_2 = H_4^3(\theta_4) \times H_5^4(\theta_5) \times H_6^5(\theta_6) \quad \textbf{(15)}$$
$$H_3^4(\theta_4) \times U_3 = H_5^4(\theta_5) \times H_6^5(\theta_6) \quad \textbf{(16)}$$
$$H_4^5(\theta_5) \times U_4 = H_6^5(\theta_6) \quad \textbf{(17)}$$

- **Calculation of $(\theta_1, \theta_2, \theta_3)$**

We already know the position and orientation vectors for the end effector, as follow:

$$\begin{pmatrix} P_x \\ P_y \\ P_z \end{pmatrix} = \begin{pmatrix} C1(-S23R_4 + C2D_3) \\ S1(-S23R_4 + C2D_3) \\ C23R_4 + S2D_3 \end{pmatrix} \qquad \textbf{(18)}$$

These equations do not provide the solution of θ_1, therefore we pre-multiple the two members of **eq 18** by $H_0^1(\theta_1)$, we obtain:

$$\begin{aligned} U(1) &= C1P_x + S1P_y \\ U(2) &= -S1P_x + C1P_y \\ U(3) &= P_z \end{aligned} \qquad \text{and} \qquad \begin{aligned} H(1) &= \text{-S23}R_4 + \text{C2}D_3 \\ H(2) &= 0 \\ H(3) &= C23R_4 + S2D_3 \end{aligned}$$

By identification of U(2) with H(2), we obtain θ_1:

$$\theta_1 = \text{atan2}(\text{P}_y, \text{P}_x)$$
$$\theta_1' = \theta_1 + \pi$$

We complete the same process for θ_2 by multiplying **eq 13** by H_1^2 from both sides:

$$\begin{aligned} U(1) &= C2(C1P_x + S1P_y) + S2P_z \\ U(2) &= -S2(C1P_x + S1P_y) + C2P_z \\ U(3) &= S1P_x - C1P_y \end{aligned} \qquad \text{and} \qquad \begin{aligned} H(1) &= \text{-S2R4 + D}_3 \\ H(2) &= C3R_4 \\ H(3) &= 0 \end{aligned}$$

We can calculate θ_2 and θ_3 by considering the first two equations and by resultant a system of type 6. (see **Appendix 4**)

$$\begin{cases} W \cdot S\theta_j = X \cdot C\theta_i + Y \cdot S\theta_i + Z1 \\ W \cdot C\theta_j = X \cdot S\theta_i - Y \cdot C\theta_i + Z2 \end{cases}$$

$$\begin{cases} \text{-S3R}_4 + \text{D}_3 = \text{C2(C1P}_x + \text{S1P}_y) + \text{S2P}_z \\ \text{C3R}_4 = \text{-S2(C1P}_x + \text{S1P}_y) + \text{C2P}_z \end{cases}$$

$$\begin{cases} \text{R}_4\text{S3} = \text{C2(-C1P}_x - \text{S1P}_y) + (\text{-P}_z)\text{S2} + \text{D}_3 \\ \text{R}_4\text{C3} = \text{S2(-C1P}_x - \text{S1P}_y) - (\text{-P}_z)\text{C2} \end{cases}$$

With Z1 not equal to 0 and Z2 equal to 0. By squaring and sum these two equations we get:

$$B1S\theta_i + B2C\theta_i = B3 \Leftrightarrow B1S\theta_2 + B1C\theta_2 = B3 \Rightarrow type2$$

where:

$$\begin{cases} \text{B1} = 2(\text{Z1.Y} + \text{Z2.X}) \\ \text{B2} = 2(\text{Z1.X} - \text{Z2.Y}) \\ \text{B3} = \text{W}^2 - \text{X}^2 - \text{Y}^2 - \text{Z1}^2 - \text{Z2}^2 \end{cases} \Rightarrow \begin{cases} \text{B1} = -2\text{P}_z\text{D}_3 \\ \text{B2} = -2\text{D}_3(\text{C1P}_x + \text{S1P}_y) \\ \text{B3} = (\text{R}_4)^2 - (\text{D}_3)^2 - (\text{Pz})^2 - (\text{P}_x\text{C1} + \text{P}_y\text{S1})^2 \end{cases}$$

In this case, we have B1, B2, and B3 are not zero: From there we deduce that:

$$C2 = \frac{B2B3 - B1\xi\sqrt{B1^2 + B2^2 - B3^2}}{B1^2 + B2^2}$$

$$S2 = \frac{B1B3 - B2\xi\sqrt{B1^2 + B2^2 - B3^2}}{B1^2 + B2^2}$$

where: $\xi = \pm 1$

We obtain two solutions of the form:

$$\theta_2 = \text{atan2(S2,C2)}$$

Knowing θ_2, we can calculate θ_3 from:

$$S3 = \frac{-P_z S2 - (P_z C1 + P_y S1)C2 + D3}{R_4}$$

$$C3 = \frac{-(P_x C1 + P_y S1)S2 + P_z C2}{R_4}$$

we obtain:

$$\theta_3 = \text{atan2(S3,C3)}$$

➢ **Calculation of ($\theta_4, \theta_5, \theta_6$)**

After the calculation of ($\theta_1, \theta_2, \theta_3$)**,** we are now interested in the equations of the end effector orientation, well we define the orientation transformation matrix as presented earlier :

$$\begin{bmatrix} s & n & a \end{bmatrix} = A_6^0$$

$$A_0^3 \begin{bmatrix} s & n & a \end{bmatrix} = A_6^3$$

we note: $$\begin{bmatrix} F & G & H \end{bmatrix} = \begin{bmatrix} F_x & G_x & H_x \\ F_y & G_y & H_y \\ F_z & G_z & H_z \end{bmatrix} = A_0^3 \begin{bmatrix} s_x & n_x & a_x \\ s_y & n_y & a_y \\ s_z & n_z & a_z \end{bmatrix} = A_6^3(\theta_4, \theta_5, \theta_6)$$

The element of F is written as follow:

$U(1,1) = C23(C1s_x + S1s_y) + S23s_z$

$U(2,1) = -S23(C1s_x + S1s_y) + C23s_z$ The elements of *G* and *H* are obtained from *F* by replacing

$U(3,1) = S1s_x - C1s_y$

$\begin{bmatrix} \mathbf{s_x} & \mathbf{s_y} & \mathbf{s_z} \end{bmatrix}$ by $\begin{bmatrix} \mathbf{n_x} & \mathbf{n_y} & \mathbf{n_z} \end{bmatrix}$ and $\begin{bmatrix} \mathbf{a_x} & \mathbf{a_y} & \mathbf{a_z} \end{bmatrix}$ respectively. Then by equating the elements of $\begin{bmatrix} F & G & H \end{bmatrix} = A_6^3$. The elements of $\mathbf{A_6^3}$ are obtained from H_6^3, which is calculated for the DGM.

- Equating the elements of $\begin{bmatrix} F & G & H \end{bmatrix} = A_6^3$

$$A_6^3 = \begin{pmatrix} C6C5C4 - S6S4 & -S6C5C4 - C6S4 & -S5C4 \\ C6S5 & -S6S5 & C5 \\ -C6C5S4 - S6C4 & S6C5S4 - C6C4 & S5S4 \end{pmatrix}$$

We can determine θ_5 from the (2, 3) elements using an arccosine function. But this solution is not retained, considering that another one using an atan2 function may appear in the next equations;

- Equating the elements of $A_3^4\begin{bmatrix}F & G & H\end{bmatrix} = A_6^4$

The elements of the first column of the left side are written as:

$U(1,1) = C4F_x - S4F_z$
$U(2,1) = -C4F_z - S4F_x$
$U(3,1) = F_y$

The elements of the second and third columns are obtained by replacing $\begin{bmatrix}\mathbf{F_x} & \mathbf{F_y} & \mathbf{F_z}\end{bmatrix}$ them with $\begin{bmatrix}\mathbf{G_x} & \mathbf{G_y} & \mathbf{G_z}\end{bmatrix}$ and $\begin{bmatrix}\mathbf{H_x} & \mathbf{H_y} & \mathbf{H_z}\end{bmatrix}$ respectively. The elements of A_6^4 are obtained from H_6^4, which has already been calculated for the DGM:

$$A_6^4 = \begin{pmatrix} C6C5 & -S6C5 & -S5 \\ S6 & C6 & 0 \\ C6S5 & -S6S5 & C5 \end{pmatrix}$$

From the (2, 3) elements, we obtain a type-2 equation in θ_4:

$$- C4H_z - S4H_x = 0$$

Which gives two solutions:

$$\theta_4 = \text{atan2}(H_z, -H_x)$$
$$\theta_4' = \theta_4 + \pi$$

From the (1, 3) and (3, 3) elements, we obtain a type-3 system of equations in θ_5:

$$- S5 = C4H_x - S4H_z$$
$$C5 = H_y$$

Which the solution is:

$$\theta_5 = atan2(S5,\ C5)$$

Finally, by considering the (2, 1) and (2, 2) elements, we obtain a type-3 system of equations θ_6.

$$S6 = -C4F_z - S4F_x$$
$$C6 = -C4G_z - S4G_x$$

Whose solution is:

$$\theta_6 = atan2(S6,\ C6).$$

The possible solutions for joint angles for RV-2AJ and RV-2SD robot arms are defined in **Figure A.14** and **Figure A.15** (see **Appendix 4**).

4.1.3. Robots arm singularities

The singularity of the robot arm is described as the position that locks the arm, where the gripper can not reach its target, usually when a robot is redundant has more chance to avoid the singularity position.

By examining the IGM solution of RV-2AJ arm and RV-2SD arm, it can be observed that: theoretically, it is defined as the following: when the elements of $atan2$ function are equals to zero, as well as by examining also the determinant of the Jacobian matrix of each robot presented later, which describes the velocity model, calculating the articular position that satisfies $\det(J)=0$. The three case singularities of a regular robot which has a 6 degree of freedom are presented in **Figure A.16. Appendix 4**.

4.2. Forward and inverse kinematics models

4.2.1. Forward kinematic of RV-2AJ and RV-2SD arm

The forward kinematic model of a robot manipulator gives the velocity of the end effector $\dot{\mathbf{X}}$ in terms of the joint velocities $\dot{\mathbf{q}}$. It is written as [88]:

$$\dot{\mathbf{X}} = \mathbf{J(q)}\dot{\mathbf{q}} \tag{19}$$

where $\mathbf{J(q)}$ denotes the $(m\times n)$ Jacobian matrix, which presents the differential displacement of the end-effector in terms of the differential variation of the joint variables**:**

$$\mathbf{J} = \frac{\partial X}{\partial q}$$

The Jacobian matrix has multiple applications in robotics [89, 104]:

- The use of its inverse to numerically compute a solution for the inverse geometric model to compute the joint variables **q** corresponding to a given position of the end-effector **X**.
- The transpose of the Jacobian matrix is used in the static model to compute the necessary joint forces and torques to exert specified forces and moments on the environment.
- The Jacobian matrix is used to determine the singularities and to analyze the reachable workspace of robots.

Two methods exist to calculate the Jacobian matrix are presented in **Appendix 4.**

a) Forward kinematics of RV-2AJ robot arm

It consists of the application theory to calculate the Jacobian matrix ${}^5J_5$ of RV-2AJ arm with five revolute joints. Jacobian matrix of RV-2AJ arm is composed of the following elements each column represents Jacobian vector for joint *k*:

where ${}^5\mathbf{s}_1$: orientation vector around the x-axis, ${}^5\mathbf{n}_1$ orientation vector around the y-axis, ${}^5\mathbf{a}_1$ orientation vector around z-axis from the orientation matrix ${}^1A_5$ as ${}^1P_5$ taken from the transformation matrix H_5^1 (see **Appendix A4**).

b) Forward kinematics of RV-2SD robot arm

With the same previous method, we calculate the Jacobian matrix ${}^{6}J_{6}$ of RV-2SD arm with six revolute joints. Jacobian matrix of RV-2AJ arm is composed of the following elements each column represents Jacobian vector for joint *k* (see **Appendix A4**).

$$
{}^{5}J_{5} = \begin{bmatrix} J_{11} & J_{12} & J_{13} & J_{14} & J_{15} \\ J_{21} & J_{22} & J_{23} & J_{24} & J_{25} \\ J_{31} & J_{32} & J_{33} & J_{34} & J_{35} \\ J_{41} & J_{42} & J_{43} & J_{44} & J_{45} \\ J_{51} & J_{52} & J_{53} & J_{54} & J_{55} \\ J_{61} & J_{62} & J_{63} & J_{64} & J_{65} \end{bmatrix} \qquad {}^{6}J_{6} = \begin{bmatrix} J_{11} & J_{12} & J_{13} & J_{14} & J_{15} & J_{16} \\ J_{21} & J_{22} & J_{23} & J_{24} & J_{25} & J_{26} \\ J_{31} & J_{32} & J_{33} & J_{34} & J_{35} & J_{36} \\ J_{41} & J_{42} & J_{43} & J_{44} & J_{45} & J_{46} \\ J_{51} & J_{52} & J_{53} & J_{54} & J_{55} & J_{56} \\ J_{61} & J_{62} & J_{63} & J_{64} & J_{65} & J_{66} \end{bmatrix}
$$

A. Jacobian matrix using Renaud method

In order to simplify the calculation as mentioned earlier, we can get the simplest Jacobian matrix using the Renaud method [105], in general, the shift frame R_j and the projection frame R leading to a simple matrix ${}^{i}J_{n,j}$ are chosen such that i = integer (n/2) and $j = i + 1$.

Thus, for a six-degree-of-freedom and a five-degree of freedom robots, the simplest Jacobian matrixes are the following respectively ${}^{3}J_{4,6}, {}^{3}J_{4,5}$. And both robots have a spherical wrist, therefore the vectors $L_{4,6}$ and $L_{4,5}$ are zero, and consequently, we get: ${}^{3}J_{6,4} = {}^{3}J_{6}$, ${}^{3}J_{5,4} = {}^{3}J_{5}$ which has the same singularities and characteristics as ${}^{5}J_{5}, {}^{6}J_{6}$.

$$
{}^{3}J_{5} = \begin{bmatrix} 0 & 250S3 & 0 & 0 & 0 \\ 0 & 160+250C3 & 160 & 0 & 0 \\ -250C2+160S2S3-160C2C3 & 0 & 0 & 0 & 0 \\ C2S3+C3S2 & 0 & 0 & 0 & -S4 \\ C2C3-S2S3 & 0 & 0 & 0 & C4 \\ 0 & 1 & 1 & 1 & 0 \end{bmatrix} \qquad {}^{3}J_{6} = \begin{bmatrix} 0 & -R4+S3D3 & -R4 & 0 & 0 & 0 \\ 0 & C3D3 & 0 & 0 & 0 & 0 \\ S23R4-C2D3 & 0 & 0 & 0 & 0 & 0 \\ S23 & 0 & 0 & 0 & S4 & -S5C4 \\ C23 & 0 & 0 & 1 & 0 & C5 \\ 0 & 1 & 1 & 0 & C4 & S5S4 \end{bmatrix}
$$

B. Dimension of the task space of a robot

At a given joint configuration q, the rank r of the Jacobian matrix ${}^{i}J_{n,j}$ is written as J for simplification. It defines the dimension of the accessible task space at this configuration. The number of degrees of freedom of the task space of a robot, **M** is equal to the maximum rank $r_{\max}$, which the Jacobian matrix can have at all possible configurations. Noting the number of degrees of freedom of the robot as **N** (equal to n for serial robots), the following cases are considered [106]:

- if **N = M**, the robot is non-redundant and has just the number of joints required to provide **M** degrees of freedom to the end-effector;

- if **N** > **M**, the robot is redundant of order (**N - M**). It has more joints than required to provide **M** degrees of freedom to the end-effector;

When the matrix **J** is square, the singularities are obtained by the zeros of $\det(J)=0$, where det(J) indicates the determinant of the Jacobian matrix of the robot. They correspond to the zeros of $\det(JJ^T)=0$ for redundant robots.

In order to study the dimension task space and the singularities for our two robots, we proceed as follows by calculating: the rank and the determinant of the Jacobian matrix for RV-2AJ and RV-SD, where when explain the inability of the robot to generate an arbitrary velocity in the task space $\det(J)=0$.

C. RV-2AJ robot arm

The rank of the Jacobian matrix ${}^5J_5$ is 5 which is equal to the number of joint, RV-2AJ is non-redundant and has just the number of joints required to provide M degrees of freedom to the end-effector. For the RV-2AJ arm $\det({}^5J_5)$ can not be calculated because J is not a square matrix, therefore we have to proceed the study of singularities by separating RV-2AJ arm in two parts, taking the first three joints, the new Jacobian ${}^3J_3$ should be calculated, ${}^3J_3$ is not a square matrix with (6×3), only three of the six rows of the Jacobian are linearly independent. Having three degrees of mobility only, it is worth considering the upper (3 × 3) block of the new Jacobian where the singularities should be studied by calculating the determinant of this new matrix taken from ${}^3J_3$ [107].

D. RV-2SD robot arm

The rank of the Jacobian matrix ${}^6J_6$ is 6 which is equal to the number of the joint, therefore RV-2SD also is a non-redundant robot. For the RV-2SD we verify $\det({}^3J_6)=0$ arm, this rank drops to five in the following three singular configurations:

$$\det({}^3J_6) = -\text{ C3 D3 RL4 S5 (S23 RL4 - C2 D3)=0}$$

$$\begin{cases} \text{C3} = 0 \\ \text{S23RL4-C2D3} = 0 \\ \text{S5} = 0 \end{cases}$$

We say that we have three singularities for RV-2SD robot arm concerning $\theta_2, \theta_3, \theta_5$.

4.2.2. Inverse kinematics for RV-2AJ and RV-2SD robot arms

The objective of the inverse kinematic model is to calculate, from a given configuration q, the articular speeds $\dot{q}$ which assure to the terminal reference operational speed $\dot{X}$ imposed.

a) Inverse kinematics for RV-2AJ robot arm

RV-2AJ arm presents a singular case from regular cases, where The Jacobian matrix is not a square matrix and its inverse can not be calculated, therefore The most widely proposed methods for solving the inverse kinematic problem near singularities involve the use of the pseudoinverse J^+ of the matrix J as presented in the following equation [88]:

$$\dot{\mathbf{q}} = \mathbf{J}^+\dot{\mathbf{X}}$$

The pseudo-inverse for The Jacobian matrix ${}^5J_5$ is calculated using the Greville method that proposed a sufficient algorithm, we created a program in MATLAB in order to facilitate the computation (see **Appendix A4**).

The final iteration when k=5, it executes the pseudo inverse of the Jacobian matrix where we get:

Jp{5}= $\mathbf{J}^+$

b) Inverse kinematics for RV-2SD robot

In the regular case with 6 degrees of freedom, the Jacobian matrix J is square of order 6 and its determinant is not zero. The most general method consists of calculating J^{-1}, the inverse matrix of J, which makes it possible to determine the joint speeds $\dot{q}$ by the following equation:

$$\dot{\mathbf{q}} = \mathbf{J}^{-1}\dot{\mathbf{X}}$$
$$\dot{\mathbf{X}} = \mathbf{J(q)}\dot{\mathbf{q}}$$

Where we write:

$$\text{J(q)} = \begin{pmatrix} A & 0 \\ B & C \end{pmatrix} \qquad \text{so:} \qquad \begin{pmatrix} \dot{X}_a \\ \dot{X}_b \end{pmatrix} = \begin{pmatrix} A & 0 \\ B & C \end{pmatrix}\begin{pmatrix} \dot{q}_a \\ \dot{q}_b \end{pmatrix}$$

we get:

$$\dot{\mathbf{q}}_a = A^{-1}\dot{\mathbf{X}}_a \qquad \textbf{(20)}$$

$$\dot{\mathbf{q}}_b = C^{-1}(\dot{\mathbf{X}}_b - B\dot{\mathbf{q}}_a) \qquad \textbf{(21)}$$

Consequently, the inverse of J reduces to the inverse of two matrices of smaller dimension. For a six degree-of-freedom robot with a spherical wrist, the general form of J is given by **eq 20** and **eq 21** where A and C are (3x3) matrices [106].

The inverse of A and C for RV-2SD arm are calculated as follow:

$$A^{-1} = \begin{pmatrix} 0 & 0 & V1 \\ 0 & V3 & 0 \\ -1/R_4 & V2V3/R_4 & 0 \end{pmatrix} \qquad \text{and} \qquad C^{-1} = \begin{pmatrix} V4 & 1 & -V5 \\ S4 & 0 & C4 \\ -C4/S5 & 0 & S4/S5 \end{pmatrix}$$

where :

$$\text{V 1} = \frac{1}{S23R_4 - C2D_3}$$

$$\text{V 2} = -R_4 + S3D_3$$

$$\text{V 3} = \frac{1}{C3D_3}$$

$$\text{V 4} = \text{C4 tan 5}$$

$$\text{V 5} = \text{S4 tan 5}$$

From **eq 20** and **eq 21** we get:

$$\dot{q}_a = \begin{pmatrix} \dot{q}_1 \\ \dot{q}_2 \\ \dot{q}_3 \end{pmatrix} = \begin{pmatrix} V1\dot{X}_3 \\ V2\dot{X}_3 \\ (-\dot{X}_1 + V2V3\dot{X}_2)/R_4 \end{pmatrix} \qquad (\dot{X}_a - B\dot{q}_a) = \begin{pmatrix} \dot{X}'_4 \\ \dot{X}'_5 \\ \dot{X}'_6 \end{pmatrix} = \begin{pmatrix} \dot{X}_4 - S23\dot{q}_1 \\ \dot{X}_5 - C23\dot{q}_1 \\ \dot{X}_6 - \dot{q}_2 - \dot{q}_3 \end{pmatrix}$$

$$\dot{q}_a = \begin{pmatrix} \dot{q}_4 \\ \dot{q}_5 \\ \dot{q}_6 \end{pmatrix} = C^{-1} \begin{pmatrix} \dot{X}'_4 \\ \dot{X}'_5 \\ \dot{X}'_6 \end{pmatrix} = \begin{pmatrix} V4\dot{X}'_4 + \dot{X}'_5 + V5\dot{X}'_6 \\ S4\dot{X}'_5 + C4\dot{X}'_6 \\ (-C4\dot{X}'_5 + S4\dot{X}'_6)/R_4 \end{pmatrix}$$

In order to simulate and control the motion of both robots in MATLAB software, the previous mathematical analysis should be done, the explanation of the methods used are taken from different references as highlighted in **Appendix 4** [99, 108]. I explained also the method to calculate direct and invers dynamic model [109–111], in my case I interested on the first two models (Geometry – Kinematics) models. Nowadays, CAD softwares as SolidWorks give us the capabilities to calculate the inertia matrix for different structures, also a powerful tool which allows us to import the whole structure to MATLAB [112, 113], where we can control it easily, this part will be presented in the following.

5. STRUCTURAL DESIGN OF MANIPULATOR ARM USING SIMULATION TOOLS

The motion planning of a manipulator's arm is always based on the degree of freedom characterized by the joints placed in each link, where the number of the degree of freedom limits the workspace and defines the redundancy of the robot. The control of manipulator robots can be studied in two directions, depending on the requirements needed to achieve it [43]: (1) task execution, where the process is based on pick and place, welding, and painting; (2) path planning execution, where the process is based on the trajectory of the end effector, depending on each joint connected to the link of the robot arm. Therefore the study of the control theory for an industrial robot requires always the knowledge of the geometric - the kinematic and the dynamic parameters and equations, where these parameters allow us to study the trajectory planning executed by the robot by computing the reference setpoints of position, velocity, acceleration [114] presented as a function of time to ensure the transition of the robot around different desired positions.

In the robotics control field, MATLAB [115] is one of the most powerful simulation tools to simulate and control the behavior of the robot model formed by the inertia matrices for each joint. Using MATLAB Simulink we can build a model based on blocks that contains the mathematical modeling of the robot calculated previously, based on the robot model block we can create an open loop system chain by setting reference inputs that can be: desired positions - desired velocity or even desired torque to the robot model block system and getting different outputs that allow us to visualize and understand how the model will behave regarding the input data.

5.1. Design of industrial robot arms by application of SolidWorks software

Nowadays different CAD software have the capability to perform these kind of computations, we can refer to SolidWorks [98] which is more than "Computer-Aided Design" software, it is a complete set of tools intended to design, simulate, communicate, manage data and even assess the impact of products on the environment. It deploys three types of files relating to three basic concepts: the part, the assembly, and drawing. These files are related. Any robot structure designed in SolidWorks should be studied in two phases: - external structure - internal structure.

5.1.1. Methodology to build up an external robot structure

As we mentioned before, industrial robot arms are usually composed of several rotational joints and links. Several parameters have to be taken into consideration which are described in the next sub-chapters.

A. Dimensions of morphology modules

The shape designed for each link and the placement of joint between two links define the dimension where the main purpose is creating a stable morphology that can resist and achieve different task in the industrial environment, dimensions define also the workspace of a robot arm. The performing of dynamic analysis is needed in this phase in order to set the length and the masses for each link.

B. Material selection for modules

After finalizing the body shape and dimensions needed, SolidWorks offers a large material choice to create a real structure with real masses. Most of robots made today are based on several materials such as metal for industrial and service robots or plastic for amateur and prototyping robots. The use of each robot in a specific environment puts us under certain conditions in order to choose the right building materials. SolidWorks has a suitable module for calculation of inertia matrix if the length and the masses of links are chosen precisely according to the reality.

5.1.2. Methodology to build up an internal robot structure

The creation of external structure can only approach to the reality in simulation analysis without adding motor engines and the transmission motion systems, therefore we can study the behavior of any robot controlling only its mechanical external structure. To study more deeply and bring the reality in 3D environment it is useful to realize the internal structure where the study of the following points is needed:

C. Types of applied motors

The dynamic analysis enables us to calculate the torque applied for a given joint to rotate the module correspond to it. Industrial robot arm servomotors are the most fitting due to their precision and the internal feedback that maintain the angular position [116]. Dimension of servomotor plays an important role to determine the structure where we have to take into account the good placement for each motor inside the body.

D. Types of applied transmissions

The motion transmission system is a complex mechanical module which is responsible to transmit movement from one part (entry point) to another part (exit point). The shaping of this module can change depending on the function performed by increasing or decreasing velocity or torque or direction of movement without modifying its nature (rotation/translation), unlike the movement transformation mechanisms that serve to transform the movement between two parts. There are two types of motion transmission:

- transmission of movement in translation,
- transmission of rotational movement.

The most well-known transmission systems that can be used in a manipulator arm are strain wave gearings and toothed belt pulleys.

5.2. The CAD model constructed for RV-2AJ and RV-2SD robots arm

Modeling a real robot arm in a 3D environment requires the setting of the given parameters, therefore we started to build the external CAD structure of a Mitsubishi RV-2AJ robot arm and an RV-2SD robot arm. RV-2AJ robot arm combines 5 rotational joints and RV-2SD robot arm combines 6 rotational joints. **Figure 33 - Figure 34** present the external structure of RV-2AJ arm and RV-2SD arm in SolidWorks environment and the reality.

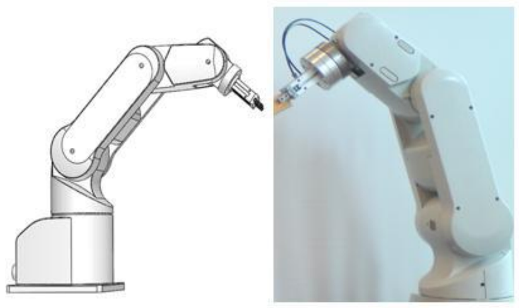


Figure 33. RV-2AJ robot arm in SolidWorks software and the reality

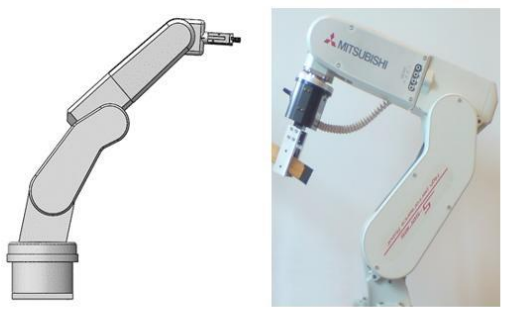


Figure 34. RV-2SD robot arm in SolidWorks software and the reality

5.2.1. Synchronization of SolidWorks software with MATLAB simulation software

SolidWorks is a powerful CAD tool where the real advantage can be shown in the ability to synchronize with other simulation software by creating two different files, these files regroup all the mechanical properties of the structure created in the assembly part of SolidWorks environment and can be exported and executed in other simulation software like MATLAB.

A. Building an URDF file in SolidWorks software

URDF file regroups all the parameters presented according to the assembly file of SolidWorks (Links – Joints – Type of Joints – Coordinate systems – Inertia matrix– Joints limits) of RV-2AJ arm and RV-2SD arm. It can be imported in Simscape. **Figure 35** highlights the functions used to create a URDF file for RV-2AJ arm as an example, where **Figure 36** demonstrates the location of URDF module in the assembly file of SolidWorks environment.

In the left side of **Figure 36**, we observe different axes regarding each revolute joint where before starting the exporting process of URDF file, we have to set the rotation axis for each joint, and set the robot in its home configuration.

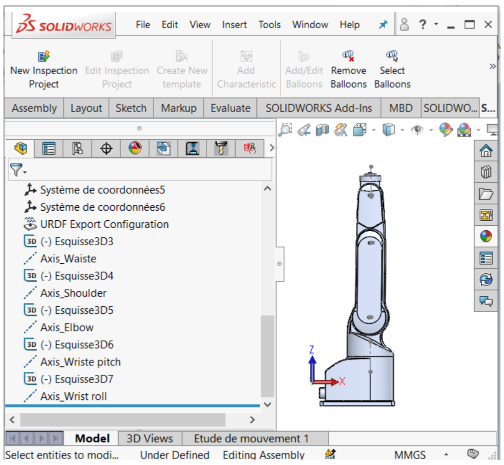


Figure 35. Generating URDF file in SolidWorks

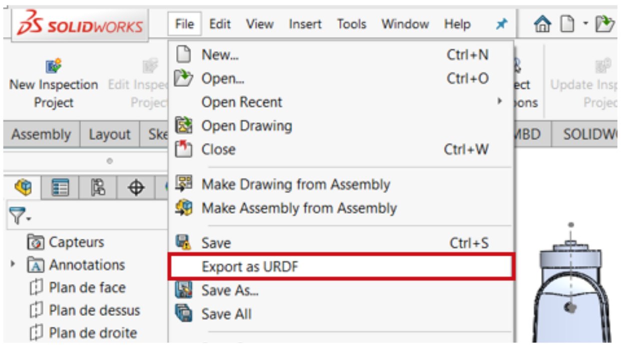


Figure 36. Exporting URDF file in SolidWorks

B. Building an XML file in SolidWorks software using *SimMechanics Module*

The module allows synchronization between SolidWorks and MATLAB where using MATLAB add-on to convert RV-2AJ assembly file to a Simscape model automatically in XML file, then all the modules and robot's parameters will be presented in MATLAB Simulink environment as shown in **Figure 37**.

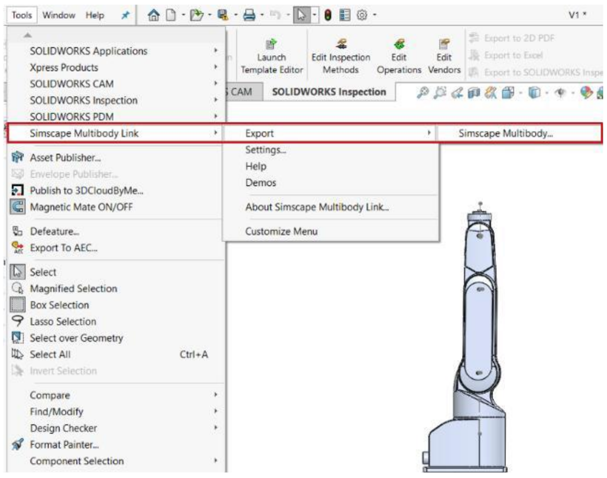


Figure 37. Exporting XML file in SolidWorks

These mentioned simulation tools and capabilities will be used for modelling our cooperating robot system in the following section.

6. “CARDS HOUSE” BUILDING IN VIRTUAL ENVIRONMENT

3D Simulation in the Smart Factory describes the modeling and visualization of real process functioning of system behavior over time in order to predict, evaluate and validate the performance of complex stochastic systems, processes, and production tasks. Thanks to the exponential growth of computing power, industrial simulation tools have been widely used in order to optimize the design and operation of the systems. Therefore, this section describes a scenario of building “house of cards” by RV-2AJ and RV-2SD robots in 3D robotic simulation tool. We can find many software in this area that have several powerful characteristics regardless the autonomy of scenario, adding equipment, control motions.

In our case we implemented the model of “card house” using CoppeliaSim software that is a 3D robot simulator based on a distributed control architecture [117]. This makes CoppeliaSim software very versatile and ideal for multi-robot applications, and allows users to model robotic systems similarly as in reality. Controllers can be written in C/C++, Python, Java, Lua, MATLAB, or Octave.

6.1. Design of the “card house” model in CoppeliaSim

The building model in CoppeliaSim software requires the import of all the pieces of equipment needed to achieve the task application, therefore, I modeled the following parts using SolidWorks software: RV-2AJ arm – RV-2SD arm – “cards” – table – mechanical holder – basement – “card” holder – gripper, the models of components should be imported from SolidWorks to CoppeliaSim as URDF files as seen previously or using STL file.

To realize a clean and flexible simulation, within the import process, CoppeliaSim is required to apply the mashing process to reduce the triangle number of components. **Figure 38** describes the pieces of equipment after the mashing process.

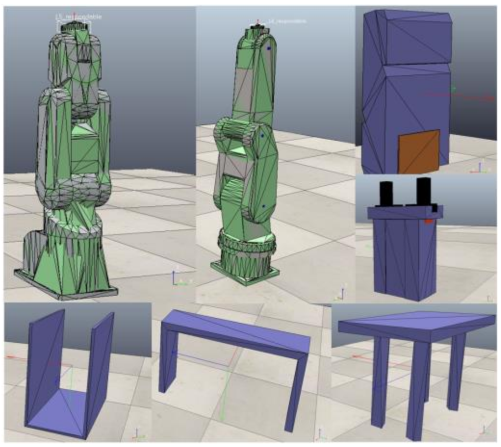


Figure 38. The pieces of equipment after the mashing process

6.1.1. Dynamic configuration of robots in CoppeliaSim

In CoppeliaSim the robots RV-2AJ and RV-2SD structures are presented as bodies with different links and without any actuations and configuration that allow them to move and execute motions, thus, the step after the mashing process is all about the configuration of the two robots according to the reality, by adding for both robots actuation in each joint, respecting the direction of rotation joint and its limit angles. **Figure 39** presents the configuration process of RV-2AJ arm in CoppeliaSim environment.

From the left side of **Figure 39**, we observe that the links and rotational joints that build RV-2AJ arm are cited in the scene hierarchy, where the scene hierarchy regroups all the objects presented in the scene (the environment).

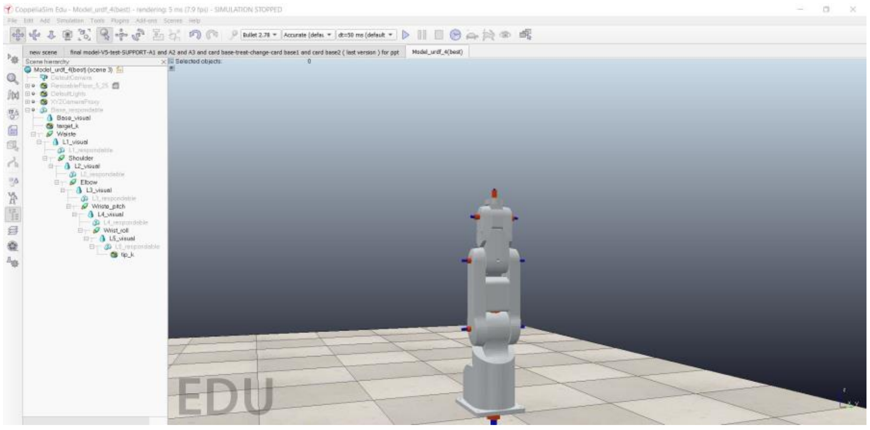


Figure 39. Photo of configuration + angles + hierarchy

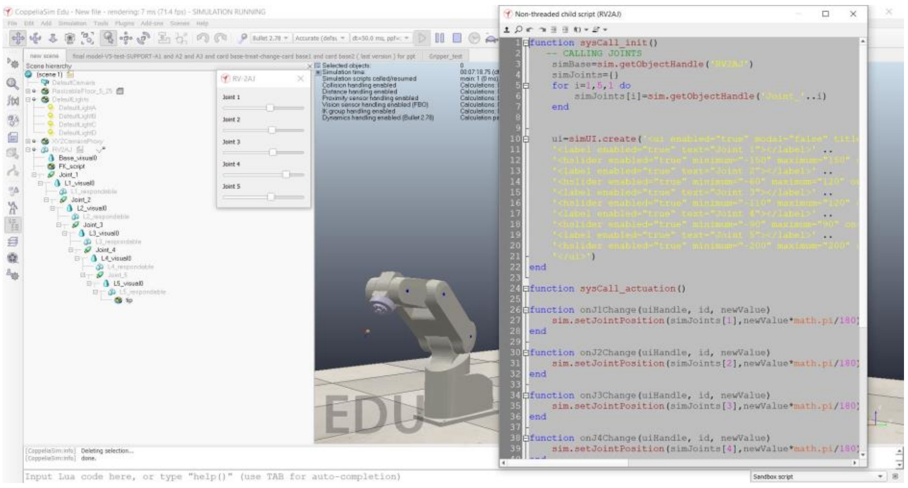
A. Forward kinematics of RV-2AJ arm in CoppeliaSim

Forward Kinematics (FK mode) for robots in CoppeliaSim is executed through a program written in a threaded or non-threaded script by using a specific command, as presented earlier that forward kinematics is the ability of the robot to execute a determined position by entering joints angle vector to the actuations taking into consideration the limits configured in the first step.

Figure 40 presents the code script written in Lua syntax language, where the kinematics functionality of the software is available through two different, independent methods:

- Via the kinematics plugin API functions [114]: this method allows setting up complex kinematics tasks via API functions exclusively. It is the recommended method since it allows to nicely isolate the kinematics functionality from the other aspects a simulation model might be involved with (e.g. dynamics, etc.).
- Via the built-in kinematics functionality via a graphical user interface (GUI).

The Coppelia Kinematic functionality is a collection of C++ functions that allow solving forward/inverse kinematics tasks for any type of mechanism (redundant/non-redundant, containing nested loops, etc.). Those functions give CoppeliaSim its kinematics calculation capability.



```
function sysCall_init()
    -- CALLING JOINTS
    simBase=sim.getObjectHandle('RV2AJ')
    simJoints={}
    for i=1,5,1 do
        simJoints[i]=sim.getObjectHandle('Joint_'..i)
    end


    ui=simUI.create('<ui enabled="true" modal="false" titl
    '<label enabled="true" text="Joint 1"></label>' ..
    '<hslider enabled="true" minimum="-150" maximum="150"
    '<label enabled="true" text="Joint 2"></label>' ..
    '<hslider enabled="true" minimum="-60" maximum="120" on
    '<label enabled="true" text="Joint 3"></label>' ..
    '<hslider enabled="true" minimum="-110" maximum="120"
    '<label enabled="true" text="Joint 4"></label>' ..
    '<hslider enabled="true" minimum="-90" maximum="90" on
    '<label enabled="true" text="Joint 5"></label>' ..
    '<hslider enabled="true" minimum="-200" maximum="200"
    '</ui>')
end

function sysCall_actuation()

function onJ1Change(uiHandle, id, newValue)
    sim.setJointPosition(simJoints[1],newValue*math.pi/180)
end

function onJ2Change(uiHandle, id, newValue)
    sim.setJointPosition(simJoints[2],newValue*math.pi/180)
end

function onJ3Change(uiHandle, id, newValue)
    sim.setJointPosition(simJoints[3],newValue*math.pi/180)
end

function onJ4Change(uiHandle, id, newValue)
    sim.setJointPosition(simJoints[4],newValue*math.pi/180)
```

Figure 40. Code script of Forward kinematics for RV-2AJ arm

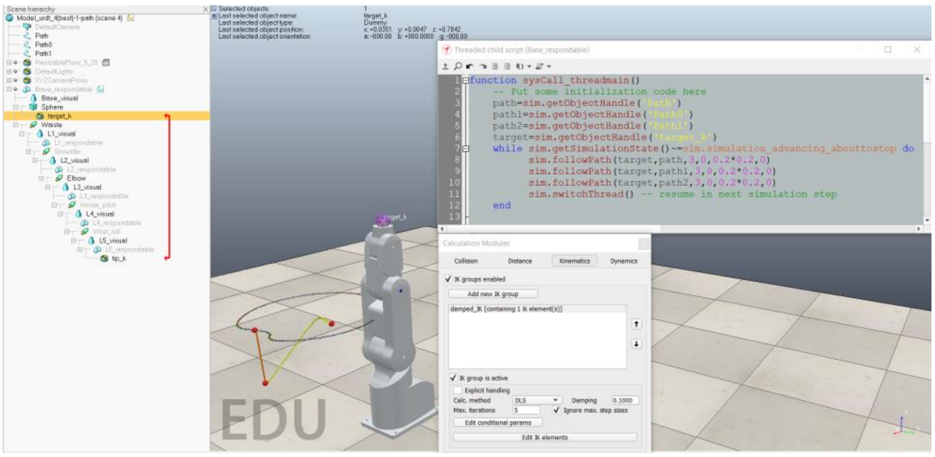
B. Inverse kinematic of RV-2AJ arm in CoppeliaSim

The software offers us inverse kinematics (IK mode) calculation module which is very powerful and flexible. It allows handling virtually any type of mechanism in IK mode. The problem of IK can be seen as the one of finding the joint values corresponding to some specific position and/or orientation of a given body element (generally the end effector). More generally, it is a transformation from the task space coordinates into the joint space coordinates. For a serial manipulator, for instance, the problem would be to find the value of all joints in the manipulator given the position (and/or orientation) of the end effector.

IK calculation module is based on two solvers (Pseudo Inverse – DLS "damping-least-square") [117], Pseudo inverse is useful for a redundant robot with a high degree of freedom on the other hand DLS based on using damping coefficient is good for a robot with 5 degrees of freedom or less.

The configuration of IK mode for RV-2AJ starts by creating at first an open kinematic chain between the base of the robot and its tip located at the end effector (the gripper). **Figure 41** presents the configuration process of IK mode for RV-2AJ arm.

Using code script or GUI available on the software, I created an IK environment that involves an IK group based on two objects (the base – the tip), then I choose DLS solver due to the degree number of RV-2AJ arm, setting some parameters as the coefficient of damping, number of calculation iteration, and selecting the position and orientation vectors [X, Y, Z, alpha, beta, gama]. This operation allows RV-2AJ arm to execute motions based on many positions as presented in **Figure 41**. where we create a circle and linear motions, by using code script we control RV-2AJ to move around these lines.



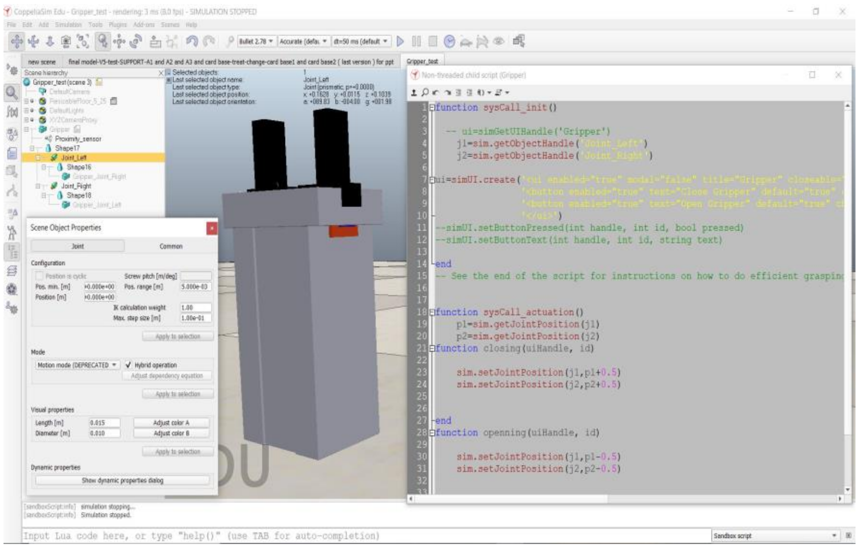
```
function sysCall_threadmain()
    -- Put some initialization code here
    path=sim.getObjectHandle('Path')
    path1=sim.getObjectHandle('Path0')
    path2=sim.getObjectHandle('Path1')
    target=sim.getObjectHandle('target_k')
    while sim.getSimulationState()~=sim.simulation_advancing_abouttostop do
        sim.followPath(target,path,3,0,0.2*0.2,0)
        sim.followPath(target,path1,3,0,0.2*0.2,0)
        sim.followPath(target,path2,3,0,0.2*0.2,0)
        sim.switchThread() -- resume in next simulation step
    end
```

Figure 41. Inverse kinematic configuration for RV-2AJ arm

C. The end effector of RV-2AJ arm in CoppeliaSim

RV-2AJ gripper in the reality is a pneumatic end-effector, the executed open-close motion is based on the pressure as presented earlier, in CoppeliaSim we set the motion of the virtual gripper by adding two prismatic joints resulting in a translation movement taking into consideration the distance of opening and closing for pick and place “cards”. **Figure 42** describes the configuration motion of RV-2AJ gripper.

From **Figure 42** we observe that the two prismatic joints in red color inside the gripper can handle the motion of the two fingers in black color by configuring the range of opening and closing distance, in the right side a code script is written to control the motion automatically by calling just the function responsible or manually by a user interface.

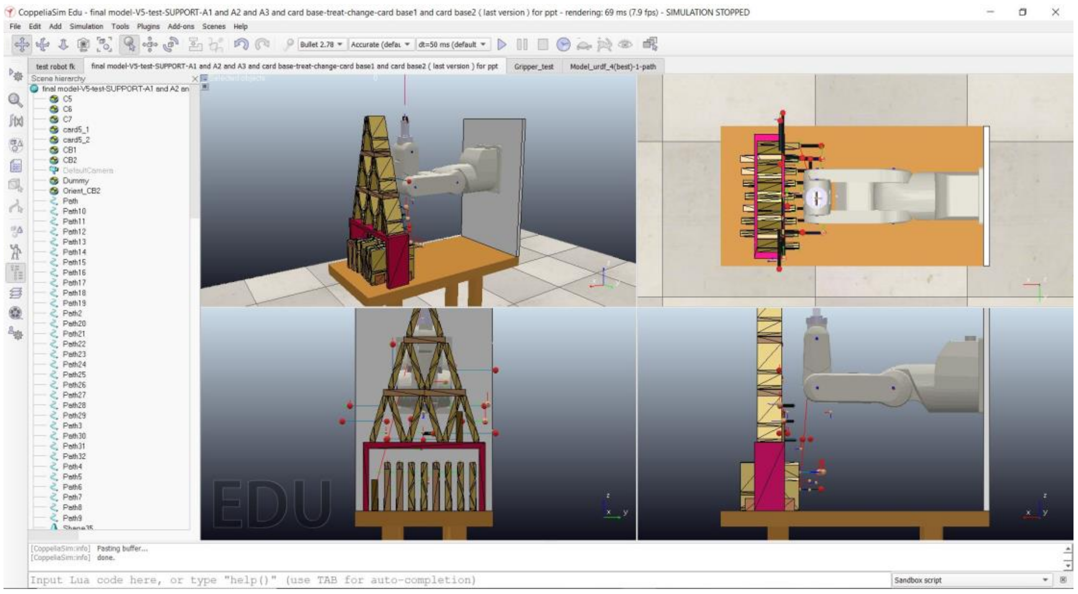


```
function sysCall_init()

  -- ui=simGetUIHandle('Gripper')
    j1=sim.getObjectHandle('Joint_Left')
    j2=sim.getObjectHandle('Joint_Right')

ui=simUI.create('<ui enabled="true" modal="false" title="Gripper" closeable=
                '<button enabled="true" text="Close Gripper" default="true" 
                '<button enabled="true" text="Open Gripper" default="true" 
                '</ui>')
--simUI.setButtonPressed(int handle, int id, bool pressed)
--simUI.setButtonText(int handle, int id, string text)

end
-- See the end of the script for instructions on how to do efficient graspin


function sysCall_actuation()
    p1=sim.getJointPosition(j1)
    p2=sim.getJointPosition(j2)
function closing(uiHandle, id)

    sim.setJointPosition(j1,p1+0.5)
    sim.setJointPosition(j2,p2+0.5)


end
function openning(uiHandle, id)

    sim.setJointPosition(j1,p1-0.5)
    sim.setJointPosition(j2,p2-0.5)
```

Figure 42. The configuration motion of RV-2AJ gripper

6.2. Building of “card house” in CoppeliaSim

At the final step, the assembly process has been realized after positioning all the objects and the robot as well as the real environment, **Figure 43** describes the final assembly model for “card” house building in CoppeliaSim. In this step, RV-2AJ can execute any motion and handle the pick and place task, the detection of “cards” to pick them successfully and precisely is achieved using an infrared sensor placed in the center of the gripper which test the detection and if the sensor detects an object than the gripper will be closed to pick the “card”, the robot moves to a specific position where it will place the “card” by the opening of the gripper.


CoppeliaSim Edu - final model-V5-test-SUPPORT-A1 and A2 and A3 and card base-treat-change-card base1 and card base2 (last version) for ppt - rendering: 69 ms (7.9 fps) - SIMULATION STOPPED

```
[CoppeliaSim:info]  Pasting buffer...
[CoppeliaSim:info]  done.
```

File Edit Add Simulation Tools Plugins Add-ons Scenes Help
Bullet 2.78
Accurate (defau
dt=50 ms (default
test robot fk
final model-V5-test-SUPPORT-A1 and A2 and A3 and card base-treat-change-card base1 and card base2 (last version) for ppt
Gripper_test
Model_urdf_4(best)-1-path
Scene hierarchy
EDU
Input Lua code here, or type "help()" (use TAB for auto-completion)
Sandbox script

Figure 43. The final assembly model for “card house” building in CoppeliaSim

Figure 43 shows the initial and the final states of the process, where RV-2AJ arm has different obstacles to avoid which make the execution task harder, especially that RV-2AJ has 5 degrees of freedom, therefore, the singularities are more supposed in a smaller workspace, the operation of pick the support “card” and place it in the aim to maintain the slanted “card” take a large time comparing to the cooperating of RV-2AJ and RV-2SD robots where the optimization of energy and time is guaranteed. In the abnormal scenario when only RV-2AJ arm builds the “card house”, the motion is executed by many positions using forward and inverse kinematics, especially when the robot place the “card” in the center area the singularities hamper the motion of RV-2AJ, thus we try to solve the problem by using forward kinematics instead of the inverse.

The next step is all about cooperating robots concept where RV-2AJ is installed in front of the process, together RV-2AJ and RV-2SD work in a parallel way. The scenario will be as following: RV-2AJ and RV-2SD start working together with picking and placing the “cards”, if some malfunction occurs in the process, the other robot will continue the work by using the support element which is an optimal solution for continuing the work. The detection problem and the reaction regarding it are executed using a tree search algorithm which is presented as one AI algorithm that will be described later. The program is written in Lua language in the threaded script at CoppeliaSim environment.

6.3. Cooperating robots in CoppeliaSim

In this section I intend to emerge intelligent thinking to the industrial robots in order to optimize the lead time of the process in case of errors, the methodology proposed was realized in the virtual environment where the control can be handled easily using sensors and camera, either than the reality where the controllers are black boxes, so their basic control theories cannot be modified easily. In case if we would like to apply the concept on real robots, only normal or abnormal scenarios can be realized otherwise the switching from the normal to the abnormal using critical thinking is not allowed due to the following reasons: - the unavailability of the required sensors in the robots and the lab that inform the second robot that there is a problem, - the inability to handle the robot's controllers to develop new technics.

If we would like to study more deeply the situation, we have first of all to create a graphical interface and a small controller using electronic equipment. The controller interface will receive the inputs from the robot and send the commands to it as outputs, the commands are developed and decided by the developed controller which is synchronized with a computer using MATLAB or Labview, this system is known as Master/Slave control where the original controller of the robots will be the slave and the computer with the developed controller are the master.

Figure 44 describes the cooperating process between RV-2AJ and RV-2SD robots in CoppeliaSim environment

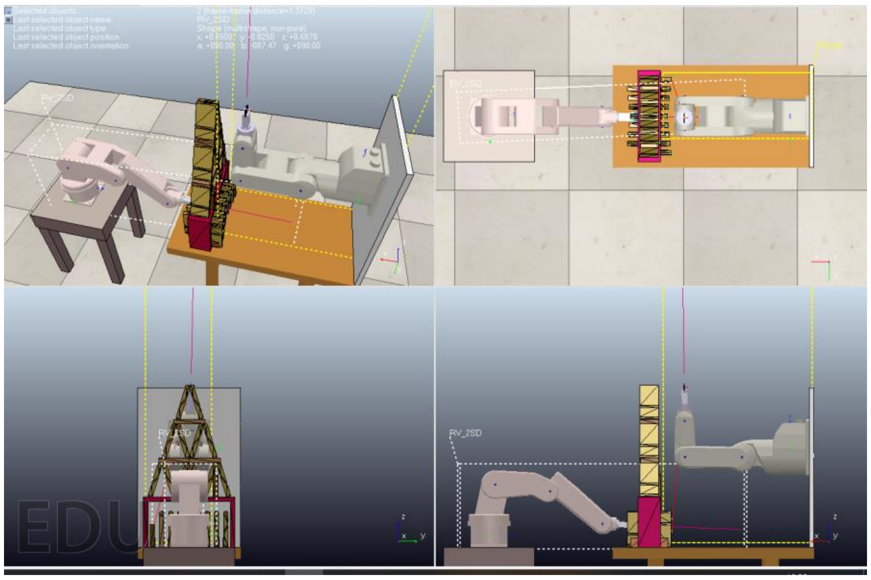


Figure 44. The cooperating process between RV-2AJ and RV-2SD robots in CoppeliaSim environment.

The simulation of cooperating robots in CoppeliaSim is advantageous as well as we can test different strategies of control as motion control- dynamic control. In the next section we intend to introduce the role of AI within the collaborating concept, by showing to the reader the most effective algorithms can be used for industrial robots.

7. MOTION CONTROL OF MANIPULATOR ARM USING SIMULATION TOOLS

7.1. Dynamic analysis and controlling tool in MATLAB software

The creation of an XML file and URDF file for RV-2AJ and RV-2SD arms allow us to generate a trajectory plan for the robot and define kinematic analysis by calculating the forward and inverse kinematics using MATLAB script and MATLAB Simulink modeling as presented in the next sub-chapters.

7.1.1. Simulation of robot motion with URDF file using MATLAB Script

Executing the script code presented in **Figure A.17** from **Appendix 5**, we got the following results presented in **Figure 45**. where RV-2AJ arm is located in its home position and in a random position.

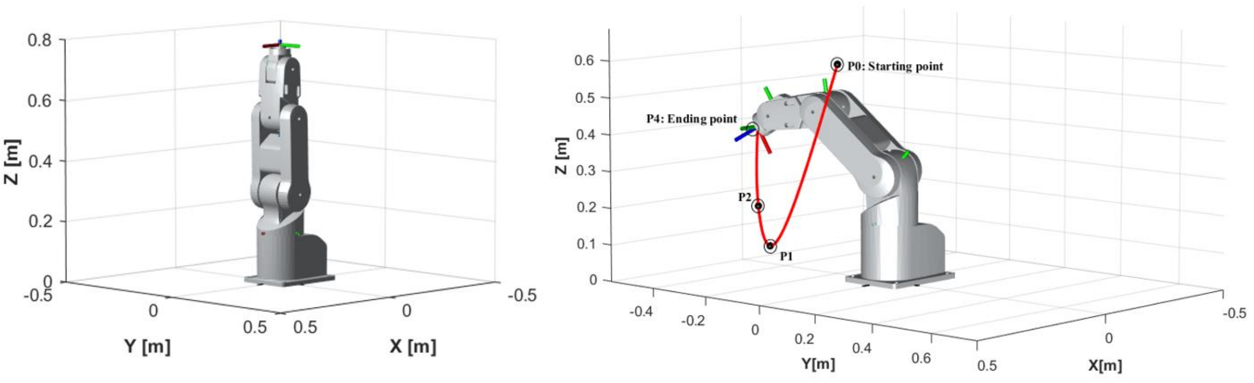


Figure 45. RV-2AJ arm is located in its home position and in a random position

The home position for RV-2AJ robot arm is when all the five joints are setting in 0°, otherwise the random position can be executed be provided setting the joints angles randomly.

In MATLAB Script, we can generate such code presented in in **Figure A.17** from **Appendix 5** to calculate forward and inverse kinematic for RV-2AJ robot exporting the model using URDF file. After setting the configuration of RV-2AJ arm and creating an inverse kinematics solver, I started by generating the desired trajectory using MATLAB script and visualize the results. The desired trajectory is represented in different waypoints including four points (P0, P1, P2, P3), using CSCVN function, we could assign a trajectory between them.

7.1.2. Simulation of robot motion using XML file in MATLAB software

As we mentioned before the loading of XML file in MATLAB generates some blocks that present the dynamics of RV-2AJ in MATLAB Simulink as shown in **Figure 46**. This file describes the links and joints of RV-2AJ robot, each component has its properties taken from SolidWorks.

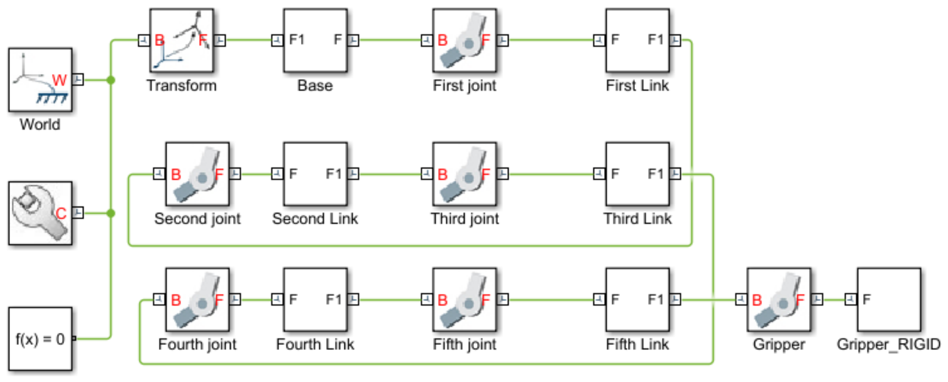


Figure 46. Internal blocks in Simulink of RV-2AJ arm

The advantage of simulation in Simulink that we can present our real robot by performing and adding new physical models, like electric components – PID controller – sensors to measure the position – torque – velocity – acceleration, also visualize the motion and define the workspace by detecting singularities of robots.

In our case, in the beginning I tried to visualize a simple trajectory by entering the desired trajectory as the input of the robot system and getting as output: the inverse kinematics represented in position and orientation vectors and the inverse dynamics represented in the torques measured in each joint. By assigning the input joint positions to a spline generator block using lookup table block, a spline trajectory is generated. This trajectory can be visualized in the Mechanical explorer of Simscape module attached to MATLAB software.

8. CYCLE TIME REDUCTION OF ROBOT ARM

The purpose of this study is to elaborate upon a new "Whip-lashing" method that aims to realize an improved trajectory for the five-degree-of-freedom RV-2AJ robot arm, i.e., to generate a path for a manipulator arm taking into account the design constraints (joint angle intervals – length of links – size of the workspace). This newly elaborated method seeks to minimize the cycle time of the trajectory with constrained torque values applied in joints for executing smooth motions. This new method originates from the motion of a whip analogue applied for RV-2AJ robot arm. First, I introduce the methodology of this newly elaborated method, identifying all the steps required using SolidWorks and MATLAB software applications. After, the main features of whip-lashing are introduced and **Section 8.2.** presents concrete simulations executed for both paths, to compare the normal motion of the arm to a whip lashing motion, based on cycle time. The final section presents the results of the joints torque changing according to the cycle time calculated in the previous section to show the reader the effect of the whip-lashing motion on the cycle time and the torque consumption.

The main value of the research is that a manipulator arm can be treated as a whip in certain conditions, which can guarantee running an improved path that results in reduced cycle time. The proof of this novelty is presented by applying the real parameters of an RV-2AJ arm to simulation tools.

8.1. Modelling the robot arm and the original and improved trajectories

The idea of trajectory improvement is taken from the motion of a whip. When applying a huge force at the handle, it will propagate along the whip's length, with a wave running alongside the whip transferring the energy from the handle to the tip. In order to realize this idea and check its real effect on the cycle time of robot arm motion, SolidWorks [98] and MATLAB [115] software applications were used in the dynamic motion simulations. For comparison, an "original" path with simple angle interpolation at the joints was also simulated. The two cases will be compared. For finding a better or "quasi-optimal" solution and proving the effectiveness of the newly elaborated whip-lashing method, two paths were generated with the same starting and ending points but with different internal motions. During the application of the method, both paths require the rotation of second–third–fourth joints, whereas the first and the fifth articulation are not used.

8.1.1. Original path

The original trajectory as a usual interpolated path is presented in **Figure 47** as a continuous red arc starting from start point S and ending in final point E. In this path, the RV-2AJ robot arm executes its motion by computing the angle steps for each internal point dividing the start-end angle of every joint with the number of points minus one step.

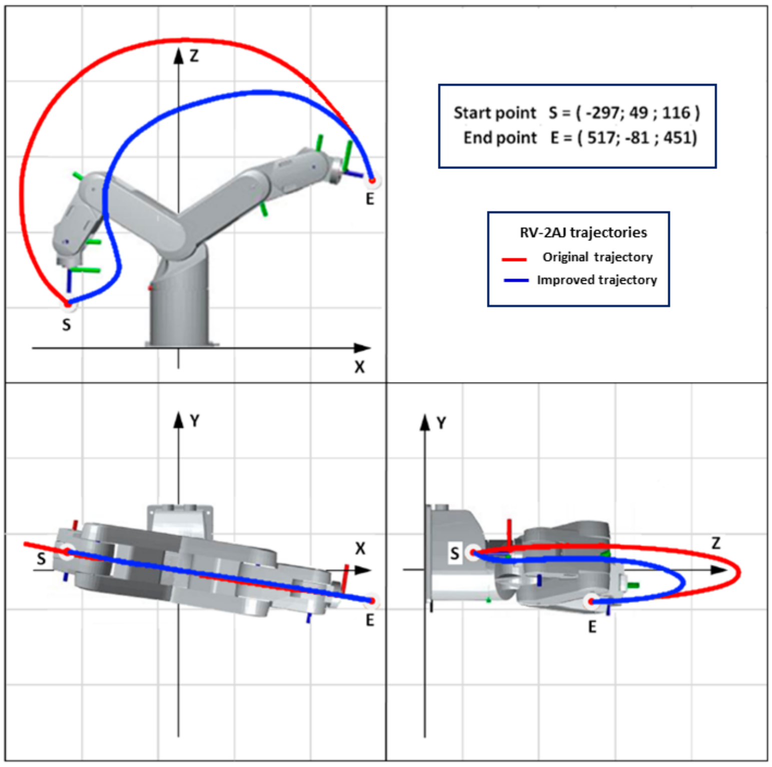


Figure 47. The trajectories of the RV-2AJ arm in three views

8.1.2. Improved path

The suggested trajectory named "improved path" given in blue in three views (front, top, and left side views) with the robot arm in **Figure 47** is based on the newly elaborated special method that imitates the natural motion of a whip. The improved path aims to decrease the cycle time for RV-2AJ arm's movement from the starting point S to the final point E. This method is based on principle of the whip-lashing motion that determines the torques applied in the closing and opening of joints of the arm, which will be discussed in the further sections.

8.2. Newly elaborated "Whip-lashing" method

For centuries whips have been considered instruments used to direct animals or torture slaves. Nowadays, this instrument has become an artform for some nations. In the last few years, researchers have been interested in the phenomenon executed by whips, which is characterized by the crack sound and the velocity that can reach supersonic speed [118, 119]. **Figure 48** presents the basic elements of a simple whip.

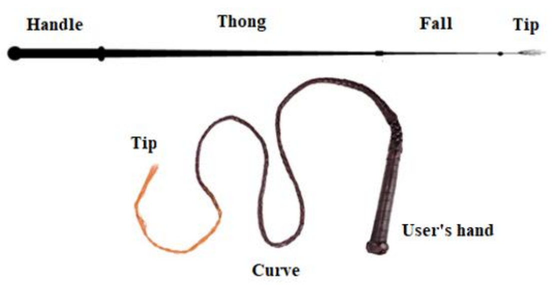


Figure 48. Basic elements of a whip

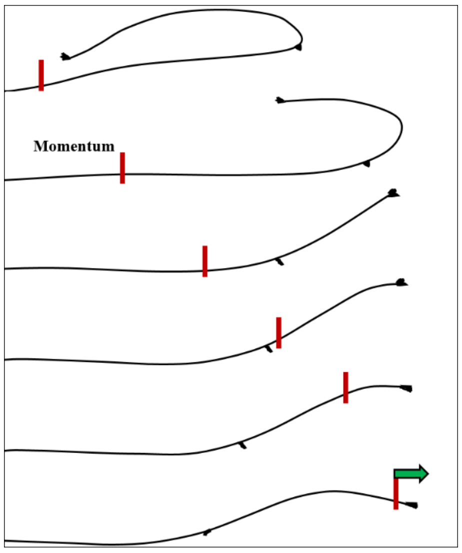


Figure 49. The motion of a whip

When we give a force for a whip with the "handle" to move up to down and then stop, that produces kinetic energy. This energy or more precisely the impulse will be transferred to the end of whip "tip" due to $p = m \times v$, where the mass (m) decreases and the velocity (v) increases.

The result of this is a sonic boom produced by a crack that is caused when some section of the whip moves faster than the speed of sound. The motion of a whip can include three types: a half-wave, a full wave and a loop. **Figure 49** and **Figure 50** present the transfer direction of momentum along with the segments of whip and the diagrams describe the change of velocity, mass, kinetic energy and torque as a function of time. The input values used in the diagrams in **Figure 50** were taken from Krehl et al. [120]. The subfigures show the following functions:

- The velocity diagram shows the changing of the velocity of the tip during a whip-lashing cycle.
- The mass diagram shows the changing of the mass of that part of the whip that has the most kinetic energy during the whip-lashing as the wave impulse goes along the whip length.
- The kinetic energy diagram corresponds to the work of the whip user who moves the whip and conveys motion energy to the whip with his hand.
- The torque diagram shows the torque value changing in the wrist joint during the whip-lashing.

As we mentioned earlier, the motion of a whip is based on the impulse conservation from the beginning of the whip to the arrival to the tip, where—before cracking—many parameters vary, such as the moving mass along the whip which will be decreased and produce high velocity, presenting the supersonic boom.

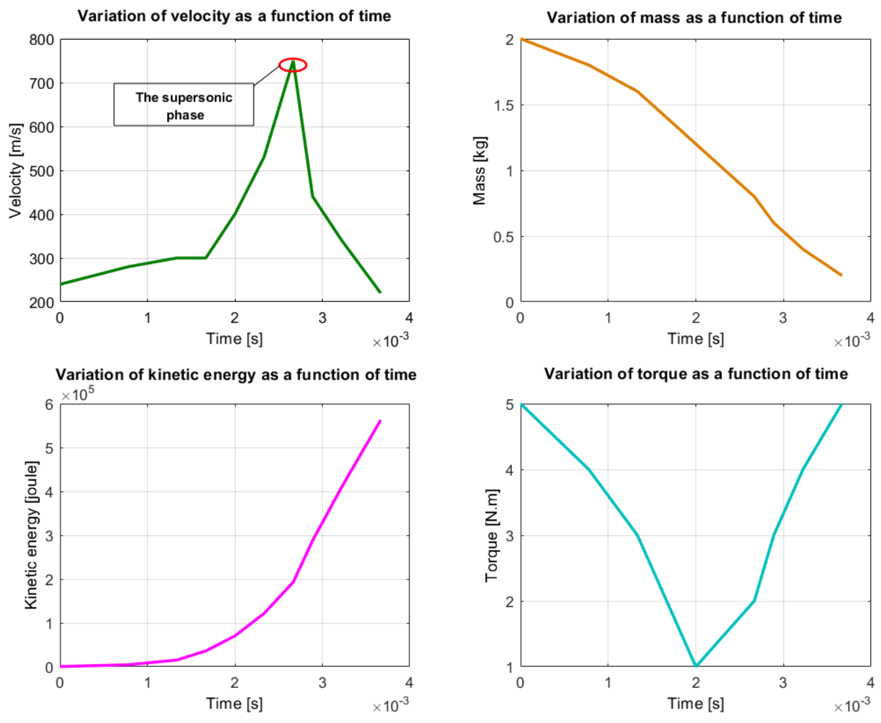


Figure 50. The variation of velocity, mass, kinetic energy, and torque as a function of time

This phenomenon results in diminution in the torque and an increase in kinetic energy, as **Figure 50** shows. A similar effect of conserving momentum can be seen in case of a figure skater doing a spin, when the skater closes his/her hands to his/her body, thus decreasing the rotational inertia and increasing the rotational velocity. Borrowing this characteristic of the motion from whip and considering the robot arm as a similar shape, the increase in the velocity of the robot parts will decrease the cycle time, so the motion of the robot arm can be used like in the mentioned whip or skater examples. The whip analogue fits the robot arm better because the arm is similar to a whip. It has many conjoined arm elements that become slimmer from the basement to the gripper.

8.2.1. Modelling of RV-2AJ arm motion as whip lashing in MATLAB Software

Based on the brief analysis on the motion of the whip and considering the structural similarity to the manipulator arm, the suggestion of a whip-analogous motion and the resulted in the trajectory of the gripper seems to be rational. In this analysis, I applied the same principle to the RV-2AJ arm, where the robot acted as a whip and the gripper executed the trajectory between two points faster than it otherwise would. The whip lashing motion of the robot arm can be followed through the sub-figures of **Figure 51**. When the second joint rotation stopped, the momentum was transferred to the third and fourth joints. As the moving mass became smaller, the rotational velocity increased at the last links of the robot arm.

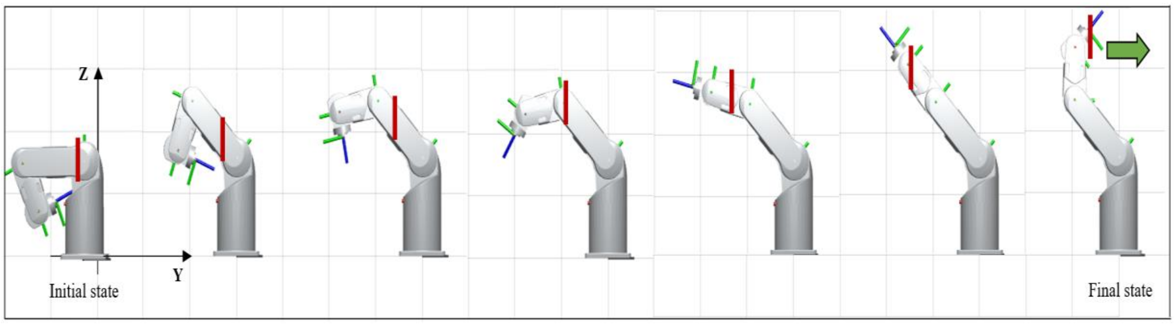


Figure 51. The motion of RV-2AJ arm

The first step of the operation of RV-2AJ arm started with the heavier segment, where the movement was realized by the torque applied at the second joint, at the "closer part to the base". In addition, the rotational inertia decreased when the motion of the links reached the smallest segment "end effector". As the motion proceeded the speedy "closer part to the base" decelerated and the outer joints opened. Finally, the "closer part to the base" stopped and the other joints finish the motion speeding up the motion of the "end effector".

In order to prove the efficiency of whip-lashing method, an iterative algorithm was created to execute the two trajectories (original and improved, see **Sections 8.1.1.** and **8.1.2.**) and calculated the minimal cycle time of each, trying to apply the maximum allowed torques for the joints.

The core of the algorithm inputted a given larger expected cycle time and determined the joint torque functions along the trajectory. If no torque maximum reached the allowed torque limit for any of the joints, the expected cycle time was decreased by a time step and the core process was repeated. If the expected cycle time was too small and the robot could only execute the trajectory with one or more joint torques exceeding the torque limit, then the process applied a backstep, halved the time step and continued the approximation of torque limit. At the end of this successive approximation, the minimum cycle time that utilized the torque limit—usually this happens for one joint only was obtained.

8.2.2. Elaboration of the Cycle Time Minimization (CTM) algorithm

By running the algorithm for the original and improved trajectories, the two minimum cycle times were determined and compared. In **Figure 52**, the elaborated cycle time minimization algorithm is introduced in detail.

The algorithm aims to calculate the cycle time *ct* of the trajectory. It was made using a successive approximation algorithm as mentioned before. The concrete parameters of the algorithm are the next. The number of trajectory points $TP = 12$, where the index of a specified point is $i = 1,\ldots, 12$. The number of RV-2AJ arm joints is 5, where the index of a specified joint is $j = 1,\ldots, 5$.

The number of trajectory points are determined when I created two paths for the robot by applying splines, in order to control these splines, I entered 12 points, where it was enough to give the form needed for both trajectories.

In the algorithm two conditions should be fulfilled:

- *ts* ≤ *ets* (*ts*—time step; *ets*—ending time step),
- **mT**[*j*] ≤ **aT**[*j*] (**mT**[*j*]—the joints' torque maximums; **aT**[*j*]—allowed torque for every *j*-th joint).

The algorithm aimed to determine the following data:

The **T**[*j*] torque vector is calculated for every *i*-th trajectory point and copied into the **TM**[*j*][*i*] torque matrix into the *i*-th column.

The **T** torque vector is determined by the "Inverse Dynamics MATLAB Robotics System Toolbox function" of the MATLAB Robotic Toolbox. Then, the maximum of every *j*-th row of **TM** torque matrix is determined to the *j*-th cell of the **mT**[*j*] maximum torque vector. Then the **mT**[*j*] ≤ **aT**[*j*] condition setting the *tof* torque overload flag selects between cycle time decreasing or time step refinement and back stepping.

The organigram presented in **Figure 52** contains different blocks (A–E) that define the following calculations:

- **A**: Filling up the **TM** torques matrix with the **T** torque vectors for every *i*-th trajectory point.
- **B**: Copying the *i*-th **T** torque vector in the *i*-th column of the **TM** torques matrix.
- **C**: Determining the **mT**[*j*] maximum torque for the *j*-th joint.
- **D**: Checking if any joint torque maximum **mT**[*j*] exceeds the allowed torque **aT**[*j*] for the *j*-th joint.
- **E**: Refinement of time step *ts* if necessary and continuing iteration, or finishing if time step *ts* goes below ending time step *ets*.

8.2.2.1. Technical measurements

Knowledge of allowable torque for robot articulation is required in order to check the condition in the algorithm for that I used a technical method to determine the torque limit for the (second-third-fourth) joints. The method is presented in **Figure 53**, it requires to measure the maximum force produced by the arm parts for each joint then calculate the allowable torque multiplying the force with the length of the arm segment, as can be followed in **Table 3**.

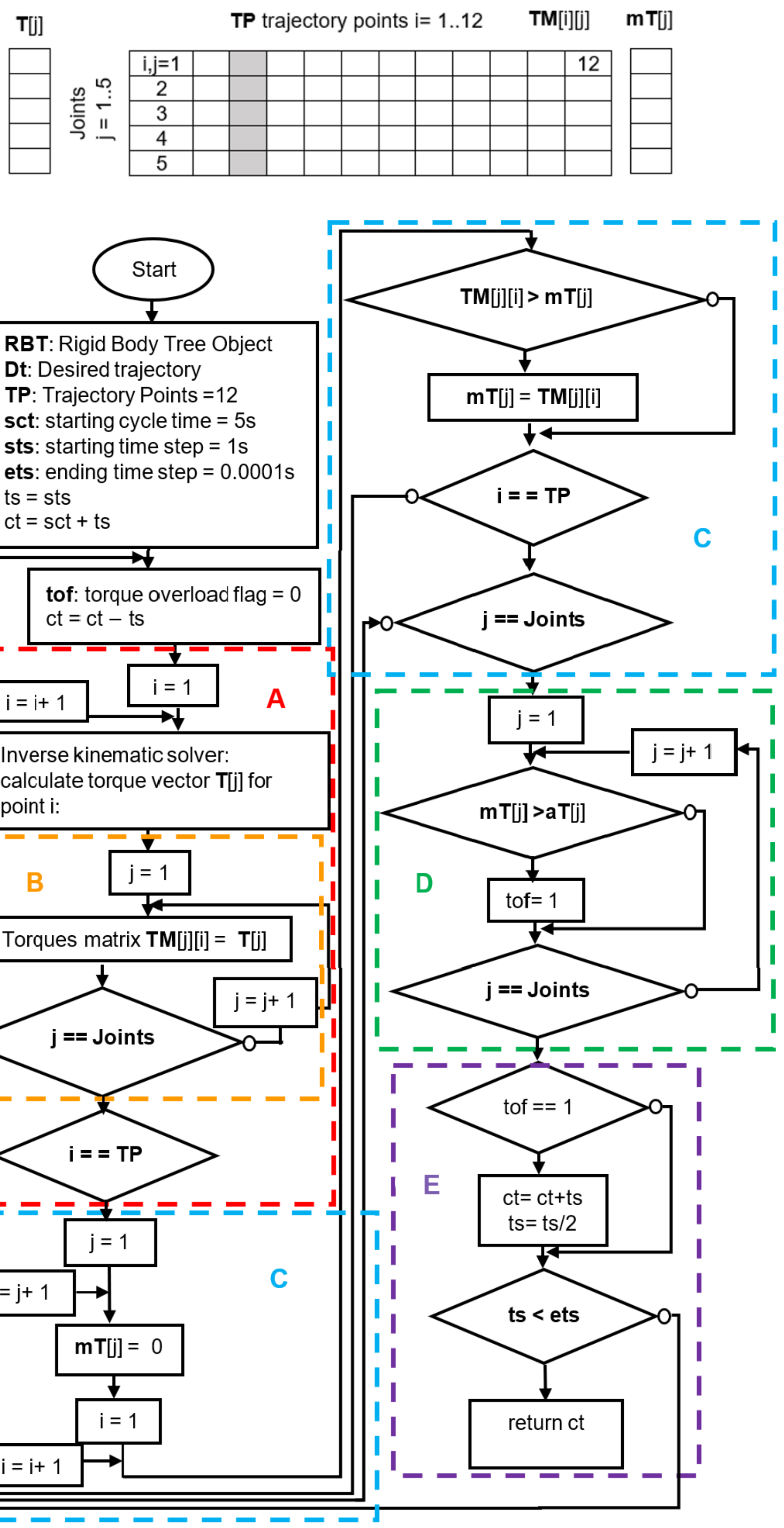


Figure 52. Cycle time minimization algorithm

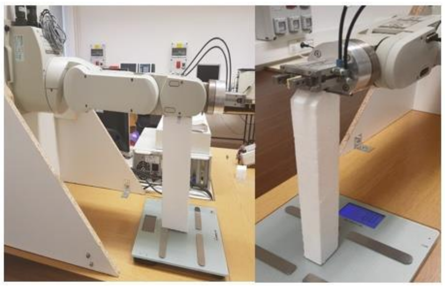


Figure 53. Technical measuring of allowable torques

Table 3. Measurements of allowable torques

Joints	2	3	4
Scale [dN]	15.8	6.7	4.5
Force [N]	154.94	65.70	44.12
Distance [m]	0.365	0.295	0.130
Allowable torque [Nm]	56.55	19.38	5.72

8.3. Trajectory optimization's results

The execution of script code, starting from a large cycle time *ct* = 5 [s], for both trajectories resulted in the possible best value of cycle time *ct* for each path. **Figure 54** presents the merging of both trajectories executed by the robot to compare their form. For the original path, the minimum cycle time was *ct* = 2.82 [s], while for the improved path, the minimum cycle time was *ct* = 1.86 [s]. Comparing the two results, we proved that the improved trajectory, which is described as a whip motion analogue, utilized the maximum allowed torques with shorter cycle time than the original trajectory.

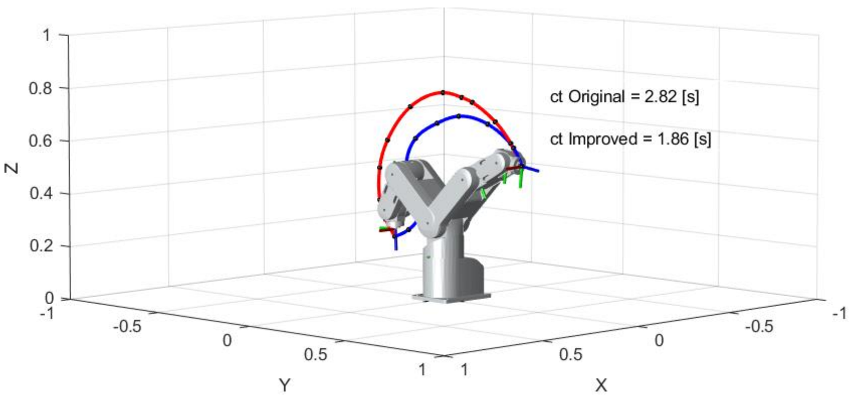


Figure 54. The original and the improved scenarios of RV-2AJ robot arm

After getting the good results for the two trajectories, I became interested in the behavior of the course of the successive approximations. After studying the cycle time variation, the analysis of torque change effect regarding the both trajectories are discussed in this section. To perform the simulation in the Simulink environment, I imported the RV-2AJ body structure XML file into MATLAB. Then, I configured each link and joint in the structure to receive the vector of positions as input blocks that represent the original and the improved path, as well as to calculate the torques applied for providing the series of such positions.

Figure 55 and **Figure 56** present the block system needed to determine the finishing cycle time *ct* and torque vectors **T** calculated during the *ct* minimizing process, where the blocks "Improved Trajectory" and "Original Trajectory" contain spline function for each joint specified by angles vector that should RV-2AJ arm execute according to each position. "RV-2AJ arm block" contains the mechanical properties of the robot arm. Using Scope Box, I visualized the torque diagram of each working joint (second–third–fourth, j = 2, 3, 4) for the original and improved paths.

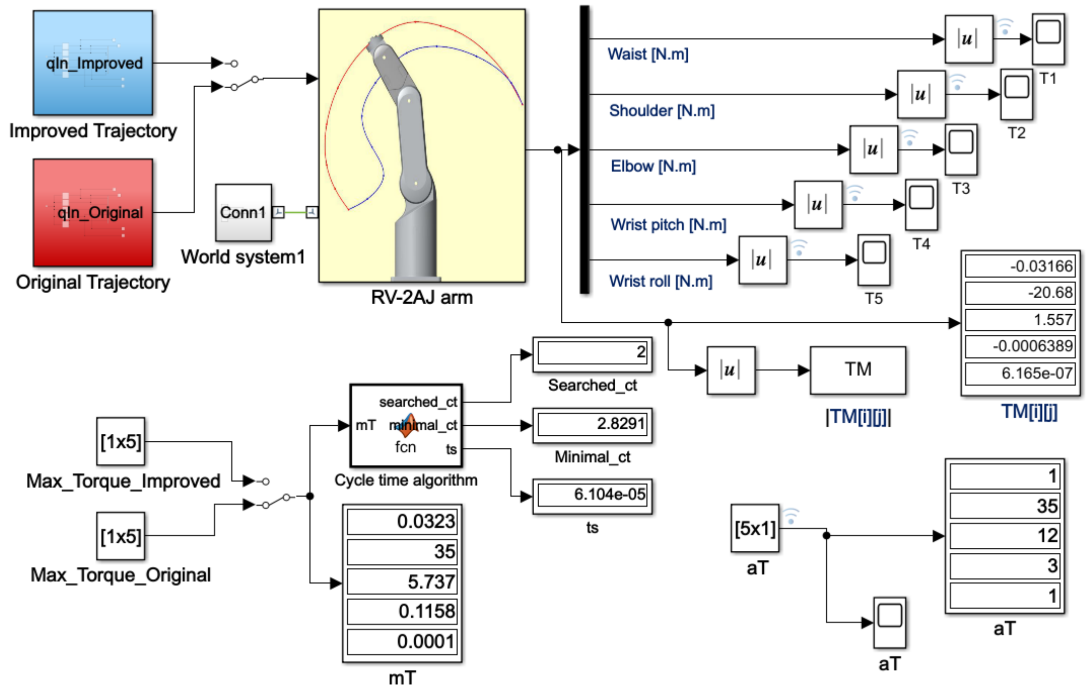


Figure 55. The block system scheme of RV-2AJ arm of the original trajectory

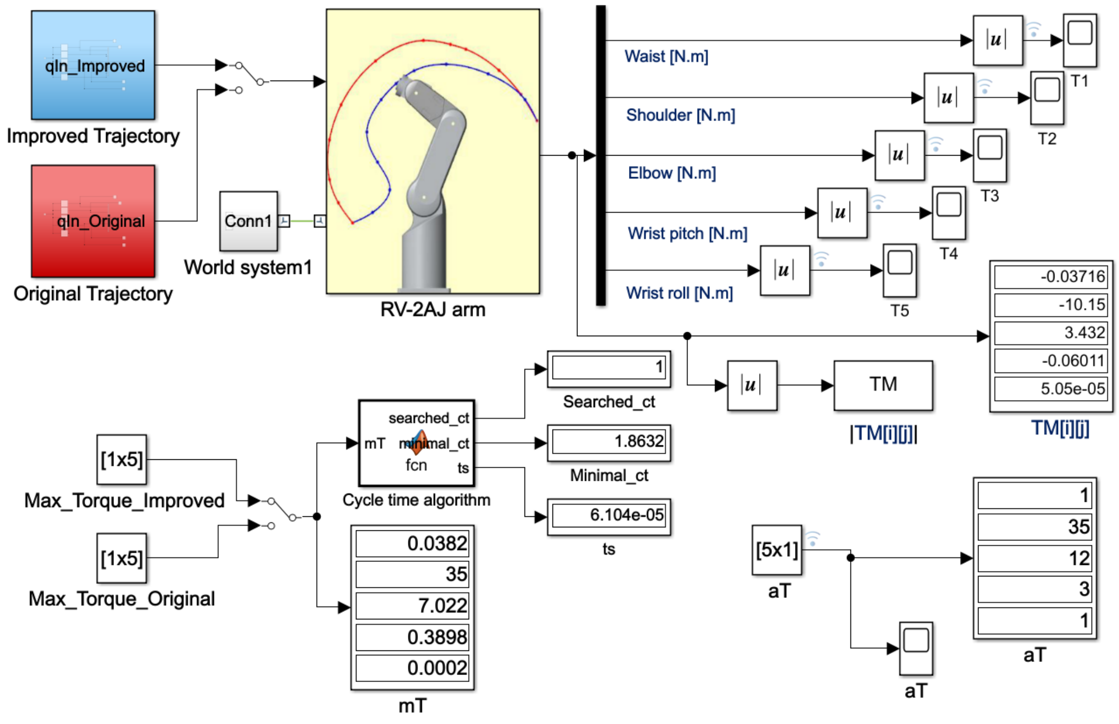


Figure 56. The block system scheme of RV-2AJ arm of the improved trajectory

I investigated especially the variation effect of the second joint (shoulder) and third joint (elbow) regarding the allowable torque values **aT** for those joints. As can be observed in **Figure 57**, the variation of torque values for the two paths for a starting cycle time *ct* = 5 [s] and *ts* = 1 [s] describes the behavior of the RV-2AJ arm.

In **Figure 57**, it can be seen that the torques of the fourth joint are close to zero due to the weight in this segment, which is very small compared to the other segments, which hold other segments. The comparison between the two paths was based on the shoulder, which presents the second joint. It was clear that in the first iteration where *ct* = 5 [s], the torque curve for the original path reached higher peak values than the improved path, where the maximum torque value for the original path was **mT** = 31 [N·m] and for the improved path was **mT** = 27 [N·m]. I also observed an overshoot in the beginning stage for both curves, which is explained as the gravity effect on the mechanical structure.

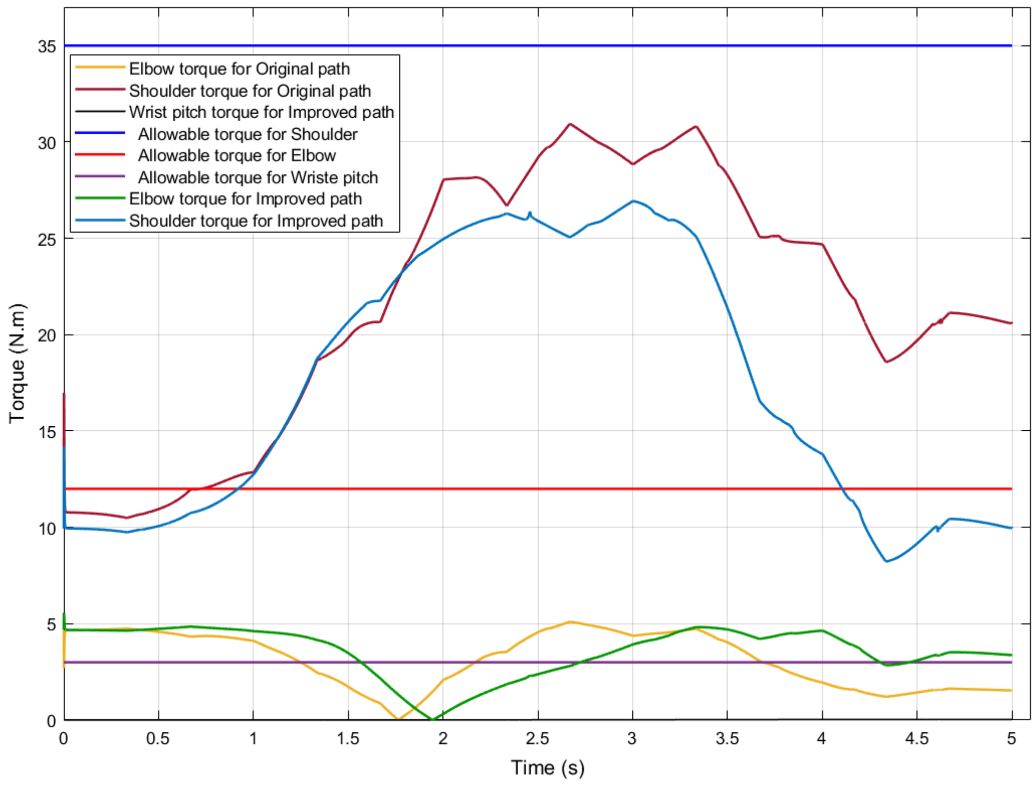


Figure 57. Variation of joint torques according to the two paths, ct = 5 [s]

Figure 58 presents the change of joints torques according to the original path for different cycle times *ct* = 5, 4, 3, 2 [s] to show the behavior of torque curves and compare their peak values in each iteration, in order to observe when the torque peak of an iteration exceeds the allowable torque value **aT**.

Regarding the second joint (shoulder), in the interval 0–5 [s] we observed that the decrease of cycle time resulted in an increase of torque value for the original trajectory, where in *ct* = 2 [s] the maximum torque value of the original trajectory exceeded the allowable torque value with **mT** = 42 [N·m].

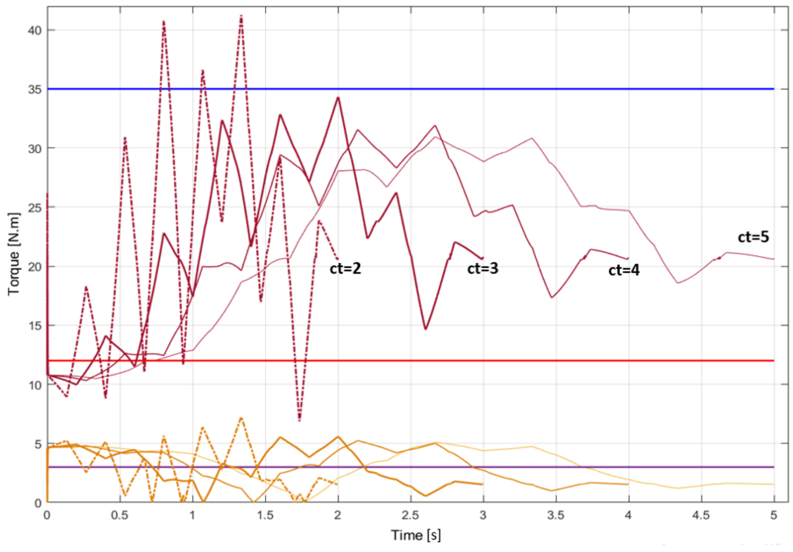


Figure 58. Variation of joint torques—original path—for different cycles

Therefore, this value *ct* = 2 [s] was stored as the searched cycle time value *sct* for the original trajectory and according to this value we calculated a new time step which was smaller than the first one, so the algorithm iterated the cycle time value more precisely.

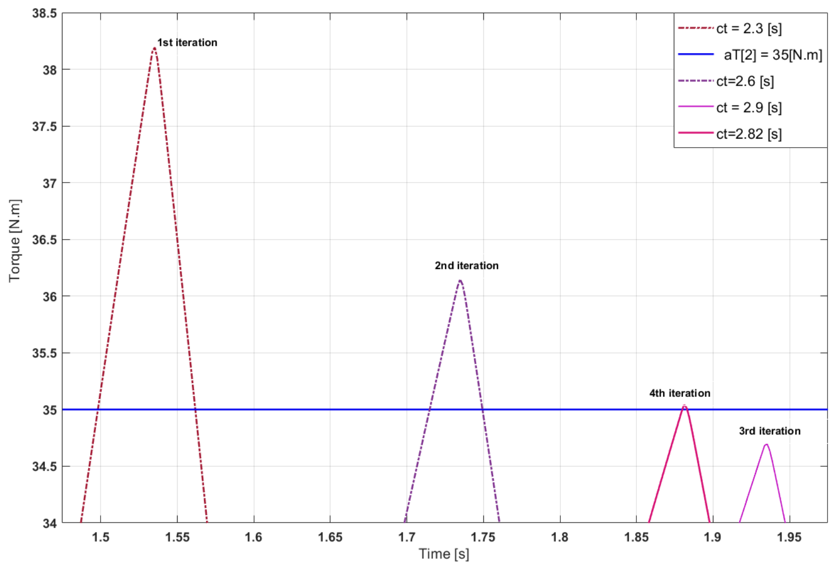


Figure 59. Different iterations of cycle time minimization algorithm for the original path around the searched cycle time sct = 2 [s]

As presented in **Figure 59**, the searched cycle time *sct* = 2 [s] obtained for the original path minimized the possible range to find the optimal maximum torque with optimal new cycle time. As a result, the optimal curve for shoulder was the red curve with *ct* = 2.82 [s]; this curve was obtained after the necessary iterations, starting from the first one, then minimizing the torque till obtaining the optimal curve, the "fourth iteration" and because we had a condition to stop iteration when the time step *ts* was smaller than the ending time step *ets*; therefore, the algorithm completed the execution and found a better solution with *ct* = 2.82 [s],

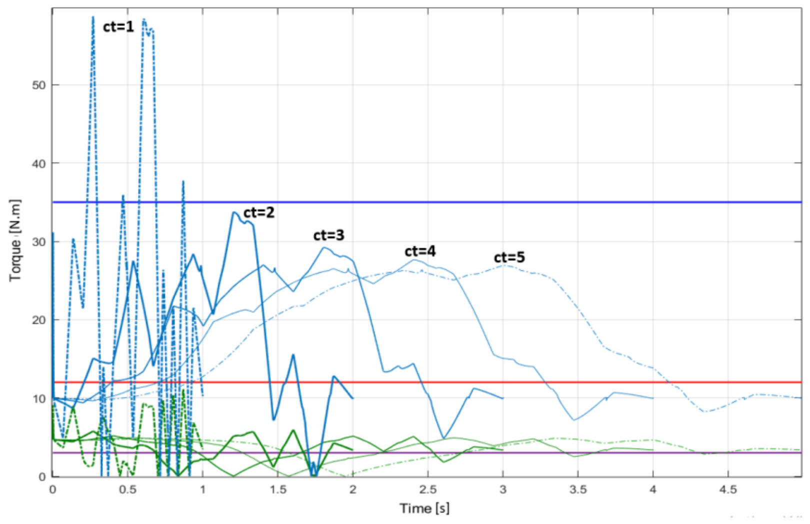


Figure 60. Variation of joints torques according to Improved path for different cycle times ct = 5, 4, 3, 2, 1 [s]

Figure 60 presents the variation of joint torques for the improved path for different cycle times *ct* = 5, 4, 3, 2, 1 [s] by executing different iterations of cycle times *ct*, in order to observe when the torque peak of an iteration exceeded the allowable torque value **aT**. For the improved path at *ct* = 1 [s], the maximum torque **mT** = 58 [N·m] exceeded the allowable torque value **aT**. Therefore, the searched cycle time for the improved path was *sct* = 1 [s].

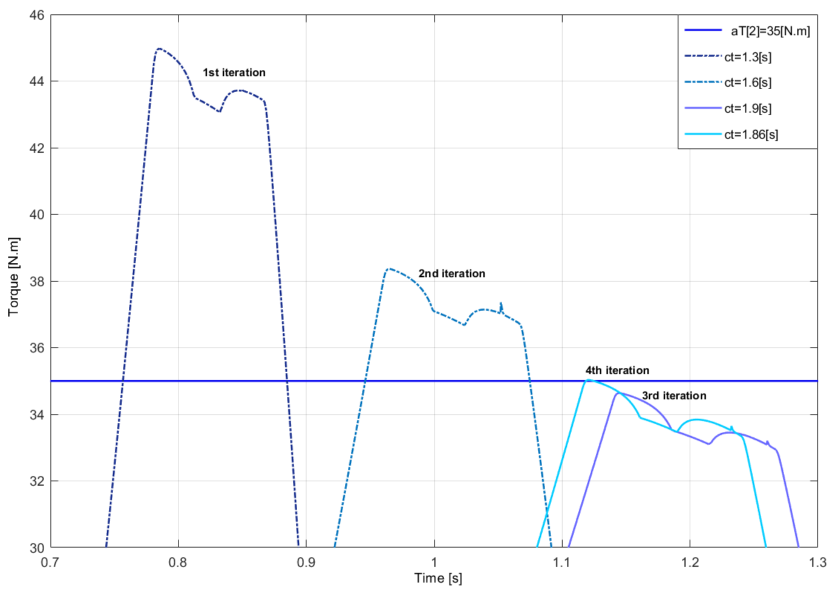


Figure 61. Different iterations of the cycle time minimization algorithm for improved path around the searched cycle time sct = 1 [s]

Figure 61 shows the execution of different iterations around the searched cycle time value obtained *sct* = 1 [s] from the results of **Figure 61** to find the optimal curve. From the diagram, after 4 iterations around the searched cycle time sct = 1 [s], the algorithm achieved the optimal curve with *ct* = 1.86 [s].

Based on the demonstration of diagrams and the comparison between the multi graphs of each path for the second joint, it was clear that the original path exceeded the allowable torque **aT** optimally with *ct* = 2.82 [s], unlike the improved path, which exceeded optimally **aT** with *ct* = 1.86 [s], as presented in **Figure 62**. Consequently, we proved that the improved path consumed 33% less time than the original path, which verified the concept of optimization.

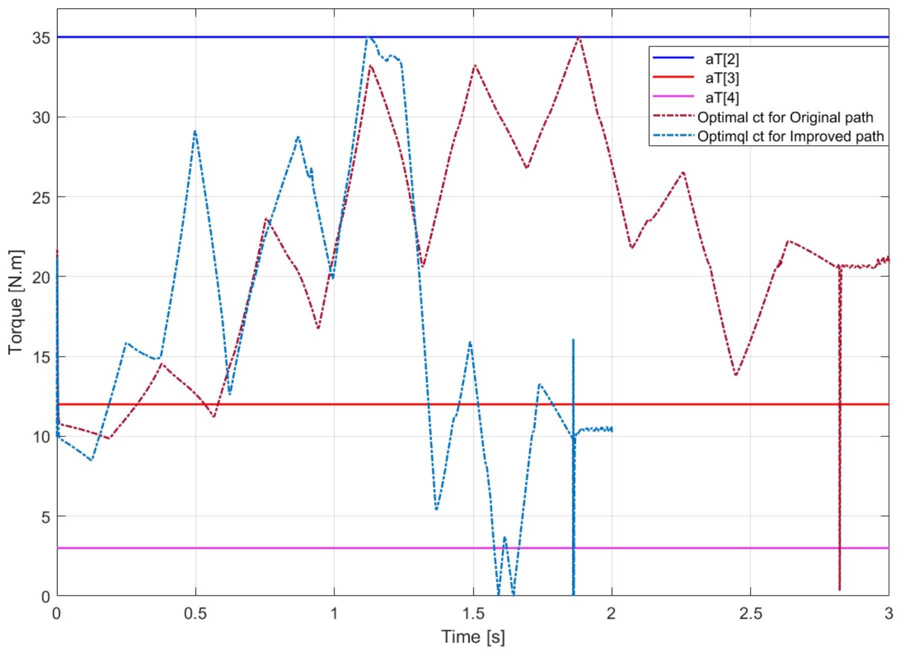


Figure 62. Optimal cycle time values for the original and improved paths

The application of this newly elaborated method can provide many advantages in industrial applications in the immediate future, where the robot arms will be designed as the shape of a whip; furthermore, the robot arms will be manufactured by usage of lightweight materials instead of recently used traditional metals. These innovative solutions (application of “whip-lashing” method and lightweight materials) will result in more flexible robot arms that can achieve motion with higher speeds without consuming higher energy, by the application of the momentum conservation law. This law can also be used by the existing rotation joint of robot arms where the motion can be achieved in a plain to increase the speed of the robot arm and decrease the motion time. This conception requires the application of new robot controllers and robot simulation software. It can be concluded that the application of the newly elaborated “whip-lashing” method results in achieving the optimized trajectory of the robot arms in order to increase velocity of the robot arm’s parts, thereby minimizing motion cycle times and to utilize the torque of the joints more effectively. Consequently, the productivity will be increased significantly. In the following research, the aim was to find a quasi-optimal trajectory between two given points. The searching for the quasi-optimum solution used the Tabu-search method.

9. NEWLY ELABORATED HYBRID ALGORITHM FOR OPTIMIZATION OF ROBOT ARMS' TRAJECTORY

The essence of the suggested methods and main characteristics are introduced in the followings.
It is a Tabu-search (TS) based optimisation method in 3D grid with interpolated point (wayPoint) insertion, plus grid step halving. The algorithm starts with a spline starting from the ***S*** endpoint and finishing in the ***E*** endpoint, fitted on a few inner interpolated points located on gridpoints of a 3D grid having larger grid step. Then a double embedded $c(c(TS-PI)-GR)$ cycle structure starts – the global function ***c*** means cycle with the cycle core between brackets that includes a $c(TS-PI)$ cycle where ***TS*** is Tabu-search and ***PI*** is Point Insertion, when a new interpolated point is inserted halving a selected segment of the spline.
The $c(TS-PI)$ cycle finishes when the last *PI* could not find better spline. In such case the last point insertion is cancelled. Then the ***GR*** Grid Refinement follows which halves the grid step. This halving finishes the core of the outer cycle. The outer cycle repeats its core, until the process stops. Several terminating condition can be imagined, e.g. the upper limit of the number of interpolated points.
The objective function for Hybrid optimization algorithm is the minimum cycle time of an iterated spline calculated using the elaborated cycle time algorithm presented in Whip-lashing theory.
The condition of the point has to fit the grid was a choice for us, and a hypothesis among many assumptions, in order to limit the working algorithm by by decreasing the possible points in the space of the task, and analyze it in simple manner, also the use of grid allow us to optimize precisely the trajectory by using the grid refinement process.

9.1. Tabu-search based optimisation method process

The process to optimize a defined trajectory modifying it properly is executed by using the following three processes:

9.1.1. Tabu-search in details

TS uses 6 neighbouring gridpoints $(x+gs,\ y,\ z)$, $(x-gs,\ y,\ z)$, $(x,\ y+gs,\ z)$, $(x,\ y-gs,\ z)$, $(x,\ y,\ z+gs)$, $(x,\ y,\ z-gs)$ of every inner interpolated point, where ***gs*** is the actual grid step. The search is executed in the space of non-tabu splines that are fitted on the 7^{N-2} variants of the neighbouring gridpoints and the $(N-2)$ inner gridpoints of the previous spline. ***TS*** cycle continues while better with a given amount of cycle time spline can be selected, so the difference between the actually found best local minima and the previous best local minima is smaller than the given terminating amount of cycle time.

I used Tabu-search instead of other algorithms as Genetic algorithm or Simulated annealing, because it has excellent performance, where the time needed for the task execution will be about 2 hours only. The following proposed Tabu-search method takes a smaller time than Genetic algorithm, where the number of population practically repeats the calculation task, so if we would like about 10 members in the population, then the calculation time would be 20 hours.

9.1.2. Point Insertion in details

The second part of the inner cycle core inserts a new interpolated point. The new point is generated as one of the halving points of the spline segments between spline interpolated points. Instead of such a halving spline point the closest gridpoint is taken. That such gridpoint is selected from the $(N-1)$ candidates which results in the best spline so the spline having the smallest cycle time.

9.1.3. Grid Refinement in details

The last step of the outer cycle core halves the ***gs*** grid step. As ***gs*** is one of the parameters of ***TS*** and ***PI***, the refined value will take effect in the continuation of the search process.

As the ***N*** number of interpolated points plays exponential effect in the algorithm based on the described search process, the stopping of the process using value of ***N*** is rational.

The goal of using Point Incrementation is to allow producing more fitting spline, but the use of only a few points in the beginning speeds up the search. The goal of using Grid Refinement is similar.

The algorithm tries to solve the task with the minimum interpolated points and with the largest grid step. The termination condition is ***N*** maximum value, because the actual ***N*** has exponential effect on running time.

The working of the $c(c(TS-PI)-GR)$ algorithm is demonstrated in 2D in **Figure 63** supposes that the optimal trajectory's spline is marked with wide red line. The starting spline marked 1 has ***N*=3** interpolated points, so 1 inner point. The blue splines numbered 2, 3, 4 shows the progress of the ***TS*** algorithm.

As there is no better spline fitting on neighbouring grid points, the ***PI*** point insertion follows. The 5-5 spline marked with brawn colour is the result of the ***PI*** point insertion. Then the ***TS*** works again and progresses through 6-6 and 7-7 brown splines.

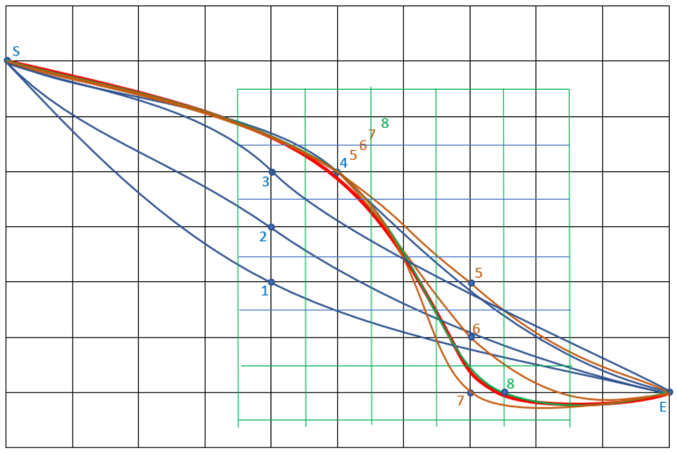


Figure 63. Tabu-search based optimisation method interpolation in 2D

Then terminates because there is no better spline with two inner points. The ***PI*** point insertion tries to insert a third inner point on the grid, but these trials does not result in better spline. So, the $c(TS-PI)$ cycle finishes and the ***GR*** grid refinement halves the grid step. Then the $c(TS-PI)$ cycle starts again and the ***TS*** works. Finds the 8-8 green spline and terminates because there is no better spline fitting on the neighbouring points. The ***PI*** point insertion tries to insert a point on the refined grid, but the terminating limit, $N < 5$ does not allow to increment N to 5.

Figure 64 presents Tabu-search flow chart for refining trajectory.

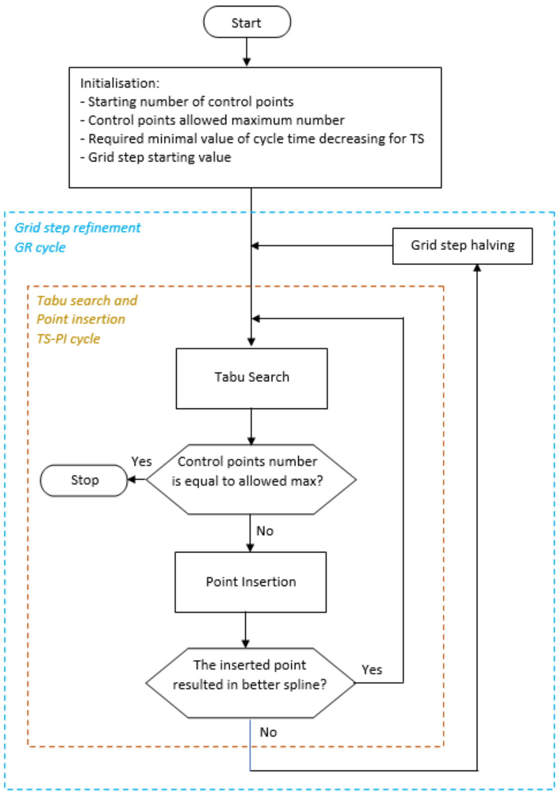


Figure 64. Tabu-search flow chart for refining trajectory

9.2. Parameters of the optimisation task

- The robot arm RV-2AJ or RV-2SD with its kinematic and dynamic properties
  - The robot arm working space V, so the limiting surfaces of those points that can be reached by the gripper of the robot arm
  - Vector of maximal allowed torques **aT** for every joints ***J*** of the robot arm
- The object **O** to be conveyed
  - The mass **m** of the **O** conveyed object
  - The vector **v** that points from the gripper coordinate-system origin to the mass center of the conveyed object
- Calculation parameters
  - Stopping criteria of the ***TS***
  - Stopping criteria of the points number incrementation cycle
- Starting parameters of
  - The searched trajectory at the initial moment
    - ✓ The number ***N*** of the trajectory interpolated points including ***S*** and ***E*** (***N***>2)
    - ✓ (X_i, Y_i, Z_i) coordinates of the trajectory interpolated points in a $P(X_i, Y_i, Z_i)$ matrix $(i = 1....N)$, where the coordinates of the starting point $S = P(X_1, Y_1, Z_1)$, ending point $E = P(X_N, Y_N, Z_N)$, and trajectory inner points $(i = 2....N-1)$.
  - Starting grid step gs

Figure 65 presents the PI point insertion cycle algorithm.

9.3. Case study of Hybrid algorithm for optimization of robot arms' trajectory

The algorithm execution was done in MATLAB software by entering the necessary input arguments for each block from the organigram and writing an optimal code script, **Figure A.18** from **Appendix 5** presents the inputs arguments of the Hybrid algorithm for optimization of the robot arm's trajectory. **Figure 66** describes the results of the hybrid algorithm execution for all iterations starting from the first trajectory till the final one, to observe clearly the trajectories obtained the result is presented in different views, where the first trajectory is presented in a purple color and the final trajectory in red color, the iterations between them are presented in blue color with the points which are presented in black color.

To see the iteration of the algorithm more precisely and study the nature of trajectories obtained, I plotted only the best iterations executed from **tabu_list** known as the best candidates during the iteration, starting from the initial trajectory till the final trajectory with their minimum cycle times as presented in **Figure 67**.

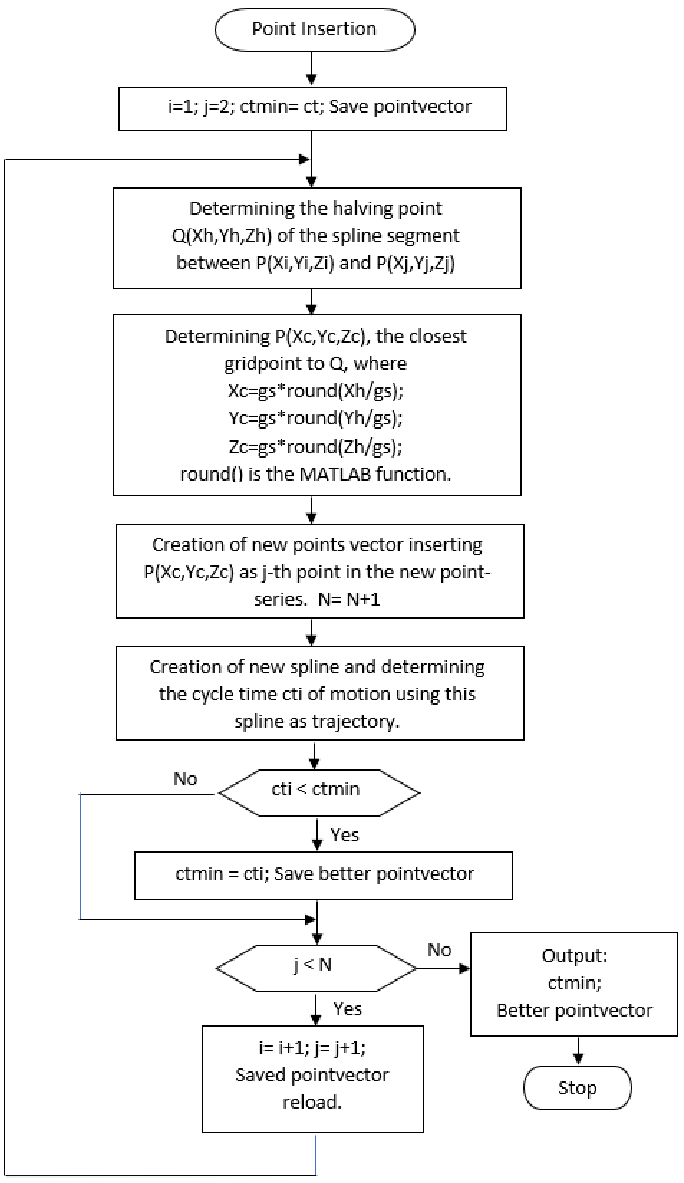

Point Insertion
i=1; j=2; ctmin= ct; Save pointvector
Determining the halving point Q(Xh,Yh,Zh) of the spline segment between P(Xi,Yi,Zi) and P(Xj,Yj,Zj)
Determining P(Xc,Yc,Zc), the closest gridpoint to Q, where Xc=gs*round(Xh/gs); Yc=gs*round(Yh/gs); Zc=gs*round(Zh/gs); round() is the MATLAB function.
Creation of new points vector inserting P(Xc,Yc,Zc) as j-th point in the new point-series. N= N+1
Creation of new spline and determining the cycle time cti of motion using this spline as trajectory.
cti < ctmin
No
Yes
ctmin = cti; Save better pointvector
j < N
No
Yes
Output: ctmin; Better pointvector
Stop
i= i+1; j= j+1; Saved pointvector reload.

Figure 65. Position insertion cycle flow chart

Figure 67 presents the best candidates trajectories executed by the hybrid algorithm, we can see that the algorithm works precisely, and it could converge and find the quasi optimal trajectory among many possibilities, beginning with a starting trajectory which includes 3 points (Starting point - First inner point – Ending point), this trajectory was initialized in the inputs arguments of the code script presented earlier. From this trajectory the algorithm starts to converge to the quasi optimal one using Tabu-search optimization based on evolving of the neighboring non-tabu splines of previous spline. A neighbour spline goes over one of the neighbour points of the first inner point creating a trajectory such a way. Neighbour splines are created first using the set of neighboring points then the algorithm calculates the cycle time for every of them. Comparing the values of cycle times for the neighbouring non-tabu candidate splines the local optimum spline for the next step is selected. The neighbouring represents the locality of the TS algorithm.

The calculation of a trajectory cycle time is made using the same algorithm that was used for Whip-lashing method, utilizing the limits of the joints torques, so every cycle time for a given spline is a minimal cycle time for that spline.

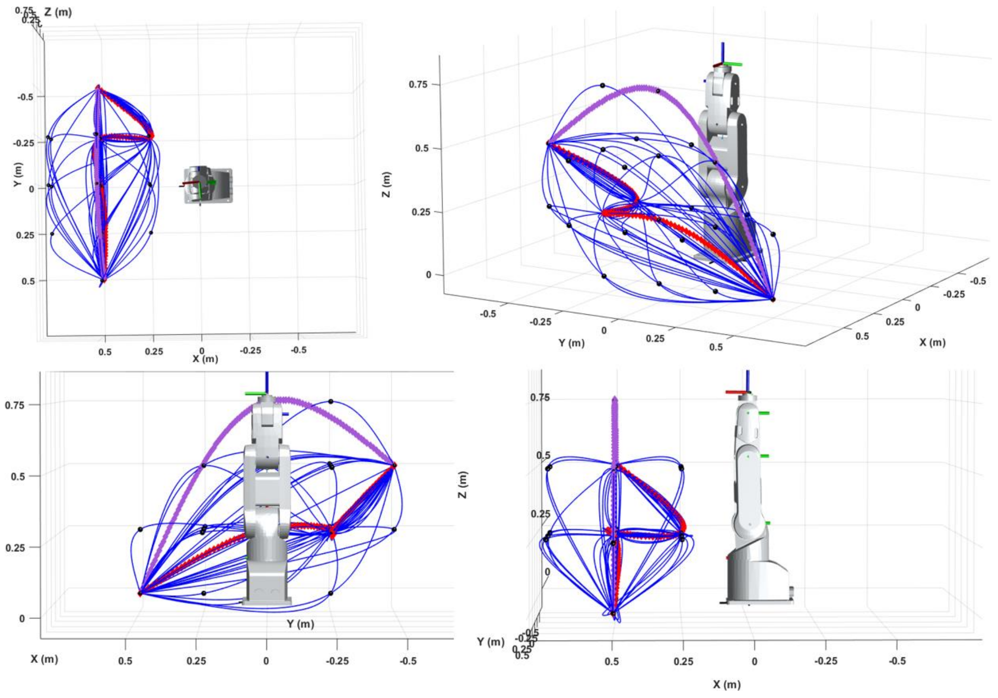


Figure 66. Hybrid algorithm execution for all iterations

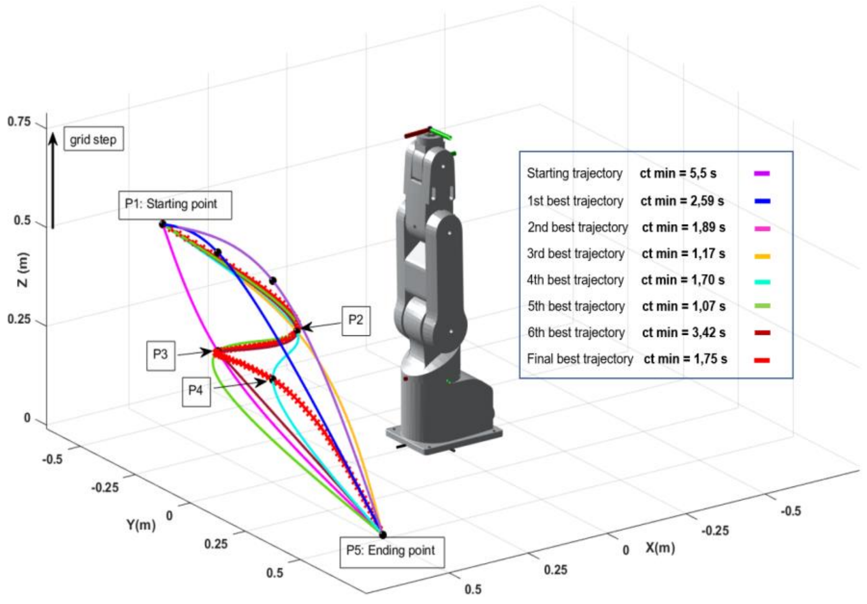


Figure 67. The best candidates trajectories executed by the hybrid algorithm

Tabu-search will choose the locally best and using it converges to the global best trajectory, as presented in **Figure 67**, where the algorithm goes from starting trajectory to 1st best trajectory, the minimum cycle time for this trajectory ***ct min*** = 2.59 [s] is smaller than the starting one which has ***ct min*** = 5.5 [s], so from the first insight we remark that the algorithm optimizes the cycle time with the great percentage. By completing the same process, the algorithm converges to the 2nd best trajectory – 3rd best trajectory with number points included in these trajectories N = 3, as we can see that the minimum cycle time in each iteration be more optimal like for 3rd best trajectory the value of the cycle time became ***ct min*** = 1.17 [s]. From 4th best trajectory, the algorithm finishing this Tabu-search period so finding the best trajectory that can be created with 3 points, jumps to the second method (Point insertion) by inserting the second inner point in the trajectory, and the process of Tabu-search start working again to find 5th best trajectory which include also 4 points and minimum cycle time ***ct min*** = 1.07 [s]. By iterating another time the algorithm executes 6th trajectory which results in inacceptable cycle time comparing to previous values, therefore the second method is applied by inserting the third inner point in the trajectory. The algorithm executes the final best trajectory with ***ct min*** = 1.75 [s] and stopped here due to the condition of the number point limit N=5. As a remark the algorithm did not go to the third method which is grid refinement now, because we reached the limit of N and among these iterations, the algorithm does not find any difficulty to insert a good position that satisfies our conditions. As a conclusion from this process, we can confirm that the hybrid algorithm can find quasi optimal trajectory starting with a random one by converging to the best trajectory with minimum cycle time value.

9.3.1. Case study of joints torque effect regarding starting and ending paths

In this section, I wanted to check the torque effect behavior on the best iterations, therefore I created a Simulink model presented in **Figure 68** that contains the following blocks:

- **Trajectory** block where inside this block we have 5 spline function for 5 joints, inside the spline function we find the parameters of the inverse kinematic solution, and the cycle time calculated using cycle algorithm.
- **RV-2AJ arm** block includes the mechanical structure of RV-2AJ robot arm with its given configuration.
- **Cost function** block which is an essential parameter to calculate the torque function, this parameter is used if Tabu-search finds two or more trajectories with the same minim cycle time, so it will choose the best one by comparing them regarding the minimum torque cost value.

Executing the file of Simulink, RV-2AJ will iterate according to the optimal iterations in the mechanics explorers window, as presented in **Figure 69**.

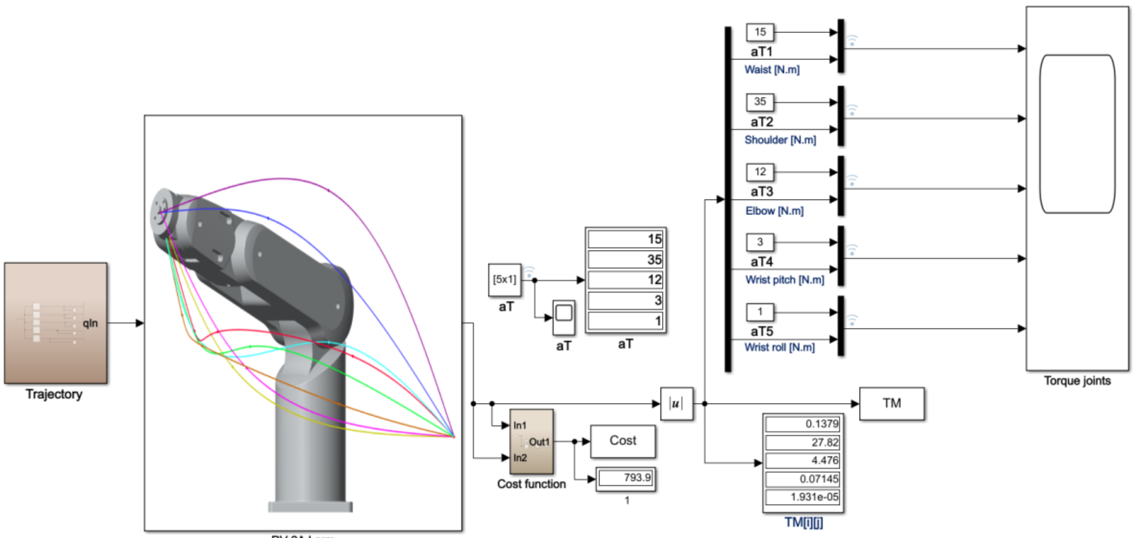


Figure 68. The block system scheme of hybrid algorithm

Using scoop box we can study the torque effect for (the first – second – third – fourth) joints where the fifth joint does not affect, as we can see in the diagrams presented in **Figure 70** the torque effect for the starting and final paths with *ct* = 6 [s] did not accede the allowable torque values, so the cycle time algorithm starts to decrease the time step.

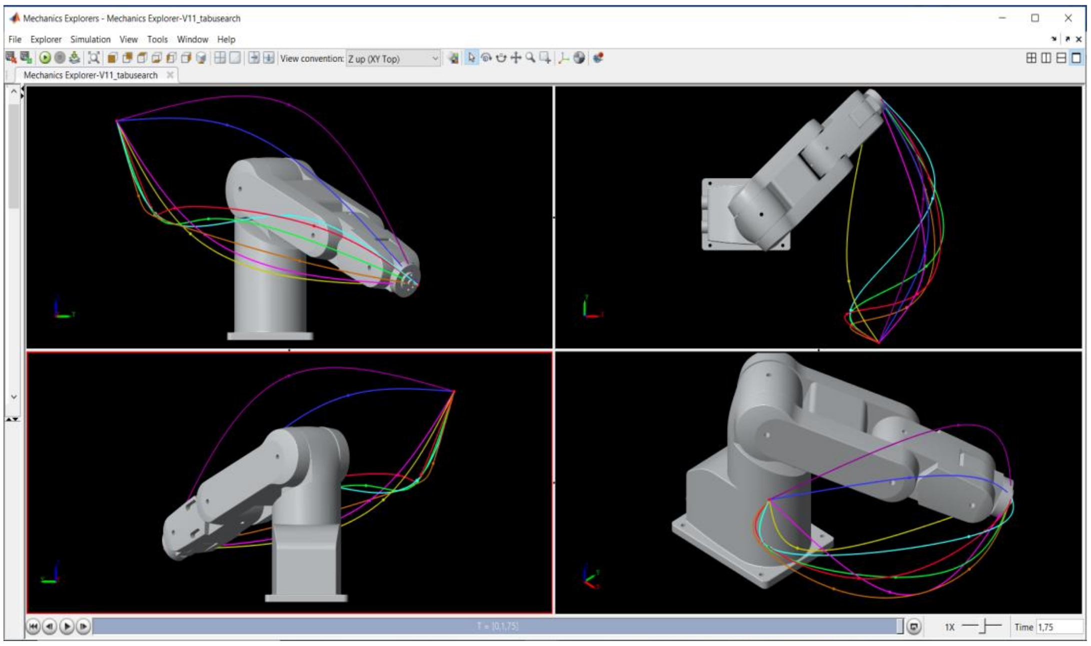

Mechanics Explorers - Mechanics Explorer-V11_tabusearch
File Explorer Simulation View Tools Window Help
View convention: Z up (XY Top)
Mechanics Explorer-V11_tabusearch
1X
Time 1.75

Figure 69. Hybrid algorithm results for best trajectories in the mechanics explorer from different views

Figure 71 Presents the process of searching the minimal cycle time for both trajectories, where we can see that with the decreasing of the cycle time the torque of the first joint for the starting path exceeds the aT_1= 15 [N·m] when ct= 5.5 [s], where for the last path the torque of the second joint exceeds aT_2 = 35 [N·m] when ct= 1.75 [s].

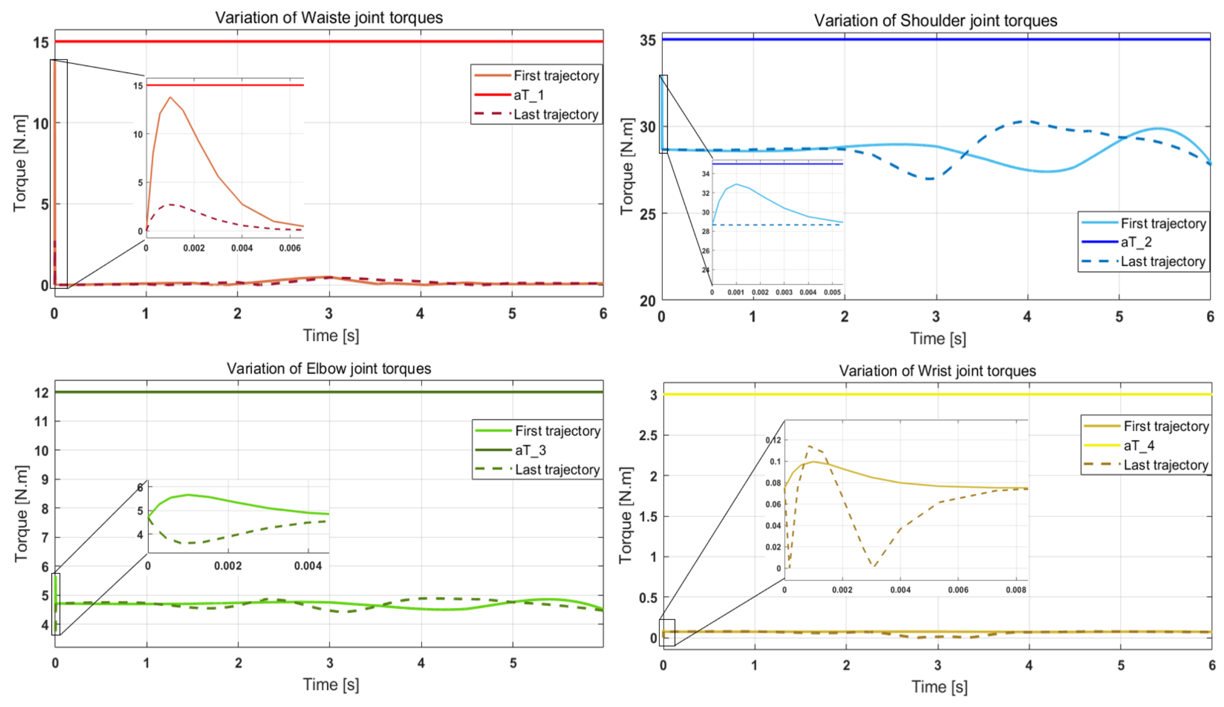


Figure 70. The joint torques variation for initial and final paths with ct=6[s]

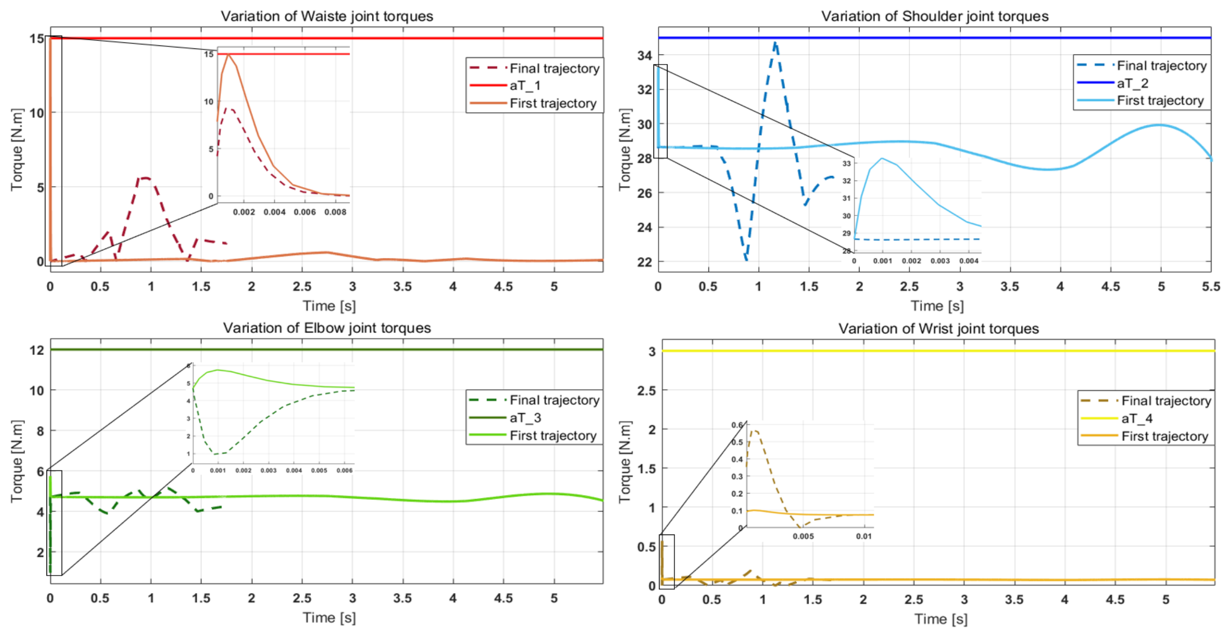


Figure 71. The joint torques variation for initial and final paths with their minimum cycle time values *ct*= 5.5 [s] and *ct*= 1.75 [s]

The hybrid algorithm based on Tabu-search is a totally newly elaborated optimization technique, which aims to refine the motion trajectory executed by the robot arm. As we can conclude from the analysis and the diagrams presented earlier that the hybrid algorithm optimizes the given trajectory by minimizing the cycle time of the path and the torque cost effect executed from joints.

10. THESES – NEW SCIENTIFIC RESULTS

T1. Performing the “card house” building task is suitable for modelling the collaboration of cooperative robots, which can be executed by one individual robot only if supporting element is used. Accomplishing this task using the RV-2SD 6 DoF and the RV-2AJ 5 DoF robot arms is impossible in the original vertical position of RV-2AJ because of the low 5 DoF. By setting this robot in a horizontal position, with the J5 axis becoming parallel to the J1 axis, the necessary orientations are achievable, and the suitable inclined orientation of the “cards” can be granted.
Verification: With the real implementation of the task using the RV-2AJ robot arm fixed in a horizontal position on a built stand [P3, P4, P5, P9, P11].

T2. Applying SolidWorks models of two cooperating robot arms (RV-2SD robot arm and RV-2AJ robot arm in horizontal placement) is suitable for modelling and simulating the “card house” building task in a CoppeliaSim virtual environment in both normal and abnormal scenarios.
Verification: Building of models and running simulations [P15, P16].

T3. Significant cycle time saving can be achieved by applying the newly elaborated “whip-lashing” method on the movement of the gripper of the RV-2AJ robot arm (which has RRR-RR kinematic chain) compared to the general path with simple angle interpolation at the joints. The efficiency of the elaborated “whip-lashing” method was confirmed by a case study, in which a cycle time saving of 33 % was achieved in case of motions not exceeding the maximum allowable joint torques.
Verification: Simulation by the application of MATLAB Simulink [P7, P13].

T4. Determination of quasi optimal trajectory for robot arms is possible with the elaborated new hybrid optimization algorithm that applies Tabu-search optimization method, trajectory point insertion and grid refinement. The trajectory calculation algorithm providing quasi-optimal cycle time has moderate, not combinatorically exploding time complexity.
Verification: Elaboration of the new hybrid optimization algorithm and application in a case study for proving the efficiency [P14].

11. SUMMARY

Reducing and maintain the lead time and increasing productivity present the major keys for the industry development especially after the industrial revolution known by Industry 4.0, The concept emerged the newest technologies in the manufacturing process, where the production cell became more flexible and smart within the installation of cooperative robots arm, these robots work in parallel way to execute the main task goal, many challenges are taking into consideration: regarding the flexibility to cooperate in the right way and the optimality of the motion control for each robot. From this global idea, the presented work describes the achievements gained during 7 semesters, starting from the basic scenario founded on creating a virtual smart manufacturing process that includes several robot arms endowed by different sensors, these robots as members of a multi-agent system can help each other and cooperate to finalize the appropriate line tasks using efficient algorithms. If some malfunction or another problem occurs in the production line, the robots can reconfigure themselves and reorganize the steps of the same task.

To reach the main target of the thesis, a real cooperative task was determined, by structuring all the scenario steps, hypotheses, and equipment needed to realize it in the real environment. The task is taking from our childhood when we learned how to build up a house of cards using two hands, therefrom our insight the hands present two robot arms. To start realizing the task, the training process about robot arms, motion trajectory, and control theory have been done.

After being familiar with industrial robot controlling, a manufacturing process of equipment needed has been started, where the task requires two robots and a set member of “cards”, these “cards” are manufactured using CNC machine to be sophisticated and guarantee the task requirements.

The “card house” building task is presented in two scenarios: the normal scenario executed using the cooperative work of two robot arms RV-2AJ of 5 Dof and RV-2SD of 6 Dof. The disturbing scenario executed in case of an error occurred the process and one of the robots will be out of order, in this case, the second robot should find a quasi-optimal solution to complete the building process, an excellent solution is developed in this scenario where we suppose that we have only one robot RV-2AJ arm and this robot should complete the building alone, the solution proposed based on using a support element which maintains the slanted “card” placed by RV-2AJ robot arm.

The availability of one robot in our institute pushed us to start with realizing the building house using only one robot. Within the building process using RV-2AJ arm only, the robot was not capable to pick and place precisely the “cards”, and this problem was due to the insufficient degrees of freedom, a new solution is proposed regarding this problem based on positioning the robot on a horizontal axis instead of the vertical one, the optimal formation of the workspace where the arm in the horizontal placement can move and rotate the end effector more precisely to eliminate the positioning errors providing stability of the “card house”.

In the meantime to complete the execution of both scenarios, we started to deal with the virtual environment using different simulation tools, wherein the execution of the task in the virtual environment can be controlled and done in a flexible way, The simulation part was done by modeling all the equipment of the task (RV-2AJ arm - RV-2SD – “card elements” - mechanical holder to maintain RV-2AJ horizontally), the modeling process was done using SolidWorks software, after this step the cad models were imported to CoppeliaSim environment which presents a powerful robotic software, the software allow us to control and execute the robot motion precisely as the real environment.

Dealing with motion robot problems from building the task in real and virtual environments inspired us to develop new methods with the aim to optimize the trajectory of an industrial robot arm, the first method is taking from the whip motion where we suppose that the motion arm can act as a whip, where it results in increasing the velocity of the robot arm’s parts during operation; therefore, it reduces the cycle time and utilizes the torque of joints more effectively. On the other hand, the second method deals with the refinement of robot trajectories in order to avoid obstacles presented in the workspace, the proposed theory is based on the Tabu-search algorithm which presents a powerful AI tool to find optimal solutions by testing the neighborhood corner points, the algorithm is endowed by two other techniques to minimize the inconvenience can find using Tabu-search.

FUTURE WORK

Due to the COVID 19 pandemic situation, the work in the laboratory was pended for one year, therefore in future work, we would like to complete the real process by installing RV-2SD and prove the cooperative task in the real environment.

As a future target also we would like to control the RV-2AJ arm with a developed controller using Raspberry Pi synchronized with our computer since the original controller of the robot is a black box, therefore, it will be a good direction to turn it into a slave and the computer will be the master, this solution gives us the flexibility to develop and study more effectively the control theory for industrial robot arms. On the other hand, completing the development of the Tabu-search algorithm with the aim to optimize the cycle time of motion and of process.

With the purpose to deal with AI part more effectively, I will figure out some optimization algorithms which can execute the steps without any error by executing Dijkstras algorithm to find the best trajectory and Tree search algorithm to order the rules and tasks of each robot on python and compare its features to produce an efficient algorithm.

ACKNOWLEDGEMENTS

Alhamdu lillahi .... First and foremost, I would like to praise and thank God, ***Allah***, the most gracious and the cherisher sustainer of the worlds, the almighty, who has granted countless blessings, knowledge, inspiration, and opportunity to me, so that I was able to accomplish my dissertation successfully.

Apart from my efforts to harvest the fruits of this work, the success of this thesis depends largely on the encouragement and guidelines of many others. Therefore, I would like to take this opportunity to express my gratitude to the people who have been instrumental in the successful completion of this large work.

To ***Dr. László Dudás,*** my supervisor, my friend and my second family in Hungary. Thank you for being a second father, for helping me, supporting me in the good and hard times. I am glad to meet you and collaborate with you, discovering your humanity and wisdom regarding the practical life, learning from you was the best gift for me.

To ***Dr. György Kovács,*** my co-supervisor, my friend, my second family in Hungary. Thank you for being with me in the good and hard times, your support during the pandemic, and the PhD period. I am grateful to work with you, and learn from you, your strategies and advice that always pushed me to be a good researcher and publisher in my research field.

I would like to extend my sincere and heartfelt obligation towards all members of the Institute of Information Science and the Faculty of Mechanical engineering and Informatics for their help, cooperation, and encouragement. A special thanks to ***Prof. Dr. Gabriella Vadaszné Bognár***, ***Prof. Dr. Miklós Tisza***, ***Dr. Samad Dadvandipour***, ***Dr. Péter Bikfalvi***, ***Dr. Tamás Szabó*** and for conscientious guidance and encouragement to accomplish this assignment.

To ***My parents***, my angels who always protect me with their blessings. I feel a deep sense of reverence and gratitude towards them, for their love, prayers, caring, and sacrifices for educating and preparing me for the future. Thank you both for giving me strength and trustees to reach for the stars and chase my dreams. I hope to be always a pride for you. I am very much thankful to my sister ***Amina***, my soul mate, and my best friend who always believes in me, and my brothers ***Ilyes*** and ***Mokhtar*** who have always supported me morally and encourage me.

At last but not least, gratitude goes to all of my friends who directly or indirectly helped me: ***Bochra, Sabah, Hanene, Chahra, Ikram, Feriel, Asma, Ines, Amina, Imene, Rosa, Meriem, Nesrine, Abderrahim***, ***Marouane, Amish, Amine, Fouad, Teo.***

Special thanks go to the persons who played an effective role in my study period:

To ***Seyf eddine***, who I always consider as a family member in Algeria. Thank you for your wise advice, and your friendship.

To ***Yassine***, who I always consider as a family member in Hungary. Thank you for being here for us, and for the keen interest shown to support me.

To ***Wafaa Cheikh***, my lovely sister in Hungary. Thank you for presenting me with this opportunity because you are always believing in me, and for being here beside me all the time.

All my thanks to my home country ***Algeria*** and the Stipendium Hungaricum Scholarship Program for giving me the opportunity to pursue my PhD study in ***Hungary***.

REFERENCES

[1] A. G. Frank, L. S. Dalenogare, and N. F. Ayala, “Industry 4.0 technologies: Implementation patterns in manufacturing companies,” *International Journal of Production Economics*, 2019, 210, pp. 15–26.

[2] T. Stock and G. Seliger, “Opportunities of Sustainable Manufacturing in Industry 4.0,” *Procedia CIRP*, 2016, 40, pp. 536–541.

[3] I. C. Dima, *Industrial production management in flexible manufacturing systems*. IGI Global, 2013.

[4] G. Kovács, “Combination of Lean value-oriented conception and facility layout design for even more significant efficiency improvement and cost reduction,” *International Journal of Production Research*, 2020 58(10), pp. 2916–2936.

[5] H. Ahuett-Garza and T. Kurfess, “A brief discussion on the trends of habilitating technologies for Industry 4.0 and Smart manufacturing,” *Manufacturing Letters*, 2018, 15, pp. 60–63.

[6] J. Lee, B. Bagheri, and H. A. Kao, “A Cyber-Physical Systems architecture for Industry 4.0-based manufacturing systems,” *Manufacturing Letters*, 2015, 3, pp. 18–23.

[7] J. Lee, H. Davari, J. Singh, and V. Pandhare, “Industrial Artificial Intelligence for industry 4.0-based manufacturing systems,” *Manufacturing Letters*, 2018,18, pp. 20–23.

[8] S. Luthra and S. K. Mangla, “Evaluating challenges to Industry 4.0 initiatives for supply chain sustainability in emerging economies,” *Process Safety and Environmental Protection*, 2018, 117, pp. 168–179.

[9] D. R. D. Sobrino, P. Košťál, D. Cagáňová, and M. Čambál, “On the possibilities of intelligence implementation in manufacturing the role of simulation,” *Applied Mechanics and Materials*, 2013, 309, pp. 96–104.

[10] C. Yildirim, B. Sevil Oflaç, and O. Yurt, “The doer effect of failure and recovery in multi-agent cases: service supply chain perspective,” *Journal of Service Theory and Practice*, 2018, 28(3), pp. 274–297.

[11] Á. Cservenák, “Simulation and modeling of a DC motor used in a mobile robot,” *Academic Journal Of Manufacturing Engineering*, 2020, 18(4), pp.183–190.

[12] A. Gilchrist, *Industry 4.0 - The Industrial Internet of Things*. Apress, 2016.

[13] P. Stone and M. Veloso, “Multiagent systems: a survey from a machine learning perspective,” *Autonomous Robots*, 2000, 8(3), pp. 345–383.

[14] I. Karabegović, “The Role of Industrial Robots in the Development of Automotive Industry in China,” *International Journal of Engineering Works Kambohwell Publisher Enterprises*, 2016, 3(12), pp. 92–97.

[15] P. G. Ranky, “Collaborative, synchronous robots serving machines and cells,” *Industrial Robot*, 2003, 30(3) pp. 213–217.

[16] P. Francesco and G. G. Paolo, “AURA: An Example of Collaborative Robot for Automotive and General Industry Applications,” *Procedia Manufacturing*, 2017, 11, pp. 338–345.

[17] B. Zhang, J. Wu, L. Wang, and Z. Yu, “Accurate dynamic modeling and control parameters design of an industrial hybrid spray-painting robot,” *Robotics and Computer-Integrated Manufacturing*, 2020, 63(101923).

[18] N. Jazdi, “Cyber physical systems in the context of Industry 4.0,” in *Proceedings of 2014 IEEE International Conference on Automation, Quality and Testing, Robotics, AQTR 2014*, 2014.

[19] Y. Hu, Y. Dong, and Batunacun, *An automatic approach for land-change detection and land updates based on integrated NDVI timing analysis and the CVAPS method with GEE support*, 2018, 146.

[20] T. K. Sung, “Industry 4.0: A Korea perspective,” *Technological Forecasting and Social Change*, 2017, 132, pp. 40–45.

[21] The Boston Consulting Group (BCG), “Industry 4.0: The Future of Productivity and Growth in Manufacturing Industries,” *The Boston Consulting Group*, 2015.

[22] R. Y. Zhong, X. Xu, E. Klotz, and S. T. Newman, “Intelligent Manufacturing in the Context of Industry 4.0: A Review,” *Engineering*, 2017, 3(5), pp. 616–630.

[23] G. Miklós and K. György, “Industry 4.0 conception,” *Acta Technica Corviniensis -Bulletin of Engineering*, 2017, 1, pp. 1–4.

[24] J. Smit, S. Kreutzer, C. Moeller, and M. Carlberg, “Industry4.0 a Study for the European Parliament,” Brussels, 2016.

[25] F. Shrouf, J. Ordieres, and G. Miragliotta, “Smart factories in Industry 4.0: A review of the concept and of energy management approached in production based on the Internet of Things paradigm,” in *IEEE International Conference on Industrial Engineering and Engineering Management*, 2014, 2015, pp. 697–701.

[26] S. Vaidya, P. Ambad, and S. Bhosle, “Industry 4.0 - A Glimpse,” in *Procedia Manufacturing*, 2018, 20, pp. 233–238.

[27] N. Mostafa, W. Hamdy, and H. Alawady, “Impacts of Internet of Things on Supply Chains: A&#13; Framework for Warehousing,” *Social Sciences*, 2019, 8(3).

[28] Á. Ballagi, L. T. Kóczy, and C. Pozna, "Intelligent robot cooperation with fuzzy communication," *Studies in Computational Intelligence*, 2014, 530, pp. 185–197.

[29] P. Corke, R. Peterson, and D. Rus, "Localization and Navigation Assisted by Networked Cooperating Sensors and Robots," *The International Journal of Robotics Research*, 2005, 24(9), pp. 771–786.

[30] P. U. and L. M. M. Custodio, "Artificial Intelligence and Systems Theory: Applied to Cooperative Robots," *International Journal of Advanced Robotic Systems*, 2004, 1(3).

[31] S. Makris, G. Michalos, A. Eytan, and G. Chryssolouris, "Cooperating robots for reconfigurable assembly operations: Review and challenges," in *Procedia CIRP*, 2012, 3(1), pp. 346–351.

[32] E. Magrini, F. Ferraguti, A. J. Ronga, F. Pini, A. De Luca, and F. Leali, "Human-robot coexistence and interaction in open industrial cells," *Robotics and Computer-Integrated Manufacturing*, 2020, (61).

[33] S. Robla-Gomez, V. M. Becerra, J. R. Llata, E. Gonzalez-Sarabia, C. Torre-Ferrero, and J. Perez-Oria, "Working Together: A Review on Safe Human-Robot Collaboration in Industrial Environments," *IEEE Access*, 2017, 5, pp. 26754–26773.

[34] A. Schönberg and R. Schmitt, "Cooperation of Industrial Robots with Indoor-GPS AIMFREE-Agile Montage von Elektrofahrzeugen durch freie Verkettung View project INTERAQCT View project," 2014.

[35] O. A. Amodu and M. Othman, "Machine-to-Machine Communication: An Overview of Opportunities," *Computer Networks*, 2018, 145, pp. 255–276.

[36] N. El Zoghby, V. Loscrí, E. Natalizio, and V. Cherfaoui, "Robot cooperation and swarm intelligence," in *Wireless Sensor and Robot Networks: From Topology Control to Communication Aspects*, World Scientific Publishing Co., 2013, pp. 163–201.

[37] Z. Yan, N. Jouandeau, and A. A. Cherif, "A survey and analysis of multi-robot coordination," *International Journal of Advanced Robotic Systems*, 2013, 10.

[38] J. Xu, E. Huang, L. Hsieh, L. H. Lee, Q. S. Jia, and C. H. Chen, "Simulation optimization in the era of Industrial 4.0 and the Industrial Internet," *Journal of Simulation*, 2016, 10(4), pp. 310–320.

[39] A. Hernansanz, A. Casals, and J. Amat, "A multi-robot cooperation strategy for dexterous task oriented teleoperation," *Robotics and Autonomous Systems*, 2015, 68, pp. 156–172.

[40] N. Papakostas, G. Pintzos, M. Matsas, and G. Chryssolouris, "Knowledge-enabled design of cooperating robots assembly cells," in *Procedia CIRP*, 2014, 23, pp. 165–170.

[41] K. Wissama and D. Etienne, *Modélisation, identification et commande des robots* , 2nd ed. France: Hermés: Lavoisier, 1999.

[42] Z. Luo, *Robotics, automation, and control in industrial and service settings*. IGI Global, 2015.

[43] F. L. Lewis, D. M. Dawson, and C. T. Abdallah, *Robot Manipulator Control: Theory and Practice* , 2nd ed. CRC Press, 2003.

[44] D. Etienne and K. Wisama, *Robot Manipulators: Modeling, Performance Analysis and Control* . Wiley-ISTE, 2013.

[45] H. A. F. Almurib, H. F. Al-Qrimli, and N. Kumar, "A review of application industrial robotic design," in *International Conference on ICT and Knowledge Engineering*, 2011, pp. 105–112.

[46] M. Baboria and M. Kaith, "Literature Review on Design and Simulation of Industrial Robotic Arm using CAD/CAM Software," *International Journal of Engineering Science and Computing*, 2018, 8(3).

[47] M. Hägele, K. Nilsson, and J. N. Pires, "Industrial Robotics," in *Springer Handbook of Robotics*, Berlin, Heidelberg: Springer Berlin Heidelberg, 2008, pp. 963–986.

[48] K. M. Lynch and F. C. Park, *Modern Robotics: Mechanics, Planning, and Control*. Cambridge University Press, 2017.

[49] W. Yu, *PID control with intelligent compensation for exoskeleton robots*. Elsevier, 2018.

[50] C. Hughes and T. Hughes, *Robot Programming: A Guide to Controlling Autonomous Robots*. 2016.

[51] F. Fahimi, *Autonomous Robots*. Springer US, 2009.

[52] R. Zhao, "Trajectory planning and control for robot manipulations," Université Paul Sabatier - Toulouse III, Toulouse, 2015.

[53] M. Brady, J. Hollerbach, J. Timothy L., T. Lozano-Pérez, and M. T. Mason, Eds., *Robot Motion: Planning and Control*. MIT press, 1983.

[54] S. B. Niku, *Introduction to Robotics: Analysis, Control, Applications*, 2nd ed. Wiley, 2010.

[55] X. Liu, C. Qiu, Q. Zeng, and A. Li, "Kinematics analysis and trajectory planning of collaborative welding robot with multiple manipulators," in *Procedia CIRP*, 2019, 81, pp. 1034–1039.

[56] S. Kucuk and Z. Bingul, "Robot Kinematics: Forward and Inverse Kinematics," in *Industrial Robotics: Theory, Modelling and Control*, IntechOpen, 2006, pp. 117–148.

[57] J. J. Craig, *Introduction to Robotics Mechanics and Control Third Edition*, 3rd ed. Pearson Education International, 2005.

[58] P. Coiffet, *Les robots . Tome 1 . Modélisation et commande / par Philippe Coiffet,...* France: Hermes publishing, 1981.

[59] B. Siciliano, L. Sciavicco, L. Villani, and G. Oriolo, *Robotics*. London: Springer London, 2009.

[60] H. R. Kim, J. S. Hong, and K. C. Ko, “Optimal design of industrial manipulator trajectory for minimal time operation,” *KSME Journal*, 1990, 4(1), pp. 3–9.

[61] M. Straka, E. Žatkovič, and R. Schréter, “Simulation as a means of activity streamlining of continuously and discrete production in specific enterprise,” *Acta Logistica-International Scientific Journal*, 2014, (3), pp. 11–16.

[62] Q. V. Doan, A. T. Vo, T. D. Le, H.-J. Kang, and N. H. A. Nguyen, “A Novel Fast Terminal Sliding Mode Tracking Control Methodology for Robot Manipulators,” *Applied Sciences*, 2020, 10(9).

[63] S.-H. Joo *et al.*, “Autonomous Navigation Framework for Intelligent Robots Based on a Semantic Environment Modeling,” *Applied Sciences*, 2020, 10(9).

[64] J. Kim, S. R. Kim, S. J. Kim, and D. H. Kim, “A practical approach for minimum-time trajectory planning for industrial robots,” *Industrial Robot*, 2010, 37(1), pp. 51–61.

[65] S. Perumaal and N. Jawahar, “Synchronized trigonometric S-curve trajectory for jerk-bounded time-optimal pick and place operation,” *International Journal of Robotics and Automation*, 2012, 27(4), pp. 385–395.

[66] O. Avram and A. Valente, “Trajectory Planning for Reconfigurable Industrial Robots Designed to Operate in a High Precision Manufacturing Industry,” in *Procedia CIRP*, 2016, 57, pp. 461–466.

[67] S. Macfarlane and E. A. Croft, “Jerk-bounded manipulator trajectory planning: Design for real-time applications,” *IEEE Transactions on Robotics and Automation*, 2003, 19(1), pp. 42–52.

[68] A. Gasparetto, A. Lanzutti, R. Vidoni, and V. Zanotto, “Experimental validation and comparative analysis of optimal time-jerk algorithms for trajectory planning,” *Robotics and Computer-Integrated Manufacturing*, 2012, 28(2), pp. 164–181.

[69] H. Liu, X. Lai, and W. Wu, “Time-optimal and jerk-continuous trajectory planning for robot manipulators with kinematic constraints,” *Robotics and Computer-Integrated Manufacturing*, 2013, 29(2), pp. 309–317.

[70] J. R. García Martínez, J. Rodríguez Reséndiz, M. Á. Martínez Prado, and E. E. Cruz Miguel, “Assessment of jerk performance s-curve and trapezoidal velocity profiles,” in *2017 13th International Engineering Congress, CONIIN 2017*, 2017.

[71] Y. Fang, J. Hu, W. Liu, Q. Shao, J. Qi, and Y. Peng, “Smooth and time-optimal S-curve trajectory planning for automated robots and machines,” *Mechanism and Machine Theory*, vol. 137, pp. 127–153, Jul. 2019.

[72] K. Zheng, Y. Hu, and B. Wu, “Trajectory planning of multi-degree-of-freedom robot with coupling effect,” *Journal of Mechanical Science and Technology*, 2019, 33(1), pp. 413–421.

[73] A. Gasparetto and V. Zanotto, “A new method for smooth trajectory planning of robot manipulators,” *Mechanism and Machine Theory*, 2007, 42(4), pp. 455–471.

[74] E. K. Xidias, “Time-optimal trajectory planning for hyper-redundant manipulators in 3D workspaces,” *Robotics and Computer-Integrated Manufacturing*, vol. 50, pp. 286–298, Apr. 2018.

[75] A. R. Hirakawa and A. Kawamura, “Trajectory planning of redundant manipulators for minimum energy consumption without matrix inversion,” in *Proceedings - IEEE International Conference on Robotics and Automation*, 1997, 3, pp. 2415–2420.

[76] F. Z. Baghli, L. El Bakkali, and Y. Lakhal, “Optimization of Arm Manipulator Trajectory Planning in the Presence of Obstacles by Ant Colony Algorithm,” in *Procedia Engineering*, 2017, 181, pp. 560–567.

[77] S. F. P. Saramago and V. Steffen, “Optimization of the trajectory planning of robot manipulators taking into account the dynamics of the system,” *Mechanism and Machine Theory*, 1998, 33(7), pp. 883–894.

[78] D. P. Garg and M. Kumar, “Optimization techniques applied to multiple manipulators for path planning and torque minimization,” *Engineering Applications of Artificial Intelligence*, 2002, 15(3), pp. 241–252.

[79] Y. Zhang, S. S. Ge, and T. H. Lee, “A unified quadratic-programming-based dynamical system approach to joint torque optimization of physically constrained redundant manipulators,” *IEEE Transactions on Systems, Man, and Cybernetics, Part B: Cybernetics*, 2004, 34(5), pp. 2126–2132.

[80] H. Ding, Y. F. Li, and S. K. Tso, “Dynamic optimization of redundant manipulators in worst case using recurrent neural networks,” *Mechanism and Machine Theory*, 2000, 35(1), pp. 55–70.

[81] P. Saxena, P. Stavropoulos, J. Kechagias, and K. Salonitis, “Sustainability Assessment for Manufacturing Operations,” *Energies*, 2000, 13(11).

[82] W. Wang and K. Siau, “Artificial intelligence, machine learning, automation, robotics, future of work and future of humanity: A review and research agenda,” *Journal of Database Management*, 2019, 30(1), pp. 61–79.

[83] M. Awad and R. Khanna, “Machine Learning,” in *Efficient Learning Machines*, Berkeley, CA: Apress, 2015, pp. 1–18.

[84] B. Siciliano and O. Khatib, “Robotics and the handbook,” in *Springer Handbook of Robotics*, Springer International Publishing, 2016, pp. 1–6.

[85] V. François-Lavet, P. Henderson, R. Islam, M. G. Bellemare, and J. Pineau, “An introduction to deep reinforcement learning,” *Foundations and Trends in Machine Learning*, 2018, 11(3), pp. 219–354.

[86] M. S. Ashfaq, “A Tribute to Father of Fuzzy Set Theory and Fuzzy Logic ,” *International Journal of Swarm Intelligence and Evolutionary Computation*, 2018, 7(2).

[87] E. Bjorlykhaug and O. Egeland, "Mechanical design optimization of a 6DOF serial manipulator using genetic algorithm," *IEEE Access*, 2018, 6, pp.59087–59095.

[88] M. N. Ab Wahab, S. Nefti-Meziani, and A. Atyabi, "A Comprehensive Review of Swarm Optimization Algorithms," *PLOS ONE*, 2015, 10(5).

[89] F. A. Raheem and U. I. Hameed, "Heuristic D* Algorithm Based on Particle Swarm Optimization for Path Planning of Two-Link Robot Arm in Dynamic Environment," *Al-Khwarizmi Engineering Journal*, 2019, 15(2), pp. 108–123.

[90] E. C. Salander, *Computer Search Algorithms* . Nova Science Pub Inc, 2011.

[91] F. Glover, "Tabu Search—Part I," *ORSA Journal on Computing*, 2019, 1(3), pp. 190–206.

[92] M. Alshibli, A. El Sayed, E. Kongar, T. M. Sobh, and S. M. Gupta, "Disassembly Sequencing Using Tabu Search," *Journal of Intelligent and Robotic Systems: Theory and Applications*, 2016, 82(1), pp. 69–79.

[93] F. Glover and M. Laguna, *Tabu Search*. Springer US, 1997.

[94] L. Hofer, "Decision-making algorithms for autonomous robots," Université de Bordeaux, France, 2017.

[95] F. Torres, S. Puente, and C. Díaz, "Automatic cooperative disassembly robotic system: Task planner to distribute tasks among robots," *Control Engineering Practice*, 2009, 17(1), pp. 112–121.

[96] "RV-1A/RV-2AJ Robot Arm Setup & Maintenance ," 2001.

[97] "RV-2SD/2SDB Instruction Manual Robot Arm Setup & Maintenance ," 2010.

[98] G. Onwubolu, *A Comprehensive Introduction to SolidWorks 2013*. SDC Publications , 2013.

[99] W. Khalil and E. Dombre, *Modeling, Identification and Control of Robots*. Elsevier Ltd, 2004.

[100] C. D. Crane, III and J. Duffy, *Kinematic Analysis of Robot Manipulators*. Cambridge University Press, 1998.

[101] M. A. Ayob, W. N. W. Zakaria, and J. Jalani, "Forward kinematics analysis of a 5-axis RV-2AJ robot manipulator," in *Proceedings - 2014 Electrical Power, Electronics, Communications, Control and Informatics Seminar, EECCIS 2014. In conjunction with the 1st Joint Conference UB-UTHM*, 2014, pp. 87–92.

[102] J. Denavit and R. S. Hartenberg, "A kinematic notation for lower pair mechanismbased on matrices," *Journal of Applied Mechanics*, 1955, 22, pp. 215–221.

[103] R. P. Paul, *Robot Manipulators: Mathematics, Programming, and Control*. Cambridge: The Mit Press, 1981.

[104] M. A. Qassem, I. Abuhadrous, and H. Elaydi, "Modeling and simulation of 5 DOF educational robot arm," in *Proceedings - 2nd IEEE International Conference on Advanced Computer Control, ICACC 2010*, 2010, 5, pp. 569–574.

[105] M. Renaud, "Calcul de la matrice Jacobienne nécessaire à la commande coordonnée d'un manipulateur," *Mechanism and Machine Theory*, 1980, 15(2), pp. 81–91.

[106] M. Renaud and B. Gorla, *Modèles des robots manipulateurs : application à leur commande* . Toulouse: Cepadues-Editions, 1984.

[107] M. Haklidir and I. Tasdelen, "Modeling , simulation and fuzzy control of an anthropomorphic robot arm by using Dymola", 2009, pp. 177–186.

[108] J. Y. S. Luh, M. W. Walker, and R. P. C. Paul, "On-line computational scheme for mechanical manipulators," *Journal of Dynamic Systems, Measurement and Control, Transactions of the ASME*, 1980, 102(2), pp. 69–76.

[109] V. Potkonjak, "Thermal analysis and dynamic capabilities of DC motors in industrial robotic systems," *Robotics and Computer Integrated Manufacturing*, 1989, 5(2–3), pp. 137–143.

[110] W. Khalil and J.-F. Kleinfinger, "Minimum operations and minimum parameters of the dynamic models of tree structure robots," *IEEE Journal on Robotics and Automation*, 1987, 3(6), pp. 517–526.

[111] P. K. Khosla, "Real-time control and identification of direct-drive manipulators ," Carnegie Mellon University, USA, 1986.

[112] P. Chedmail and M. Gautier, "Optimum choice of robot actuators," *Journal of Manufacturing Science and Engineering, Transactions of the ASME*, 1990, 112(4), pp. 361–367.

[113] P. Corke, "Introduction," in *Springer Tracts in Advanced Robotics*, vol. 118, Springer Verlag, 2017, pp. 1–14.

[114] "Path and motion planning." [Online]. Available: https://www.coppeliarobotics.com/helpFiles/en/pathAndMotionPlanningModules.htm. [Accessed: 18-Feb-2021].

[115] K. Perutka, *MATLAB for Engineers - Applications in Control, Electrical Engineering, IT and Robotics*. InTech, 2012.

[116] Z. Deng, H. Zhu, and S. Lin, "Adaptive Sliding Mode Control Method For Reconfigurable Modular Robots Under Dynamic Constraints," *Academic Journal of Manufacturing Engineering*, 2020, 18(1), pp. 1–5.

[117] "Robot simulator CoppeliaSim: create, compose, simulate, any robot - Coppelia Robotics." [Online]. Available: https://www.coppeliarobotics.com/. [Accessed: 18-Feb-2021].

[118] A. Goriely, and T. McMillen, “Shape of a cracking whip,” *Physical Review Letters*, 2002, 88(24), pp. 2443011–2443014.
[119] C. Henrot, “Characterization of Whip Targeting Kinematics in Discrete and Rhythmic Tasks,” Massachusetts Institute of Technology, USA, 2016.
[120] P. Krehl, S. Engemann, and D. Schwenkel, “The puzzle of whip cracking - Uncovered by a correlation of whip-tip kinematics with shock wave emission,” *Shock Waves*, 1998, 8(1), pp. 1–9.

LIST OF PUBLICATIONS RELATED TO THE TOPIC OF THE RESEARCH FIELD

(P1) R. Benotsmane, M. R. Benachenhou, L. Dudás, and G. Kovács, Preliminary study of 6 axis manipulator arm, Doktoranduszok Foruma, 2017, pp. 82–90.

(P2) R. Benotsmane, S. E. Kacemi, and M. R. Benachenhou, Calculation methodology for trajectory planning of a 6 axis-manipulator arm. Annals of Faculty Engineering Hunedoara – International Journal of Engineering. 2018, 3, pp. 27–32.

(P3) R. Benotsmane, L. Dudás, G. Kovács, Collaborating robots using artificial intelligence, Multiscience XXXII. MicroCAD International Multidisciplinary Scientific Conference, Miskolc-Egyetemváros, Hungary, 5-6 September 2018, pp. 1–2.

(P4) R. Benotsmane, L. Dudás, G. Kovács, Cooperating robots using artificial intelligence, in 6th International Scientific Conference on Advances in Mechanical Engineering (ISCAME 2018), Debrecen, Hungary, 11-12 October 2018, pp. 17–18.

(P5) R. Benotsmane, L. Dudás, and G. Kovács, Collaborating robots in Industry 4.0 conception. In Proceedings of the XXIII International Conference on Manufacturing, IOP Conference Series: Materials Science and Engineering, Kecskemét, Hungary, 7–8 June 2018, pp. 1–9. (Scopus index)

(P6) L. Dudás, P. Kapitány, R. Benotsmane, Komplex kapcsolódó felületpárok gyártástechnológiai elemzése, Műszaki Tudomány az Észak-Kelet Magyarországi Régióban 2018: konferencia előadása, Debrecen, Hungary, 29 May 2019, pp. 77–83.

(P7) R. Benotsmane, G. Kovács, and L. Dudás, Economic, social impacts and operation of smart factories in Industry 4.0 focusing on simulation and artificial intelligence of collaborating robots. Social sciences. 2019, 8, 143. (Q3 + IF: 1.3, Scopus index)

(P8) R. Benotsmane, G. Kovács, and L. Dudás, The concept of autonomous systems in industry 4.0, Advanced logistics systeme - Theory practice, 2019, 12 (1), pp. 77–87.

(P9) R. Benotsmane, L. Dudás, G. Kovács, Optimization of pick and place task problem for a manipulator arm, in 7th International Scientific Conference on Advances in Mechanical Engineering (ISCAME 2019), Debrecen, Hungary, 7-9 November 2019, pp. 19–20. (Q3 + Scopus index)

(P10) R. Benotsmane, L. Dudás, and G. Kovács, Survey on new trends of robotic tools in the automotive industry. In Vehicle and Automotive Engineering 3; VAE 2020. Lecture notes in Mechanical Engineering; Springer: Singapore, 2021, pp. 443–457. (Q3 + Scopus index)

(P11) R. Benotsmane, L. Dudás, and G. Kovács, Trial - and - Error Optimization Method of Pick and Place Task for RV-2AJ Robot Arm, In Vehicle and Automotive Engineering 3; VAE 2020. Lecture notes in Mechanical Engineering; Springer: Singapore, 2021, pp. 458–467. (Q3 + Scopus index)

(P12) R. Benotsmane and L. Dudás, Robotic Production Oriented Engine Design and Manufacturing In Vehicle and Automotive Engineering 3; VAE 2020. Lecture notes in Mechanical Engineering; Springer: Singapore, 2021, pp. 390–400. (Q3 + Scopus index)

(P13) R. Benotsmane, L. Dudás, and G. Kovács, Trajectory Optimization of Industrial Robot Arms Using a Newly Elaborated 'Whip-Lashing' Method, Applied sciences journal, 2020, 10 (23), pp. 1–18. (Q1 + IF: 2.47, Scopus index)

(P14) R. Benotsmane, L. Dudás, and G. Kovács, Survey on artificial intelligence algorithms used in industrial robotics, Multidiszciplináris tudományok, 2020, 10 (4), pp. 194–205.

(P15) R. Benotsmane, L. Dudás, and G. Kovács, Simulation and trajectory optimization of collaborating robots by application of solidworks and matlab software in industry 4.0, Academic journal of Manufacturing Engineering, 2020, 18 (4), pp. 191–197. (Q3 + Scopus index)

(P16) R. Benotsmane, S. Kacemi, L. Dudás, and G. Kovács, Simulation of industrial robot's six axes manipulator arms - a case study, Academic journal of Manufacturing Engineering, 2021, 19 (1), pp. 1–9. (Q3 + Scopus index)

APPENDICES

A1 Characteristics of an industrial robot arm

A2 AI methods and their possible use for a robotic arm

A3 Technical measurements of RV-2AJ and RV-2SD robot arms

A4 Modeling analysis calculation

A5 Hybrid algorithm for optimization of robot arms’ trajectory

Appendix 1: Characterestics of an industrial arm

Joints are attached with sensors that read join position and speed, the joint primitives describe how adjacent links are connected to each other. Two primitive joint types defined in this area, see **Figure A.1**:

- **Prismatic (sliding) joint:** Pair of links makes a translational displacement along a fixed axis. One link slides on the other along a straight line.
- **Revolute (rotary) joint:** Two links rotate about a fixed axis. This type of joint is often referred to as a hinge, articulated, or rotational joint.

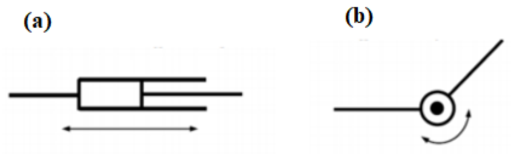


Figure A.1. Prismatic joint (a) and revolute joint (b)

Most of the industrial robots are serial combinations of revolute and prismatic joints. The most fundamental functional requirements for a robotic system is to be able to locate its end-effector (hand, tool, or end-device) in 3D space with respect to the world coordinate frame.

A typical industrial arm is made up of seven metal segments as a human arm, joined by six joints as present in **Figure A.2** The computer controls the robot by rotating individual motors connected to each joint. This allows the computer to move the arm very precisely, repeating exactly the same movement over and over again. The robot uses motion sensors to make sure it moves just the right amount.

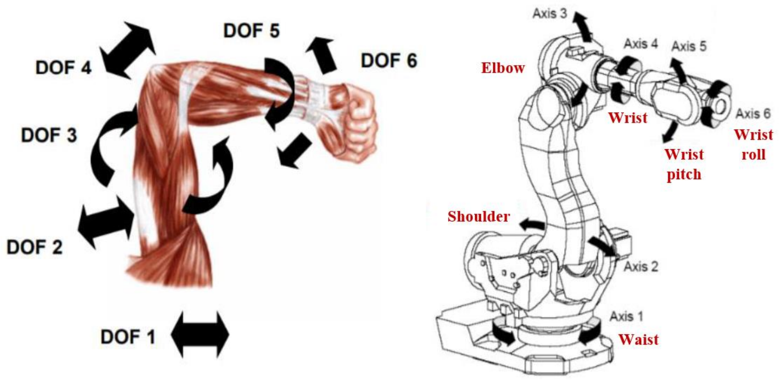


Figure A.2. Human arm behind an industrial arm

A robot must be chosen according to the application reserved for it. Here are some parameters to take:

- Workspace: this is the set of situations in space that the terminal organ of the robot can reach. It is defined by its limits, imposed essentially by the number of degrees of freedom, the deflections articular and by the length of the segments of the manipulator.
- The payload: or the maximum load transportable by the robot.
- The maximum speeds and accelerations: which condition the times of cycle.
- The mechanical structure articulated: the choice is guided by the task to be realized.
- The resolution: it is the smallest modification of the configuration of the robot both observable and controllable by the control system.

Closed-loop control scheme at the end-effector level

The end–effector's state $\boldsymbol{x}$ is measured by the sensing devices and the corresponding measurement $\hat{\boldsymbol{x}}$ is given as feedback; then is compared to the desired state $\boldsymbol{x_d}$, the difference is fed to the regulator and the regulation output is sent to the robot controller to move the manipulator as seen in **Figure A.3** The joint level loop is also displayed.

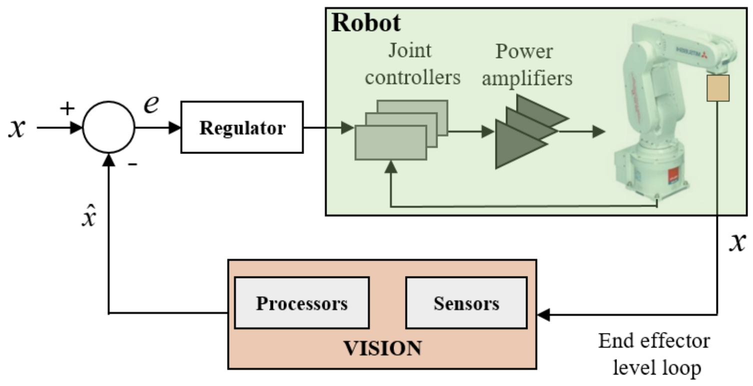


Figure A.3. Control scheme of industrial robot

For tasks requiring low accuracy a blind trajectory execution with closed-loop control at the joint level is acceptable. However, for the tasks that require high accuracy, closed-loop control at the end-effector level is necessary, during which the next interpolation points have to be calculated in each interpolation cycle using external input; this need is satisfied by the open architecture robot controllers, which unlike the typical closed architecture controllers, permit the user or an external system to define precisely the interpolation points and oblige the end-effector to strictly pass through them. However, this restriction may produce oscillations at the end-effector level depending partially on the coarseness of the trajectory. These oscillations may be avoided if an additional regulator is used.

Appendix 2: AI methods and their possible use for a robotic arm

Table A.1. AI methods and their possible use for a robotic arm

AI Methods	Role Specification
Artificial Neural Networks	Generally used to carry out real-time flexible pattern recognition in industry using a variety of sensors. ANN can analyze very large amounts of data in real time and offer concrete answers to problems that may arise from the production process. Generalization and self-learning capabilities are used in handling modified conditions and scenarios.
Reinforcement Learning	RL is a branch of ML where an agent learns to interact in its workspace by performing actions and seeing the results, is used e.g. for calculating inverse kinematic problem of a robot arm which is a complicated task based on calculating the angles needed for each joint to reach the end effector, the process inputs are a set of equations that present the homogeneous transformation matrixes between the links, joints and the position and orientation vectors.
Logic-Based AI	Knowledge representation and inference in all tasks and decision situations treated by the AI where binary logic is sufficient, like making a decision on starting an investment.
Fuzzy Logic	Imitation of the way of decision making in humans that involves all intermediate possibilities between the extreme values YES and NO. Its effect in industrial robotics can be appear in the training of a robotic arm to plan its movements to avoid a collision with obstacles in its workspace, or in collaborative robot system where executing a parallel task, the use of fuzzy logic can facilitate which robot has more ability to achieve better the task.

Genetic Algorithm	Very robust methods based on search heuristic inspired by Charles Darwin’s theory of natural evolution, used in quasi-optimum searching when time is not critical and model formulation is problematic, useful also for solving the point-to-point trajectory planning for redundant robot arm. Usually the algorithm uses different fitness criteria to reach an optimal result as inverse kinematic problem, the minimization of energy consumption, minimization of operation time as well as the criterion of minimal total angular changes of the robotic arm.
Swarm Intelligence	Creation of multi-agent systems and complex structures characterized by self-reorganization and collective behavior planning and decision making capability.

Appendix 3: Technical measurments of RV-2AJ and RV-2SD robot arms

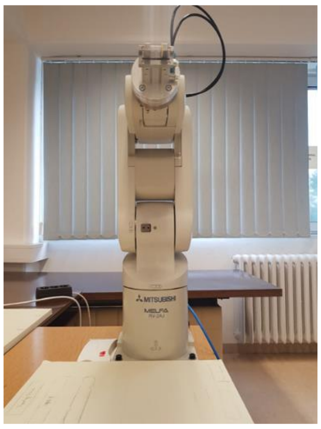


Figure A.4. RV-2AJ arm

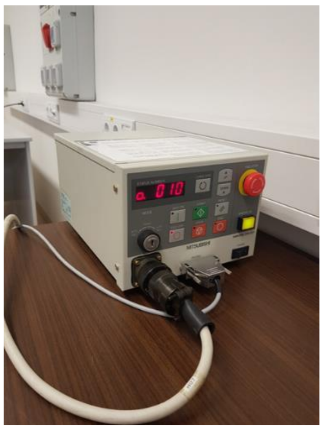


Figure A.5. Controller

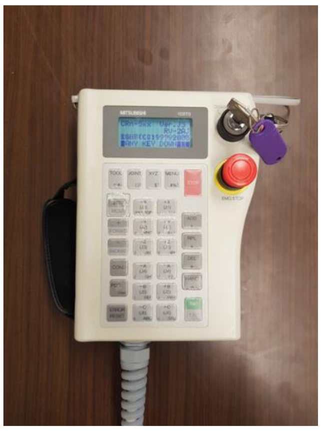


Figure A.6. Teaching box

Figure A.7. COSIROP software

Figure A.8. Air compressor

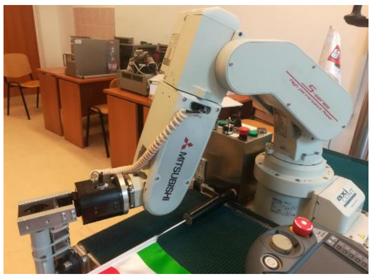


Figure A.9. RV-2SD arm

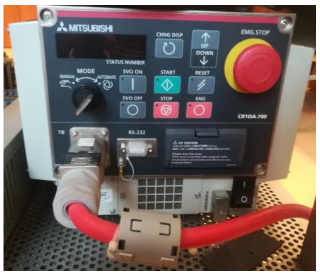


Figure A.10. Controller CR1DA-700 series

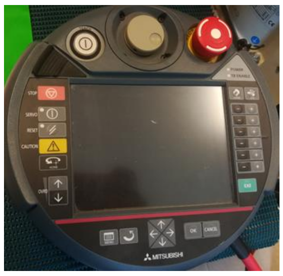


Figure A.11. R56 Teaching box

Table A.2. Specifications of RV-2AJ

Joints number	5 rotational joints
Payload	2 kg
Speed	2100 mm/sec
Repeatability	+/- 0.020 mm
Reach length	410 mm
Programming language	MELFA-BASIC IV Language
Gripper	Pneumatic type

Table A.5. Specifications of RV-2SD

Joints number	6 rotational joints
Payload	3 kg
Speed	4400 mm/sec
Repeatability	+/- 0.02 mm
Arm reachable	504 mm
Programming language	MELFA-BASIC IV Language
Gripper	Electric type

Table A.3. Length of links for RV-2AJ

Link	Distance
1: Waist to shoulder	300 mm
2: Shoulder to elbow	250 mm
3: Elbow to wrist pitch	160 mm
4: Wrist pitch to wrist roll	72 mm
5: Gripper	70 mm

Table A.6. Length of links for RV-2SD

Link	Distance
1: Waist to shoulder	
2: Upper arm	230 mm
3: Elbow to wrist twist	37.5mm
4: Forearm	270 mm
5: Wrist pitch to wrist roll	70 mm
6: Gripper	

Table A.4. Operating range for RV-2AJ

J1: Waist	-150° to +150°
J2: Shoulder	-60° to +120°
J3: Elbow	-110° to +120°
J5: Wrist pitch	-90° to +90°
J6: Wrist roll	-200° to +200°

- Joint 4 does not exist

Table A.7. Operating range for RV-2SD

J1: Waist	-240° to +240°
J2: Shoulder	-120° to +120°
J3: Elbow	0° to +160°
J4: Wrist twist	-200° to +200°
J5: Wrist pitch	-120° to +120°
J6: Wrist roll	-360° to +360°

Appendix 4: Modeling analysis calculation

Denavit-Hartenberg parameters for forward and inverse geometry models

The links are numbered such that link 0 constitutes the base of the robot and link $\boldsymbol{n}$ is the terminal link **Figure A.12**.

Joint $\boldsymbol{j}$ connects link $\boldsymbol{j}$ to link $\boldsymbol{j-1}$ and its variable is denoted $\boldsymbol{q_j}$. In order to define the relationship between the location of links, we assign a frame $\boldsymbol{R_j}$. attached to each link $\boldsymbol{j}$, such that:

- the $\mathbf{z}_j$ axis is along the axis of joint $\boldsymbol{j}$;
- the $\mathbf{x}_j$ axis is aligned with the common normal between $\mathbf{z}_j$ and $\mathbf{z}_{j+1}$. If $\mathbf{z}_j$ and $\mathbf{z}_{j+1}$ are parallel or collinear, the choice of $\mathbf{x}_j$ is not unique. The intersection of $\mathbf{x}_j$ and $\mathbf{z}_j$ defines the origin $\boldsymbol{O_j}$. In the case of intersecting joint axes, the origin is at the point of intersection of the joint axes;
- the $\mathbf{y}_j$ axis is formed by the right-hand rule to complete the coordinate system $(\mathbf{x}_j, \mathbf{y}_j, \mathbf{z}_j)$.

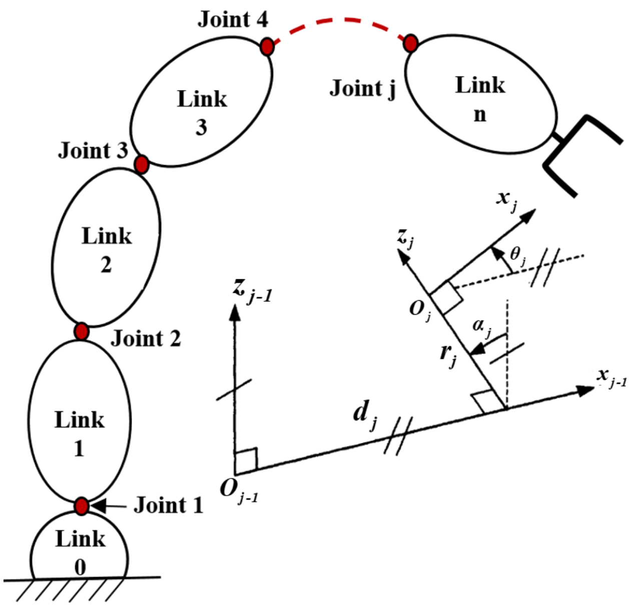


Figure A.12. The geometric parameters in the case of a simple open structure

The transformation matrix from frame $\boldsymbol{R_{j-1}}$ to frame $\boldsymbol{R_j}$ is expressed as a function of the following four geometric parameters:

- α_j: the angle between $\mathbf{z}_{j-1}$ and $\mathbf{z}_j$ about $\mathbf{x}_{j-1}$;
- d_j: the distance between $\mathbf{z}_{j-1}$ and $\mathbf{z}_j$ along $\mathbf{x}_{j-1}$;

- θ_j: the angle between $\mathbf{x}_{j-1}$ and $\mathbf{x}_j$ about $\mathbf{z}_j$;
- r_j: the distance between $\mathbf{x}_{j-1}$ and $\mathbf{x}_j$ along $\mathbf{z}_j$.

The variable of joint j, defining the relative orientation or position between links $j-1$ and j, is either O_j or r_j, depending on whether the joint is revolute or prismatic respectively. This is defined by the relation:

$$q_j = \bar{\sigma}_j \theta_j + \sigma_j r_j$$

with:

- σ_j = 0 if joint j is revolute; σ_j = 1 if joint j is prismatic; $\bar{\sigma}_j = 1 - \sigma_j$

In our case, for both robots all the joints are revolute, so we have σ_j = 0, which give us:

$$q_j = \bar{\sigma}_j \cdot \theta_j = \theta_j$$

Notes should be taken into consideration:

- the frame R_0 is chosen to be aligned with the frame R_1 when $q_1 = 0$. This means that $\mathbf{z_0}$ is aligned with $\mathbf{z}_1$, whereas the origin $O_0 = O_1$ is coincident with the origin O_i if joint 1 is revolute,
- in a similar way, the choice of the $\mathbf{x_n}$ axis to be aligned with $\mathbf{x_{n-1}}$ when $q_n = 0$,
- if $\mathbf{z}_j$ is parallel to $\mathbf{z}_{j+1}$, we place $\mathbf{x}_j$ in such a way that $r_j = 0$ or $r_{j+1} = 0$;

The Homogeneous Transformation Matrix between two frames $j-1$ to j in case of RV-AJ and RV-2SD robot arms

RV-2AJ robot arm:

$$\boldsymbol{H}_1^0 = \begin{pmatrix} C\theta_1 & -S\theta_1 & 0 & 0 \\ S\theta_1 & C\theta_1 & 0 & 0 \\ 0 & 0 & 1 & r_{01} \\ 0 & 0 & 0 & 1 \end{pmatrix} \quad \boldsymbol{H}_2^1 = \begin{pmatrix} C\theta_2 & -S\theta_2 & 0 & 0 \\ 0 & 0 & -1 & 0 \\ S\theta_2 & C\theta_2 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix} \quad \boldsymbol{H}_3^2 = \begin{pmatrix} C\theta_3 & -S\theta_3 & 0 & d_{23} \\ S\theta_3 & C\theta_3 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}$$

$$\boldsymbol{H}_4^3 = \begin{pmatrix} C\theta_4 & -S\theta_4 & 0 & d_{34} \\ S\theta_4 & C\theta_4 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix} \quad \boldsymbol{H}_5^4 = \begin{pmatrix} C\theta_5 & -S\theta_5 & 0 & 0 \\ 0 & 0 & 1 & r_{45} \\ -S\theta_5 & -C\theta_5 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}$$

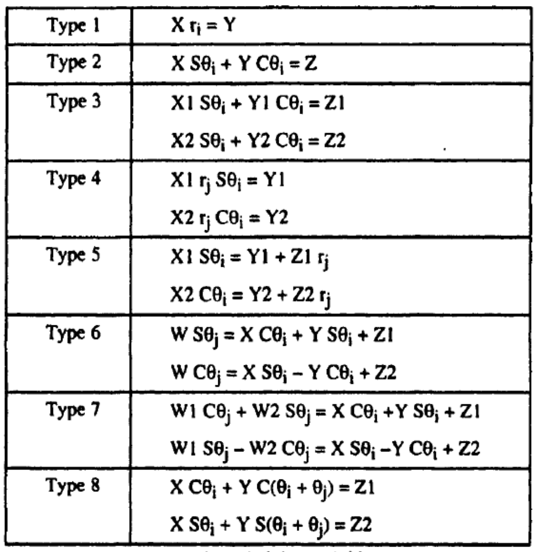
RV-2SD robot arm:

$$H_1^0 = \begin{pmatrix} C1 & -S1 & 0 & 0 \\ S1 & C1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix} \quad H_2^1 = \begin{pmatrix} C2 & -S2 & 0 & 0 \\ 0 & 0 & -1 & 0 \\ S2 & C2 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix} \quad H_3^2 = \begin{pmatrix} C3 & -S3 & 0 & D_3 \\ S3 & C3 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}$$

$$H_4^3 = \begin{pmatrix} C4 & -S4 & 0 & 0 \\ 0 & 0 & 1 & R_4 \\ -S4 & -C4 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix} \quad H_5^4 = \begin{pmatrix} C5 & -S5 & 0 & 0 \\ 0 & 0 & -1 & 0 \\ S5 & C5 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix} \quad H_6^5 = \begin{pmatrix} C6 & -S6 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ -S5 & -C6 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}$$

where C1 = $\mathrm{Cos}\,\theta_1$ and S1 = $\mathrm{Sin}\,\theta_1$ to simplify the equations.

Types of equations encountered in the Paul method [103]



Type 1	$X\, r_i = Y$
Type 2	$X\, S\theta_i + Y\, C\theta_i = Z$
Type 3	$X1\, S\theta_i + Y1\, C\theta_i = Z1$ $X2\, S\theta_i + Y2\, C\theta_i = Z2$
Type 4	$X1\, r_j\, S\theta_i = Y1$ $X2\, r_j\, C\theta_i = Y2$
Type 5	$X1\, S\theta_i = Y1 + Z1\, r_j$ $X2\, C\theta_i = Y2 + Z2\, r_j$
Type 6	$W\, S\theta_j = X\, C\theta_i + Y\, S\theta_i + Z1$ $W\, C\theta_j = X\, S\theta_i - Y\, C\theta_i + Z2$
Type 7	$W1\, C\theta_j + W2\, S\theta_j = X\, C\theta_i + Y\, S\theta_i + Z1$ $W1\, S\theta_j - W2\, C\theta_j = X\, S\theta_i - Y\, C\theta_i + Z2$
Type 8	$X\, C\theta_i + Y\, C(\theta_i + \theta_j) = Z1$ $X\, S\theta_i + Y\, S(\theta_i + \theta_j) = Z2$

r_j: prismatic joint variable,
$S\theta_i$, $C\theta_i$: sine and cosine of a revolute joint variable θ_i.

Figure A.13. Types of equations encountered in the Paul method

The possible solutions for joint angles of RV-2AJ and RV-2SD robot arms

$$\begin{tikzcd}
& \theta_1 \arrow[dl, "\varepsilon = +1"'] \arrow[dr, "\varepsilon = -1"] & & & \theta_1 + \pi \arrow[dl, "\varepsilon = +1"'] \arrow[dr, "\varepsilon = -1"] & \\
\theta_2 \arrow[d] & & \theta_2 \arrow[d] & \theta_2 \arrow[d] & & \theta_2 \arrow[d] \\
\theta_3 \arrow[d] & & \theta_3 \arrow[d] & \theta_3 \arrow[d] & & \theta_3 \arrow[d] \\
\theta_4 \arrow[d] & & \theta_4 \arrow[d] & \theta_4 \arrow[d] & & \theta_4 \arrow[d] \\
\theta_5 & & \theta_5 & \theta_5 & & \theta_5
\end{tikzcd}$$

Figure A.14. The possible solutions for joint angles of RV-2AJ robot arm

$$\begin{tikzcd}
 & \theta_1 \arrow[dl] \arrow[dr] & & & & \theta_1+\pi \arrow[dl] \arrow[dr] & & \\
\theta_2:\varepsilon=+1 \arrow[d] & & \theta_2:\varepsilon=-1 \arrow[d] & & \theta_2:\varepsilon=+1 \arrow[d] & & \theta_2:\varepsilon=-1 \arrow[d] & \\
\theta_3 \arrow[d] \arrow[dr] & & \theta_3 \arrow[d] \arrow[dr] & & \theta_3 \arrow[d] \arrow[dr] & & \theta_3 \arrow[d] \arrow[dr] & \\
\theta_4 \arrow[d] & \theta_4+\pi \arrow[d] & \theta_4 \arrow[d] & \theta_4+\pi \arrow[d] & \theta_4 \arrow[d] & \theta_4+\pi \arrow[d] & \theta_4 \arrow[d] & \theta_4+\pi \arrow[d] \\
\theta_5 \arrow[d] & \theta_5 \arrow[d] & \theta_5 \arrow[d] & \theta_5 \arrow[d] & \theta_5 \arrow[d] & \theta_5 \arrow[d] & \theta_5 \arrow[d] & \theta_5 \arrow[d] \\
\theta_6 & \theta_6 & \theta_6 & \theta_6 & \theta_6 & \theta_6 & \theta_6 & \theta_6
\end{tikzcd}$$

Figure A.15. The possible solutions for joint angles of RV-2SD robot arm

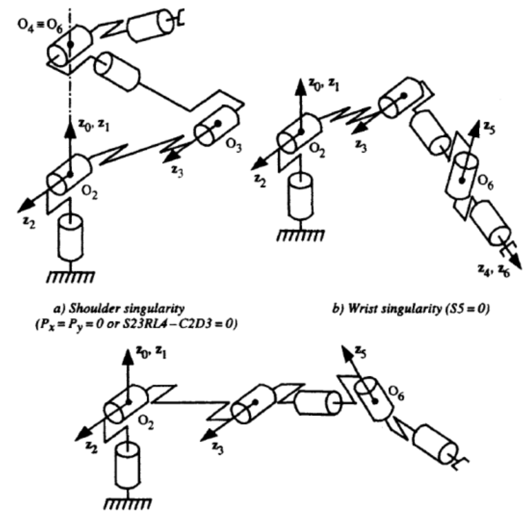


Figure A.16. Singular positions of RV-2SD robot

Homogeneous transformation matrix for RV-2AJ robot arm

$$H_5^1 = \begin{pmatrix} \text{-C5(C4(S2S3 - C2C3) + S4(C2S3 + C3S2))} & \text{S5(C4(S2S3 - C2C3) + S4(C2S3 + C3S2))} & \text{S4(S2S3 - C2C3) - C4(C2S3 + C3S2)} & \text{250C2 - 160S2S3 + 160C2C3} \\ S5 & C5 & 0 & 0 \\ \text{C5(C4(C2S3 + C3S2) - S4(S2S3 - C2C3))} & \text{-S5(C4(C2S3 + C3S2) - S4(S2S3 - C2C3))} & \text{- C4(S2S3 - C2C3) - S4(C2S3 + C3S2)} & \text{250S2 + 160C2S3 + 160C3S2} \\ 0 & 0 & 0 & 1 \end{pmatrix}$$

Methods for the calculation of Jacobian matrix of robot arm

- **Computation of the Jacobian matrix from the direct geometry model**

The Jacobian matrix can be obtained by differentiating the forward geometric model $X = f(q)$, using or the partial derivative $\frac{\partial X}{\partial q}$ as follow:

$$\boldsymbol{J}_{\boldsymbol{ij}} = \frac{\partial f_i(q)}{\partial q_j} \quad i = 1,\ldots,m \text{ and } j = 1,\ldots,n$$

where $\boldsymbol{J}_{\boldsymbol{ij}}$ is the (i, j) element of the $\mathbf{J}$ Jacobian matrix. This method is convenient for simple robots having a reduced number of degrees of freedom such as 2 or 3 degrees.

- **Kinematics Jacobian matrix « Basic »**

The Jacobian matrix can be obtained by a direct calculation method, based on the relation between the vectors of the velocity of translation V_n and rotation $\boldsymbol{\omega}_\mathbf{n}$ of the reference R_n, and the joint velocities $\dot{\mathbf{q}}$.

$$\mathbf{V_n} = \begin{pmatrix} V_n \\ \omega_n \end{pmatrix} = J_n \cdot \dot{q} \tag{1}$$

We note that $\mathbf{V_n}$ is the derivative of the position vector P_j concerning time, while $\boldsymbol{\omega}_\mathbf{n}$ is not the derivative of any orientation vector.

While computing the basic Jacobian matrix, we consider the k_{th} joint of an articulated chain, The velocity $\dot{q}_k$ of joint k produces the linear and angular velocities (V_n and $\boldsymbol{\omega}_\mathbf{n}$ respectively) at the terminal frame R_n, where V_n and $\boldsymbol{\omega}_\mathbf{n}$ in the case of the revolute joint $\sigma_k = 0$ as the same of our two robots, where the general form is written as follow:

$$\begin{cases} V_{k,n} = \left(\sigma_k \mathbf{a_k} + \bar{\sigma}_k (\mathbf{a_k} \times L_{k,n})\right) \dot{q}_k \\ \boldsymbol{\omega}_\mathbf{k,n} = \bar{\sigma}_k \mathbf{a_k} \dot{q}_k \end{cases} \tag{2}$$

Denote that $\mathbf{a_k}$ is the unit vector along the $\mathbf{z_k}$ axis and $L_{k,n}$ is the position vector connecting O_k to O_n. The linear and angular velocities of the terminal frame can be written as:

$$\begin{cases} V_n = \sum_{k=1}^{n} V_{k,n} = \sum_{k=1}^{n} \left(\sigma_k \mathbf{a_k} + \bar{\sigma}_k (\mathbf{a_k} \times L_{k,n})\right) \dot{q}_k \\ \boldsymbol{\omega}_n = \sum_{k=1}^{n} \omega_{k,n} = \sum_{k=1}^{n} \bar{\sigma}_k \mathbf{a_k} \dot{q}_k \end{cases} \tag{3}$$

Writing **eq 2** in matrix form, we deduce that:

$$J_n = \begin{bmatrix} \sigma_1 \mathbf{a}_1 + \bar{\sigma}_1 (\mathbf{a}_1 \times L_{1,n}) & \ldots. & \sigma_n \mathbf{a}_n + \bar{\sigma}_n (\mathbf{a}_n \times L_{n,n}) \\ \bar{\sigma}_1 \mathbf{a}_1 & \ldots. & \bar{\sigma}_n \mathbf{a}_n \end{bmatrix} \tag{4}$$

Referring to the vectors of *Jn* with respect to frame R_j, we obtain the $(6 \times n)$ Jacobian matrix ${}^iJ_n$ such that: ${}^i\mathbf{V}_n = {}^iJ_n \cdot \dot{q}$

In general, we calculate $\mathbf{V_n}$ and $\boldsymbol{\omega}_\mathbf{n}$ in-frame R_n or frame R_0. The corresponding Jacobian matrix is denoted by ${}^nJ_n$ or ${}^0J_n$ respectively. These matrices can also be computed using any matrix ${}^iJ_n$, for $i = 0,\ldots,n$ thanks to the following expression:

$${}^sJ_n = \begin{bmatrix} {}^\mathbf{s}\mathbf{A_i} & 0_3 \\ 0_3 & {}^\mathbf{s}\mathbf{A_i} \end{bmatrix} {}^iJ_n$$

Where ${}^{s}\mathbf{A_i}$ is the (3x3) orientation matrix of the frame R_s relative to the frame R_i . In general, we obtain the simplest matrix ${}^{i}J_n$ when i = integer (n/2). We note that the matrices ${}^{i}J_n$, for i=0..,n, have the same singular positions.

Computation of the matrix ${}^{i}J_n$

Since the vector product ${}^{k}L_{k,n} = {}^{k}P_n$, the k^{th} column of ${}^{i}J_n$ denoted as ${}^{i}J_{n,k}$ becomes:

$$ {}^{i}J_{n,k} = \begin{bmatrix} \bar{\sigma}_k(-{}^{k}P_{n_y}\,{}^{\mathbf{i}}\mathbf{s_k} + {}^{k}P_{n_x}\,{}^{\mathbf{i}}\mathbf{n_k}) \\ \bar{\sigma}_k\,{}^{\mathbf{i}}\mathbf{a_k} \end{bmatrix} \qquad \textbf{(5)} $$

Where P_{n_y} and P_{n_x} denote the x and y components of the vector $\mathbf{P_n}$ respectively.

The column ${}^{i}J_{n,k}$ is computed from the elements of the matrix H_n^k resulting from the DGM.

Jacobian matrix for RV-2AJ robot arm

$J(1,1) = -{}^{1}P_{5y}\,{}^{5}s_1 + {}^{1}P_{5x}\,{}^{5}n_1 = (250C2 - 160S2S3 + 160C2C3)S5$

$J(2,1) = -{}^{1}P_{5y}\,{}^{5}s_1 + {}^{1}P_{5x}\,{}^{5}n_1 = (250C2 - 160S2S3 + 160C2C3)C5$

$J(3,1) = -{}^{1}P_{5y}\,{}^{5}s_1 + {}^{1}P_{5x}\,{}^{5}n_1 = 0$

$J(4,1) = {}^{5}a_1 = C5(C4(C2S3 + C3S2) - S4(S2S3 - C2C3))$

$J(5,1) = {}^{5}a_1 = -S5(C4(C2S3 + C3S2) - S4(S2S3 - C2C3))$

$J(6,1) = {}^{5}a_1 = -C4(S2S3 - C2C3) - S4(C2S3 + C3S2)$

$J(1,2) = -{}^{2}P_{5y}\,{}^{5}s_2 + {}^{2}P_{5x}\,{}^{5}n_2 = C5(160S4 + 250C3S4 + 250C4S3)$

$J(2,2) = -{}^{2}P_{5y}\,{}^{5}s_2 + {}^{2}P_{5x}\,{}^{5}n_2 = -S5(160S4 + 250C3S4 + 250C4S3)$

$J(3,2) = -{}^{2}P_{5y}\,{}^{5}s_2 + {}^{2}P_{5x}\,{}^{5}n_2 = 160C4 + 250C3C4 - 250S4S3$

$J(4,2) = {}^{5}a_2 = -S5$

$J(5,2) = {}^{5}a_2 = -C5$

$J(6,2) = {}^{5}a_2 = 0$

$J(1,3) = 160C5S4$	$J(1,4) = 0$	$J(1,5) = 0$
$J(2,3) = -160S4S5$	$J(2,4) = 0$	$J(2,5) = 0$
$J(3,3) = 160C4$	$J(3,4) = 0$	$J(3,5) = 0$
$J(4,3) = -S5$	$J(4,4) = -S5$	$J(4,5) = 0$
$J(5,3) = -C5$	$J(5,4) = -C5$	$J(5,5) = 0$
$J(6,3) = 0$	$J(6,4) = 0$	$J(6,5) = 1$

Jacobian matrix for RV-2SD robot arm

J(1, 1) = (-C6C5S4 - S6C4)(S23R4 - C2D3)	J(1, 2) = (-C6C5C4 + S6C4)(R4 - S3D3) + C6S5C3D3
J(2, 1) = (-S6C5S4 - C6C4)(S23R4 - C2D3)	J(2, 2) = (S6C5C4 + C6S4)(R4 - S3D3) - S6S5C3D3
J(3, 1) = S5S4(S23R4 - C2D3)	J(3, 2) = S5C4(R4 - S3D3) + C5C3D3
J(4, 1) = (C6C5C4 - S6C4)S23 + C6S5C23	J(4, 2) = -C6C5S4 - S6C4
J(5, 1) = (-S6C5C4 - C6S4)S23 - S6S5C23	J(5, 2) = S6C5S4 - C6C4
J(6, 1) = -S5C5S23 + C5C23	J(6, 2) = S5S4

J(1, 3) = (-C6C5C4 + S6S4)R4	J(1, 4) = 0	J(1, 5) = 0	J(1, 6) = 0
J(2, 3) = (S6C5C4 + C6S4)R4	J(2, 4) = 0	J(2, 5) = 0	J(2, 6) = 0
J(3, 3) = S5C4R4	J(3, 4) = 0	J(3, 5) = 0	J(3, 6) = 0
J(4, 3) = -C6C5S4 - S6C4	J(4, 4) = C6S5	J(4, 5) = -S6	J(4, 6) = 0
J(5, 3) = S6C5S4 - C6C4	J(5, 4) = -S6S5	J(5, 5) = -C6	J(5, 6) = 0
J(6, 3) = S5S4	J(6, 4) = C5	J(6, 5) = 0	J(6, 6) = 1

Pseudo inverse script of Jacobian matrix for RV-2AJ robot arm

```
%% Jacobian matrix for RV-2AJ arm
J=      [250*C2*S5-160*S2*S3*S5+160*C2*C3*S5,C5*(160*S4+250*C3*S4+250*C4*S3),
160*C5*S4,0,0          ;          250*C2*C5-160*S2*S3*C5+160*C2*C3*C5,-
S5*(160*S4+250*C3*S4+250*C4*S3),    -160*S5*S4,    0,0;0,    160*C4+250*C3*C4-
250*S4*S3,160*C4,0,0  ;C5*C4*C2*S3+C5*C4*C3*S2-S4*S2*S3*C5+C5*C2*C3*S4,  -S5,-
S5,  -S5,0 ; -S5*(C4*C2*S3+C4*C3*S2-S4*S3*S2+C2*C3*S4), -C5, -C5,-C5,0 ;-C4*(S2*S3-
C2*C3)-S4*(C2*S3+C3*S2), 0,0,0,1]

J1=J(:,1)       %% Inisialize the first colomn of Jacobian matrix
%% Inisialize an empty cell for the dynamic values will be used in algorithm
Jp = cell(1,1)   %% The pseudo inverse of Jacobian matrix
d = cell(1, 1)   %% a value to calculate the pseudo inverse matrix
c = cell(1, 1)   %% a value to calculate the pseudo inverse matrix
b = cell(1, 1)   %% a value to calculate the pseudo inverse matrix
Jp{1}= inv(transpose(J1)*J1)*transpose(J1)        %% inisalize the first iteration of the pseudo
inverse matrix (where k=1)
for k=2:1:3
    d{k-1} = Jp{k-1}*J(:,k)
      if (k==3 | k==4 | k==5)
          c{k-1}=0
      else
    c{k-1} = J(:,k) - J(:,k-1)*d{k-1}
      end
   if (c{k-1} == 0 )
      b{k-1}= inv(1+transpose(d{k-1})*d{k-1})*transpose(d{k-1})*Jp{k-1}
   else
       b{k-1}= inv(transpose(c{k-1})*c{k-1})*transpose(c{k-1})
    end
 Jp{k}=[Jp{k-1} - d{k-1}*b{k-1}; b{k-1}]
end
```

Forward and Inverse dynamic models

The dynamic model of robots plays an important role in their design and operation, where we find two models: the inverse dynamic model provides the joint torques and forces in terms of the joint positions, velocities and accelerations. It is described by:

$$\Gamma = f\left(q,\ \dot{q},\ \ddot{q},\mathbf{Fe}\right) \qquad \textbf{(6)}$$

with:

Γ: vector of joint torques or forces, depending on whether the joint is revolute or prismatic respectively. In the sequel, we will only write joint torques;

q: vector of joint positions;

$\dot{q}$: vector of joint velocities;

$\ddot{q}$: vector of joint accelerations;

$\mathbf{Fe}$: vector of forces and moments exerted by the robot on the environment.

eq 6 is an inverse dynamic model because it defines the system input as a function of the output variables. It is often called the dynamic model. The direct dynamic model describes the joint accelerations in terms of the joint positions, velocities and torques. It is represented by the relation:

$$\ddot{q} = f\left(q,\ \dot{q},\Gamma,\mathbf{Fe}\right) \qquad \textbf{(7)}$$

For robot design, the inverse dynamic model can be used to select the actuators, while the direct dynamic model is employed to carry out simulations for the purpose of testing the performance of the robot and to study the relative merits of possible control schemes. Regarding robot operations, the inverse dynamic model is used to compute the actuator torques, which are needed to achieve a desired motion. It is also used to identify the dynamic parameters that are necessary for both control and simulation applications.

There are several formalisms to obtain the dynamic model of robots. The most often used formalisms are:

- Lagrange formalism
- Newton-Euler formalism

In our case we will use both formalism; a part we use the Lagrange formalism to bring out our matrices of inertia, Coriolis and gravity for the order because it is the simplest method given these goals. In addition, we will use the Newton-Euler formalism to calculate torque joints because Lagrange's method is not the one that gives the best performing model in terms of number of transactions.

1) The Lagrange formalism

The Lagrange formulation describes the behavior of a dynamic system in terms of work and energy stored in the system. The Lagrange equations are commonly written in the form:

$$\Gamma_i = \frac{d}{dt}\frac{\partial L}{\partial \dot{q}_i} - \frac{\partial L}{\partial q_i} \qquad \textbf{(8)}$$

Where:

- L: Lagrangian of the system equal to E - U;
- E: total kinetic energy of the system;
- U: total potential energy of the system;

From **eq 8** the couple Γ can be put in the form [58]:

$$\Gamma = A(q)\ddot{q} + C(q,\dot{q})\dot{q} + Q(q) \qquad \textbf{(9)}$$

Where:

- $\mathbf{A}$ is the $(n \times n)$ symmetric and positive definite inertia matrix of the robot. Its elements are functions of the joint positions. The (*i, j*) element of $\mathbf{A}$ is denoted by $\mathbf{A}_{i,j}$.
- $C\left(q,\dot{q}\right)\dot{q}$ is the $(n \times 1)$ vector of Coriolis and centrifugal torques, such that:

$$C\dot{q} = \dot{A}\dot{q} - \frac{\partial E}{\partial q} \qquad \textbf{(10)}$$

- $Q = \left[Q_1....Q_n\right]^T$ is the vector of gravity torques.

There exist several forms for the vector *C*. Using the *Christoffell symbols* $c_{i,jk}$ element of the matrix *C* can be written as:

$$C_{ij} = \sum_{k=1}^{n} c_{i,jk}.\dot{q}_k \qquad \textbf{(11)}$$

$$c_{i,jk} = \frac{1}{2}\left[\frac{\partial A_{ij}}{\partial q_k} + \frac{\partial A_{ik}}{\partial q_j} + \frac{\partial A_{jk}}{\partial q_i}\right]$$

The elements of the vector *Q* is calculated according to:

$$Q_i = \frac{\partial U}{\partial q_i} \qquad \textbf{(12)}$$

The elements of *A*, *C* and *Q* are functions of the geometric and inertial parameters of the robot. Consequently, the dynamic model of a robot is described by *n* coupled and nonlinear second order differential equations.

➢ **Computation of the kinetic energy**

The kinetic energy of the robot is given as:

$$E = \sum_{k=1}^{n} E_j \qquad \textbf{(13)}$$

where E_j denotes the kinetic energy of link j, which can be computed by:

$$E_j = \frac{1}{2}(\omega_j^T.I_{Gj}.\omega_j + M_j.V_{Gj}^T.V_{Gj}) \qquad \textbf{(14)}$$

Since the velocity of the center-of-mass can be expressed as follow:

$$V_{Gj} = V_j + \omega_j \times L_j \qquad \textbf{(15)}$$

and since we have:

$$J_j = I_{Gj} - M_j.\hat{L}_j.\hat{L}_j \qquad \textbf{(16)}$$

The equation becomes:

$$E_j = \left[\frac{1}{2}\omega_j^T.J_j.\omega_j + M_j.V_j^T.V_j + 2.ML_j^T(V_j \times \omega_j)\right] \qquad \textbf{(17)}$$

Eq 14 is not linear in the coordinates of the vector L_j. On the contrary, **eq 17** is linear in the elements of M_j, ML_j, and J_j, which we call the standard inertial parameters. The linear and angular velocities V_j, ω_j are computed using the following recursive equations:

$$\begin{aligned} \omega_j &= \omega_{j-1} + \dot{q}_j \cdot a_j \\ V_j &= V_{j-1} + \omega_{j-1} \times L_j \end{aligned} \qquad \textbf{(18)}$$

If the base of the robot is fixed, the previous equations are initialized by $V_0 = 0$ and $\omega_0 = 0$, All the elements appearing in **eq 17** must be expressed in the same frame. The most efficient way is to express them relative to frame R_j. Therefore, **eq 17** and **eq 18** are rewritten as:

$$E_j = \frac{1}{2}\left[{}^j\omega_j^T.{}^jJ_j.{}^j\omega_j + M_j.{}^jV_j^T.{}^jV_j + 2.{}^jML_j^T({}^jV_j \times {}^j\omega_j)\right] \qquad \textbf{(19)}$$

$$\begin{aligned} {}^j\omega_j &= {}^jA_j.{}^{j-1}\omega_{j-1} + \dot{q}_j.{}^ja_j \\ {}^jV_j &= {}^jA_j({}^{j-1}V_{j-1} + {}^{j-1}\omega_{j-1} \times {}^{j-1}P_j) \end{aligned} \qquad \textbf{(20)}$$

➢ **Computation of the potential energy**

The potential energy is given by:

$$U = \sum_{j=1}^{n} U_j = \sum_{j=1}^{n} -M_j.g^T(L_{0,j} + L_j) \qquad \textbf{(21)}$$

Where $L_{0,j}$ is the position vector from the origin O_0 to O_j. Projecting the vectors appearing in $U = \sum_{j=1}^{n} U_j = \sum_{j=1}^{n} -M_j.g^T(L_{0,j} + L_j)$ (**eq 21** into the frame R_0, we obtain:

$$U_j = -M_j.{}^0g^T({}^0P_j + {}^0A_j\,{}^jL_j) \qquad \textbf{(22)}$$

An expression that can be rewritten linearly in M_j and the elements of ${}^jML_j$ as:

$$U_j = -{}^0g^T(M_j.{}^0P_j + {}^0A_j.{}^jML_j) = (-{}^0g^T \quad 0){}^0H_j \begin{pmatrix} {}^jML_j \\ M_j \end{pmatrix} \qquad \textbf{(23)}$$

Since the kinetic and potential energies are linear in the elements of ${}^jJ_j$, ${}^jML_j$, M_j, we deduce that the dynamic model is also linear in these parameters.

The dynamic model explained in this part is considered pure without additive forces as friction force – static force - the rotor inertia of actuators.

2) Newton-Euler formalism

The Newton-Euler equations describing the forces and moments (wrench) acting on the center-of-mass of link *j* are given as:

$$\begin{aligned} F_j &= M_j.\dot{V}_{Gj} \\ \mathbf{M}_{Gj} &= I_{Gj}\dot{\omega}_j + \omega_j \times (I_{Gj}\omega_j) \end{aligned} \qquad \textbf{(24)}$$

where:

- F_j : the resultant of the external forces on the link ***j*** ;
- M_j : the mass of the link ***j*** ;
- $\dot{V}_{Gj}$: the acceleration of the center of gravity of the link ***j*** ;
- $\mathbf{M}_{Gj}$: the moment of the external forces exerted on the link ***j*** around G_j ;
- I_{Gj} : the inertia tensor of the link ***j*** with respect to a coordinate system parallel to R_j and of G_j origin;
- $\dot{\omega}_j$: the rotational acceleration of the link ***j*** ;
- ω_j : the angular velocity of the link ***j*** ;

The Newton-Euler algorithm of Luh, Walker and Paul, which is considered as one of the most efficient algorithms for real time computation of the inverse dynamic model, consists of two recursive computations: forward recursion and backward recursion. The forward recursion, from the base to the terminal link, computes the link velocities and accelerations and consequently the dynamic wrench on each link. The backward recursion, from the terminal link to the base, provides the reaction wrenches on the links and consequently the joint torques.

➢ **Newton-Euler inverse dynamics linear in the inertial parameters**

In this section, we develop a Newton-Euler algorithm based on the double recursive computations, but which uses as inertial parameters the elements of J_j, M_j and ML_j. The dynamic wrench on link ***j*** is calculated on O_j and not on the center of gravity G_j. Therefore, the resulting model is linear in the dynamic parameters. This reformulation allows us to compute the dynamic model in terms of the base inertial parameters and to use it for the identification of the dynamic parameters. The Newton-Euler equations giving the forces and moments of link $\boldsymbol{j}$ at the origin of the frame R_j are given as:

$$\begin{aligned} F_j &= M_j.\dot{V}_{Gj} + \dot{\omega}_j \times ML_j + \omega_j \times (\omega_j \times ML_j) \\ \mathbf{M}_j &= J_j\dot{\omega}_j + \omega_j \times (J_j\omega_j) + ML_j \times \dot{V}_j \end{aligned} \quad \textbf{(25)}$$

where:

- $\mathbf{M}_j$: the moment of the external forces exerted on the link j around O_j;
- $\dot{V}_j$: the acceleration of the point O_j;

Forward recursive computation: to compute F_j and $\mathbf{M}_j$, for $j = 1,...,n$, using **eq 25**, we need V_j, ω_j of R_n. The velocities are given by the recursive equations rewritten hereafter as:

$$\begin{cases} \omega_j = \omega_{j-1} + \dot{q}_j.\mathbf{a}_j \\ V_j = V_{j-1} + \omega_{j-1} \times L_j \end{cases} \quad \textbf{(26)}$$

Differentiating **eq 26** with respect to time gives:

$$\begin{cases} \dot{\omega}_j = \dot{\omega}_{j-1} + (\ddot{q}_j.\mathbf{a}_j + \omega_{j-1} \times \dot{q}.\mathbf{a}_j) \\ \dot{V}_j = \dot{V}_{j-1} + \dot{\omega}_{j-1} \times L_j + \omega_{j-1} \times (\omega_{j-1} \times L_j) \end{cases} \quad \textbf{(27)}$$

where, L_j: the vector linking the origin of the reference R_i, the antecedent of the reference R_j, and the origin of the reference R_j. it is equal to O_iO_j;

The initial conditions for a robot with a fixed base are $\dot{\omega}_j = 0, \omega_j = 0, \dot{V}_j = 0$;

Backward recursive computation: this is based on writing for each link $\boldsymbol{j}$, for $j = 1,...,n$, the Newton-Euler equations at the origin O_j, as follows (Figure 9.5):

$$\begin{cases} F_j = f_j - f_{j+1} \times M_j.g - f_{ej} \\ \mathbf{M}_j = m_j - m_{j+1} - L_{j+1} \times f_{j+1} + L_{Cj} \times M_j.g - m_{ej} \end{cases} \quad \textbf{(28)}$$

where:

f_j: the resultant of the dynamic torsor exerted on the link $\boldsymbol{j}$ by its antecedent and by the actuator $\boldsymbol{j}$;

m_j: the moment of the dynamic torsor exerted on the link $\boldsymbol{j}$ by its antecedent and by the actuator around O_j;

f_{ej}: the result of the dynamic torsor exerted by the link $\boldsymbol{j}$ on the environment ;

m_{ej}: the moment of dynamic torsor exerted by the link $\boldsymbol{j}$ on the environment around O_j;

g : the acceleration of gravity;

L_{Cj} : vector of the center of mass coordinates of link $\boldsymbol{j}$. It is equal to $\mathbf{O_j G_j}$;

We can cancel the gravity terms and take into account their effects by setting up the initial linear acceleration such that:

$$\dot{V}_0 = -g$$

Thus, we obtain:

$$\begin{cases} f_j = F_j + f_{j+1} + f_{ej} \\ m_j = \mathbf{M}_j + m_{j+1} + L_{j+1} \times f_{j+1} + m_{ej} \end{cases} \quad \textbf{(29)}$$

This backward recursive algorithm is initialized by $f_{j+1} = 0$ and $m_{j+1} = 0$.

Finally, the joint torque Γ_j can be obtained by projecting f_j or m_j on the joint axis, depending on whether the joint is prismatic or revolute respectively. We can also consider the friction forces and the rotor inertia as shown in the Lagrange method:

$$\Gamma_j = m_j^T.k_j + F_{sj}Sign(\dot{q}_h) + F_{vj}\dot{q}_j + I_{aj}\ddot{q}_j \quad \textbf{(30)}$$

where:

F_{sj} : dry friction parameter of joint $\boldsymbol{j}$;

F_{vj} : viscous friction parameter of joint $\boldsymbol{j}$;

We deduce directly that the terms f_j and m_j depend only on the parameters inertial of body $\boldsymbol{j}$ and those of bodies located downstream which are introduced by the terms f_{j+1} and m_{j+1} of the recurrence.

➢ **Practical form of the Newton Euler algorithm**

The Newton-Euler algorithm can be efficiently computed by referring the velocities, accelerations, forces, and moments to the local link coordinate system. The forward recursive equations become, for $j = 1,...,n$:

$$\begin{aligned} {}^j\omega_{j-1} &= {}^jA_{j-1}.{}^{j-1}\omega_{j-1} \\ {}^j\omega_j &= {}^j\omega_{j-1} + \dot{q}_j.{}^j\mathbf{a}_j \\ {}^j\dot{\omega}_j &= {}^jA_{j-1}.{}^{j-1}\dot{\omega}_{j-1} + \ddot{q}_j.{}^j\mathbf{a}_j + {}^j\omega_{j-1} \times \dot{q}_j.{}^j\mathbf{a}_j \\ {}^j\dot{V}_j &= {}^jA_{j-1}({}^{j-1}\dot{V}_{j-1} + {}^{j-1}U_{j-1}.{}^{j-1}P_j) \\ {}^jF_j &= M_j\,{}^j\dot{V}_j + {}^jU_j.{}^jML_j \\ {}^j\mathbf{M}_j &= {}^jJ_j.{}^j\dot{\omega}_j + {}^j\omega_j \times ({}^jJ_j.{}^j\omega_j) + {}^jML_j \times {}^j\dot{V}_j \end{aligned} \quad \textbf{(31)}$$

Where: $\dot{V}_0 = -g$, $\omega_0 = 0$, $\dot{\omega}_0 = 0$, ${}^jU_j = {}^j\hat{\dot{\omega}}_j + {}^j\hat{\omega}_j.{}^j\hat{\omega}_j$

For the backward recurrence, when : $j = 6,....,1$

$$\begin{aligned} {}^{j}f_{j} &= {}^{j}F_{j} + {}^{j}f_{j+1} + {}^{j}f_{ej} \\ {}^{j-1}f_{j} &= {}^{j-1}A_{j}.{}^{j}f_{j} \\ {}^{j}m_{j} &= {}^{j}\mathbf{M}_{j} + {}^{j}A_{j+1}{}^{j+1}m_{j+1} + {}^{j}P_{j+1} \times {}^{j}f_{j+1} \times + {}^{j}m_{ej} \\ \Gamma_{j} &= {}^{j}m_{j}^{T}.{}^{j}\mathbf{a}_{j} + F_{sj}Sign(\dot{q}_{h}) + F_{vj}\dot{q}_{j} + I_{aj}\ddot{q}_{j} \end{aligned} \quad \textbf{(32)}$$

The previous algorithm can be calculated numerically. However, to decrease significantly the number of operations, it is preferable to highlight uses an iterative symbolic calculation technique. That's why in our case we are going to use a toolbox in MATLAB called rvctools (Robotcs, Vision and Control) created by Peter Corke. The latter has a "SerialLink.rne" function which will allow us to calculate our model with great ease. However, we can't do anything with this toolbox without calculating the parameters inertial (J_j, G_j, $\mathbf{M}_j$, ML_j). This last us will obtain our parameters directly thanks to SolidWorks software.

Direct dynamic model

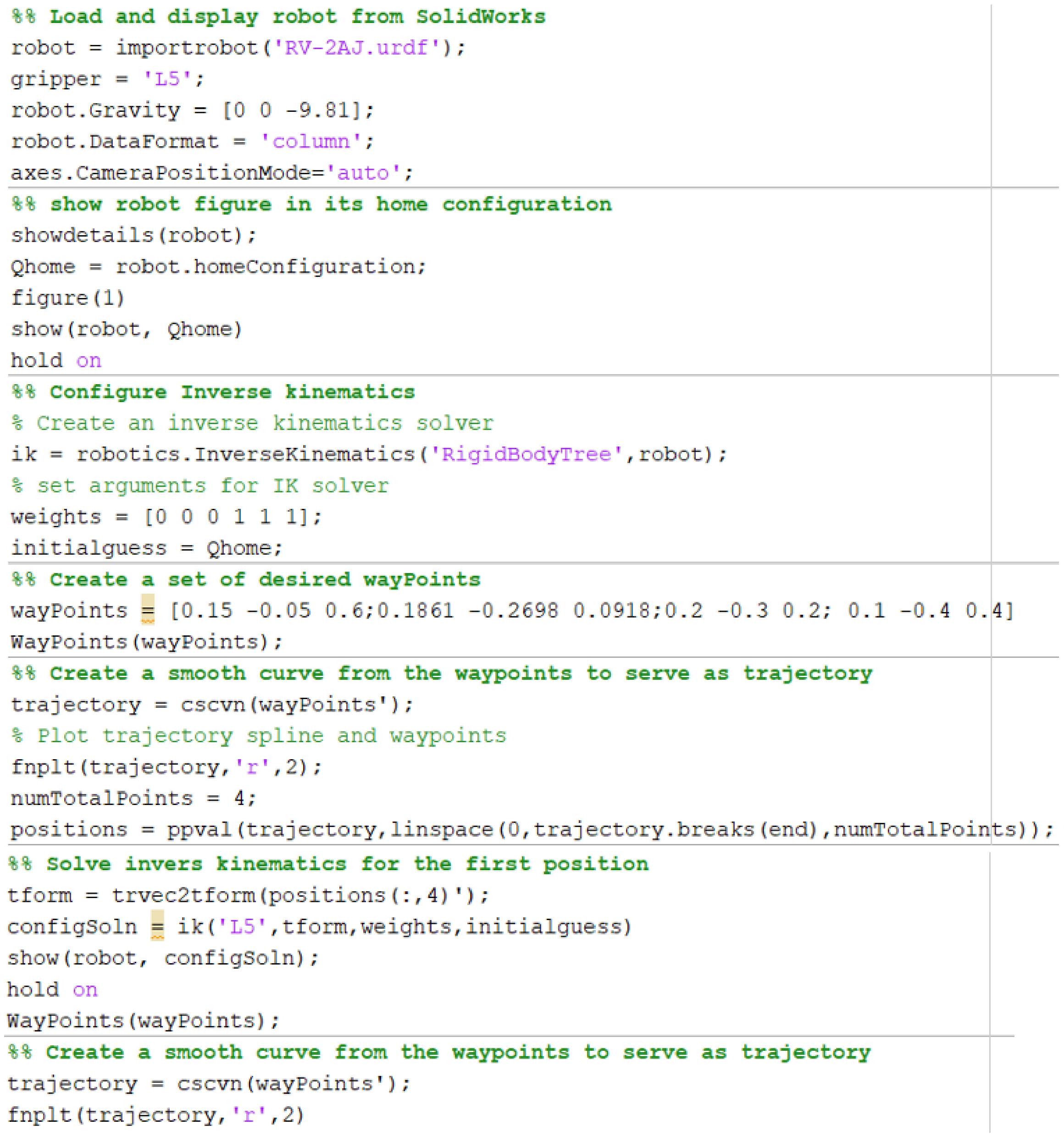
To simulate the behavior of the robot and its control loop, we use the direct dynamic model. Even if the computation time constraints are fewer reviews than for controlling, it is however interesting to have an efficient model. We present here the simplest method which proceeds by inversion of matrix $\boldsymbol{A}$. From the equation of the inverse dynamic model and assuming the existence of an external force on the terminal organ, the direct dynamic model is written:

$$\begin{aligned} \ddot{q} &= A^{-1}(\Gamma - H(q,\dot{q})) \\ H(q,\dot{q}) &= C(q,\dot{q}) + Q + J^{T}.\mathbf{fe} \end{aligned} \quad \textbf{(33)}$$

This form simulates well to digital integration. To achieve such simulation, we must therefore calculate the vector $H(q,\dot{q})$ and the matrix A.

Appendix 5: Script codes for motion control of the robot arm

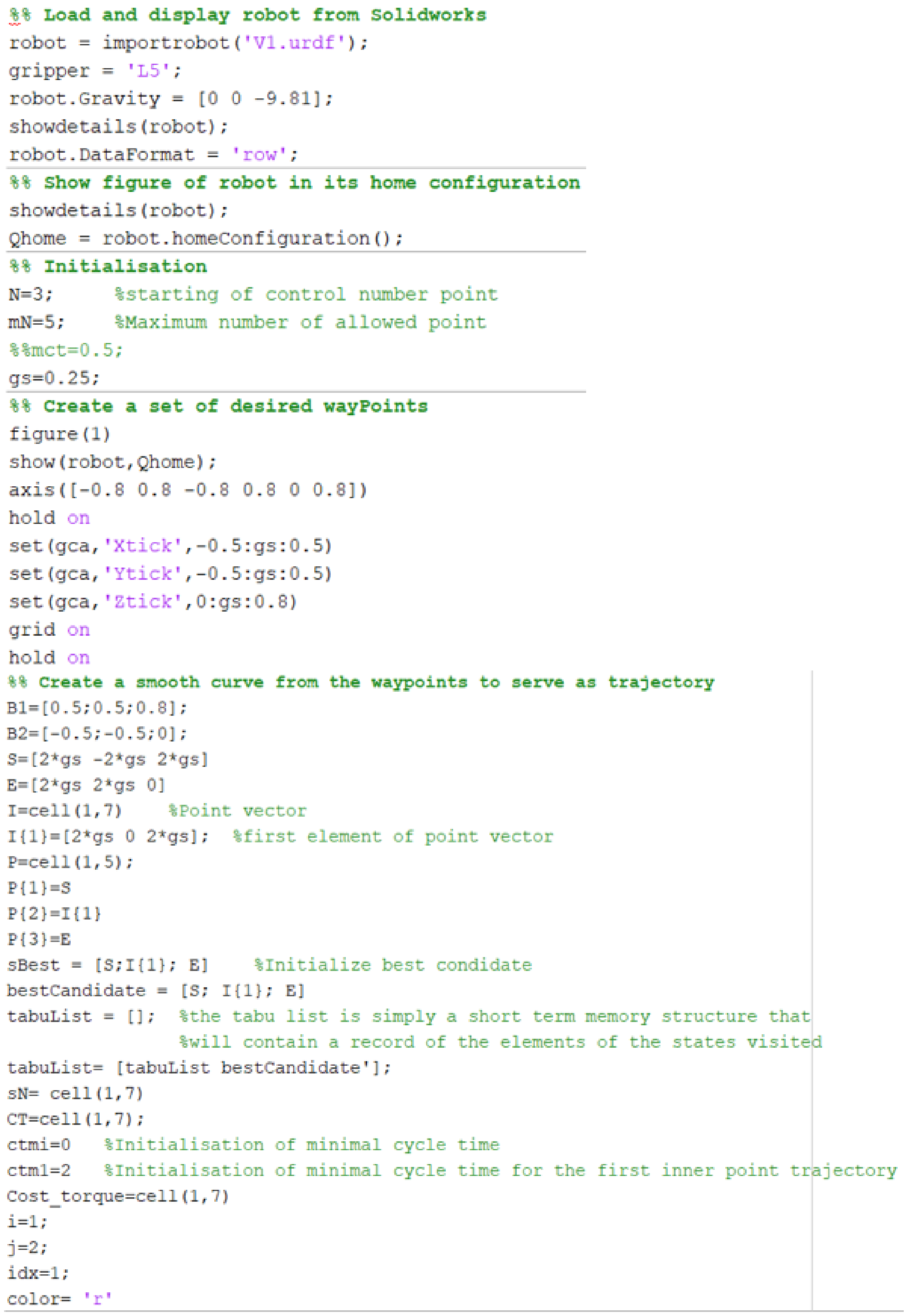
Script code for Forward and Invers kinematics configuration for RV-2AJ robot arm



```
%% Load and display robot from SolidWorks
robot = importrobot('RV-2AJ.urdf');
gripper = 'L5';
robot.Gravity = [0 0 -9.81];
robot.DataFormat = 'column';
axes.CameraPositionMode='auto';
%% show robot figure in its home configuration
showdetails(robot);
Qhome = robot.homeConfiguration;
figure(1)
show(robot, Qhome)
hold on
%% Configure Inverse kinematics
% Create an inverse kinematics solver
ik = robotics.InverseKinematics('RigidBodyTree',robot);
% set arguments for IK solver
weights = [0 0 0 1 1 1];
initialguess = Qhome;
%% Create a set of desired wayPoints
wayPoints = [0.15 -0.05 0.6;0.1861 -0.2698 0.0918;0.2 -0.3 0.2; 0.1 -0.4 0.4]
WayPoints(wayPoints);
%% Create a smooth curve from the waypoints to serve as trajectory
trajectory = cscvn(wayPoints');
% Plot trajectory spline and waypoints
fnplt(trajectory,'r',2);
numTotalPoints = 4;
positions = ppval(trajectory,linspace(0,trajectory.breaks(end),numTotalPoints));
%% Solve invers kinematics for the first position
tform = trvec2tform(positions(:,4)');
configSoln = ik('L5',tform,weights,initialguess)
show(robot, configSoln);
hold on
WayPoints(wayPoints);
%% Create a smooth curve from the waypoints to serve as trajectory
trajectory = cscvn(wayPoints');
fnplt(trajectory,'r',2)
```

Figure A.17. Script code for Forward and Inverse kinematics of RV-2AJ robot in MATLAB

Hybrid algorithm for optimization of robot arms' trajectory



```
%% Load and display robot from Solidworks
robot = importrobot('V1.urdf');
gripper = 'L5';
robot.Gravity = [0 0 -9.81];
showdetails(robot);
robot.DataFormat = 'row';
%% Show figure of robot in its home configuration
showdetails(robot);
Qhome = robot.homeConfiguration();
%% Initialisation
N=3;     %starting of control number point
mN=5;    %Maximum number of allowed point
%%mct=0.5;
gs=0.25;
%% Create a set of desired wayPoints
figure(1)
show(robot,Qhome);
axis([-0.8 0.8 -0.8 0.8 0 0.8])
hold on
set(gca,'Xtick',-0.5:gs:0.5)
set(gca,'Ytick',-0.5:gs:0.5)
set(gca,'Ztick',0:gs:0.8)
grid on
hold on
%% Create a smooth curve from the waypoints to serve as trajectory
B1=[0.5;0.5;0.8];
B2=[-0.5;-0.5;0];
S=[2*gs -2*gs 2*gs]
E=[2*gs 2*gs 0]
I=cell(1,7)     %Point vector
I{1}=[2*gs 0 2*gs];  %first element of point vector
P=cell(1,5);
P{1}=S
P{2}=I{1}
P{3}=E
sBest = [S;I{1}; E]     %Initialize best condidate
bestCandidate = [S; I{1}; E]
tabuList = [];  %the tabu list is simply a short term memory structure that
                %will contain a record of the elements of the states visited
tabuList= [tabuList bestCandidate'];
sN= cell(1,7)
CT=cell(1,7);
ctmi=0   %Initialisation of minimal cycle time
ctm1=2   %Initialisation of minimal cycle time for the first inner point trajectory
Cost_torque=cell(1,7)
i=1;
j=2;
idx=1;
color= 'r'
```

Figure A.18. Inputs arguments of hybrid algorithm for optimization of robot arm's trajectory