ISTANBUL TECHNICAL UNIVERSITY ★ GRADUATE SCHOOL OF SCIENCE ENGINEERING AND TECHNOLOGY

TRAJECTORY GENERATION FOR INDUSTRIAL ROBOTS IN PRESENCE OF WRIST SINGULARITY

M.Sc. THESIS

Cansın AKARSU

Department of Mechatronics Engineering

Mechatronics Engineering Programme

JANUARY 2020

ISTANBUL TECHNICAL UNIVERSITY ★ GRADUATE SCHOOL OF SCIENCE ENGINEERING AND TECHNOLOGY

TRAJECTORY GENERATION FOR INDUSTRIAL ROBOTS IN PRESENCE OF WRIST SINGULARITY

M.Sc. THESIS

Cansın AKARSU
(518161008)

Department of Mechatronics Engineering

Mechatronics Engineering Programme

Thesis Advisor: Assoc. Prof. Dr. Zeki Yağız BAYRAKTAROĞLU

JANUARY 2020

ISTANBUL TEKNİK ÜNİVERSİTESİ ★ FEN BİLİMLERİ ENSTİTÜSÜ

ENDÜSTRİYEL ROBOTLARDA BİLEK TEKİLLİĞİ VE YÖRÜNGE PLANLAMASI

YÜKSEK LİSANS TEZİ

Cansın AKARSU
(518161008)

Mekatronik Mühendisliği Anabilim Dalı

Mekatronik Mühendisliği Programı

Tez Danışmanı: Doç. Dr. Zeki Yağız BAYRAKTAROĞLU

OCAK 2020

Cansın Akarsu, a M.Sc. student of ITU Graduate School of Science Engineering and Technology student ID 518161008, successfully defended the thesis/dissertation entitled "TRAJECTORY GENERATION FOR INDUSTRIAL ROBOTS IN PRESENCE OF WRIST SINGULARITY", which he prepared after fulfilling the requirements specified in the associated legislations, before the jury whose signatures are below.

Thesis Advisor :	**Assoc. Prof. Dr. Zeki Yağız BAYRAKTAROĞLU** İstanbul Technical University	.....
Jury Members :	**Prof. Dr. Ata Muğan** İstanbul Technical University	.....
	Assoc. Prof. Dr. Cüneyt Yılmaz Yıldız Technical University	.....

Date of Submission : 5 December 2019
Date of Defense : 2 January 2020

In memory of my mother,

FOREWORD

I would like to thank my thesis advisor, Assoc. Prof. Dr. Zeki Yağız Bayraktaroğlu, for his guidance and encouragement, to accomplish this thesis.

I am also indebted to Altınay Robot Technoliges Inc. and my colleagues there, for introducing industrial robotics, and sharing their knowledge with me.

Last but not least, I would like thank to my family for supporting me to complete this study.

January 2020 Cansın AKARSU

TABLE OF CONTENTS

Page

ABBREVIATIONS

CNC	**:** Computer Numeric Control
DOF	**:** Degrees of Freedom
EE	**:** End-Effector
FGM	**:** Forward Geometrical Model
IFR	**:** International Federation of Robotics
IGM	**:** Inverse Geometrical Model
OEM	**:** Original Equipment Manufacturer
SVD	**:** Singular Value Decomposition
TCP	**:** Tool Center Point
TWA	**:** Twist Decomposition Approach
WCP	**:** Wrist Center Point

SYMBOLS

H	**:** Step count
L	**:** Length of the path
m	**:** Task space dimension
n	**:** Joint space dimension
o	**:** Operational space dimension
q_i	**:** Displacement of i'th joint
$\dot{q}_i$	**:** Velocity of i'th joint
$\ddot{q}_i$	**:** Acceleration of i'th joint
r_I	**:** Intrinsic redundancy
r_F	**:** Functional redundancy
r_K	**:** Kinematic redundancy
t	**:** Time
T	**:** Travel time of the path
Δt	**:** Robot sample time
$\Delta\theta$	**:** Angle increment
v_0	**:** Process speed
θ	**:** Angular displacement
$\dot{\theta}_0$	**:** Angular process speed
η_{acc}	: Acceleration ratio
$\eta_{accUsage}$	: Acceleration usage ratio
η_{vel}	: Velocity ratio
η_{rdn}	: Redundant axis ratio
$\mathfrak{J}$	**:** Joint space
$\mathfrak{O}$	**:** Operational space
$\mathfrak{T}$	**:** Task space
q_0	**:** Scalar quantity of quaternion
$\boldsymbol{d}$	**:** Direction vector of the path
$\boldsymbol{e}$	**:** Rotation axis vector

$\boldsymbol{e}_{r_i}$: Rotated axis vectors of i'th joint

$\boldsymbol{e}_{inf}$: Infeasible rotation axis vector

$\boldsymbol{o}$: Position vector between joint axes

$\boldsymbol{p}$: Position vector

$\boldsymbol{p}_i$: Position vector of i'th robot joint center

$\boldsymbol{p}_{WCP}$: Position vector of WCP

$\boldsymbol{p}_{tr}$: Position vector between WCP and TCP

$\boldsymbol{p}_{TCP}$: Position vector of TCP

$\boldsymbol{q}$: Joint displacements vector

$\dot{\boldsymbol{q}}$: Joint velocities vector

$\dot{\boldsymbol{q}}_a$: Augmented joint velocities vector

$\dot{\boldsymbol{q}}_{sh}$: Shoulder joints velocities vector

$\dot{\boldsymbol{q}}_{wr}$: Wrist joints velocities vector

$\ddot{\boldsymbol{q}}$: Joint accelerations vector

$\ddot{\boldsymbol{q}}_{sh}$: Shoulder joints accelerations vector

$\ddot{\boldsymbol{q}}_{\boldsymbol{wr}}$: Wrist joints accelerations vector

$\boldsymbol{r}_i$: Position distance vector of i'th joint

$\boldsymbol{s}$: Pose vector

$\boldsymbol{t}$: Twist vector

$\boldsymbol{t}_{WCP}$: Twist vector of WCP

$\boldsymbol{t}_{TCP}$: Twist vector of TCP

$\dot{\boldsymbol{t}}$: Twist rate vector

$\dot{\boldsymbol{t}}_{WCP}$: Twist rate vector of WCP

$\dot{\boldsymbol{t}}_{TCP}$: Twist rate vector of TCP

$\boldsymbol{x}$, **y**, **z** : Unit vectors of Cartesian coordinate system

$\Delta\boldsymbol{p}$: Position increment vector of the path

$\boldsymbol{v}$: Translational velocity vector

$\boldsymbol{v}_{WCP}$: Translational velocity vector of WCP

$\boldsymbol{v}_{TCP}$: Translational velocity vector of TCP

$\dot{\boldsymbol{v}}$: Translational acceleration vector

$\dot{\boldsymbol{v}}_{WCP}$: Translational acceleration vector of WCP

$\dot{\boldsymbol{v}}_{TCP}$: Translational acceleration vector of TCP

$\boldsymbol{\omega}$: Angular velocity vector

$\boldsymbol{\omega}_{comp}$: Compensation vector for angular velocity

$\boldsymbol{\omega}_{sh}$: Angular velocity vector of shoulder joints

$\boldsymbol{\omega}_{wr}$: Angular velocity vector of wrist joints

$[\boldsymbol{\omega}_{wr}]_{inf}$: Infeasible angular velocity vector of wrist joints

$[\boldsymbol{\omega}_{wr}]_{fsb}$: Feasible angular velocity vector of wrist joints

$\boldsymbol{\omega}_{EE}$: Angular velocity vector of End-Effector

$\boldsymbol{\omega}_{TCP}$: Angular velocity vector of TCP

$\dot{\boldsymbol{\omega}}$: Angular acceleration vector

$\dot{\boldsymbol{\omega}}_{comp}$: Compensation vector for angular acceleration

$\dot{\boldsymbol{\omega}}_{sh}$: Angular acceleration vector of shoulder joints

$\dot{\boldsymbol{\omega}}_{wr}$: Angular acceleration vector of wrist joints

$[\dot{\boldsymbol{\omega}}_{wr}]_{inf}$: Infeasible angular acceleration vector of wrist joints

$[\dot{\boldsymbol{\omega}}_{wr}]_{fsb}$: Infeasible angular acceleration vector of wrist joints

$\dot{\boldsymbol{\omega}}_{EE}$: Angular acceleration vector of End-Effector

$\dot{\boldsymbol{\omega}}_{TCP}$: Angular acceleration vector of TCP

$\boldsymbol{x}$: Cartesian space pose vector

$\boldsymbol{q}$: Vector quantity of quaternion

$\overline{\boldsymbol{q}}$: Quaternion

$\overline{\boldsymbol{q}}_{TCP}$: Quaternion of TCP

$\bar{\boldsymbol{\lambda}}$: Linear invariants of rotation

$\boldsymbol{E}$: Cross-Product matrix of rotation axis vector

$\boldsymbol{I}$: Identity matrix

$\boldsymbol{\mathcal{J}}$: Jacobian matrix

$\boldsymbol{\mathcal{J}_a}$: Augmented Jacobian matrix

$\boldsymbol{\mathcal{J}}^{\dagger}$: Pseudo-Inverse of Jacobian

$\dot{\boldsymbol{\mathcal{J}}}$: Jacobian rate matrix

$\boldsymbol{M}_{comp}$: Compensation matrix

$\boldsymbol{P}_{inf}$: Infeasible projection matrix

$\boldsymbol{P}_{fsb}$: Feasible proejction matrix

$\boldsymbol{P}_{tr}$: Cross-product matrix of $\boldsymbol{p}_{tr}$

$\boldsymbol{P}_{L}$: Line projection matrix

$\boldsymbol{P}_{P}$: Plane projection matrix

$\boldsymbol{R}$: Rotation matrix

$\boldsymbol{R}_{i}$: Rotation matrix of i'th robot joint

$\boldsymbol{R_t}$: Rotation matrix of tool orientation

$\boldsymbol{R_{TCP}}$: Rotation matrix of TCP frame

$\boldsymbol{R_{sh}}$: Rotation matrix of shoulder joints

$\boldsymbol{R_{wr}}$: Rotation matrix of wrist joints

$\boldsymbol{W}$: Weighting matrix

$\boldsymbol{\Omega}$: Angular velocity matrix

LIST OF TABLES

Page

LIST OF FIGURES

Page

TRAJECTORY GENERATION FOR INDUSTRIAL ROBOTS IN PRESENCE OF WRIST SINGULARITY

SUMMARY

Robotics is a highly interested study field that most likely shapes the future of mankind which is very exciting for scientists and engineers. It is a relatively newer engineering area starting from the late 60s and consisting of different disciplines. From there to now, robots are used almost everywhere, on Mars to Earth, on air to land, in deep space to deep oceans, in factories to houses, in laboratories to hospitals. Surely, this is because of their capabilities in three-dimensional space. They can carry out desired tasks like living beings do and become more intelligent as the time goes on.

There are still unsolved problems and active research topics in robotics despite the intense attention. It is possible to see that it is a wide spectrum where plenty of technology branches developed within. New advancements show up inherently and research topics get more complicated with the involvement of different disciplines. Nevertheless, some old problems like inverse kinematics, path planning, or singularity maintain their significance, and suggested solutions vary in the literature.

For articulated robots, kinematic singularities are inevitable phenomenons that are caused by the nonlinear relationship between joint space and operational space. At singularities, the instantaneous motion of the end-effector becomes infeasible at least in a spatial direction called degenerate direction. Singularities should be considered as natural constraints of manipulators and had better be handled specially in any task-prior robotic application. Otherwise, whenever the robot arm is at or near singularities, desired end-effector motion in task space may correspond to very high joint speeds and joint torques which are generally impractical. Fundamentally, kinematic singularities of a 6-DOF industrial robot can be classified into three types: shoulder, elbow, and wrist singularities. Shoulder and elbow singularities occur at determined positions of operational space. The former occurs whenever WCP is on the line of the first joint axis and the latter occurs, in simple terms, whenever WCP is on limits of workspace. Generally speaking, if the task is reachable and far enough to the workspace boundaries, it would be sufficient to avoid these singularities. On the other hand, there is not any pre-determined occurrence position for wrist singularity. The degenerate rotation direction is unique for every different end-effector orientation that robot posture forms. Moreover, robot posture may differ with the selected configuration for the same TCP frame. Therefore, in the level of complexity, wrist singularity is superior to other singularities and can be encountered at any position of robot workspace for a given task.

This thesis intends to conceive a new analytical solution to the wrist singularity of industrial robots which is known for decades. The overall approach is simply avoiding these wrist singular configurations in applications by regarding kinematic redundancy. Still, avoidance is nothing but a reduction of the workspace to a smaller region that selected robot configuration is not varying. The task may exceed that region even if it is in the workspace. Thus, it is not an exact remedy for this problem. The general

solution should be passing through singularity somehow. Certain methods had been applied to both robot control and trajectory generation for singularity-consisting tracking. However, none of these wiped out the infeasible motion of the robot arm whenever it is at singular state. Generally, those methods can be described as a trade-off between tracking error and robot speed. There will be a different viewpoint in this thesis to prove this problem can be solved analytically by defining singular directions on operational space and re-constructing the task in the neighborhood of singularity by concerning these singular directions.

If a smooth transition between wrist configurations were possible, all regions of the workspace would become accessible to a continuous path which greatly increases robot capabilities in various applications such as painting, arc welding, laser cutting, water jet, robot machinery, sealing applications, and others. This thesis is mainly aiming contribution to these 5-DOF required continuous processes in which industrial robots are used. After trajectory generated in specified conditions, one can simply load desired joint angles and derivatives to a typical industrial robot and precisely track the path in a continuous application in presence of wrist singular configurations.

To explain the problem and the solution clearly, mathematical models are going to be presented for both the robot kinematics side and path planning side. Solutions with the existing methods are going to be shown. Then, our novel solution will be compared with their results in different trajectories.

ENDÜSTRİYEL ROBOTLARDA BİLEK TEKİLLİĞİ VE YÖRÜNGE PLANLAMASI

ÖZET

Robotik muhtemelen insanoğlunun geleceğini şekillendirecek, oldukça yoğunlaşılmış bir araştırma alanıdır. Bu sebeple birçok bilim insanı ve mühendisin ilgisini çekmeyi başarmıştır. 60'ların sonunda başlayan ve farklı disiplinleri içeren yeni bir mühendislik alanı olduğu söylenebilir. O günden bugüne robotlar neredeyse her yerde kendilerini göstermişlerdir: Marstan dünyaya, gökyüzünden yeryüzüne, derin okyanuslardan uzayın derinliklerine, fabrikalardan evlere, laboratuarlardan hastanelere. Şüphesiz, robotların üç boyutlu uzayda yapabildikleri sayesinde her çevrede görmemizin mümkün olmuştur. Canlı varlıkların yaptığı gibi istenilen görevleri yerine getirebilirler ve her geçen gün daha da zeki hale gelmektedirler.

Bu yoğun ilgiye rağmen halen tam olarak çözülememiş problemler ve aktif araştırma alanları bulunmaktadır. Robotik, bir çok teknoloji alanının da beraber geliştiği geniş bir yelpaze oluşturur. Yenilikler kendini göstermekte ve araştırma konuları diğer disiplinlerin de dahil olmasıyla da biraz daha karmaşıklaşmaktadır. Yinede; ters kinematik, yörünge planlama veya tekillik gibi eski temel problemler önemlerini yitirmemiştir ve bu yüzden önerilen çözümler litatürde çeşitlik göstermekdir.

Eklemli robotlar için, kinematik tekillikler operasyon uzayı ile eklem uzayı arasındaki lineer olmayan ilişkiden kaynaklanan zorunlu olgulardır. Tekilliklerde, uç-uzuvun anlık hareketi dejenere yön denilen belli bir uzaysal yönde imkansız hale gelir. Tekillikler manipülatörlerin doğal kısıtları olarak görülmeli ve görev önplanlı her robotik uygulamada özel olarak ele alınmalıdır. Bu yapılmadığı takdirde; robot koldan istenilen hareket, tekilliklerde veya civarında, çok yüksek eklem hızlarına veya eklem torklarına karşılık gelebilir ki bunların uygulanması mümkün değildir. Temel olarak, 6 serbestlik dereceli endüstriyel robotların tekillikleri üç tip olarak sınıflandırılabilir. Bunlar omuz, dirsek ve bilek tekillikleridir. Omuz ve dirsek tekillikleri operasyon uzayının belirlenmiş noktalarında oluşur. İlki bilek merkezi noktasının ilk eksenle çakıştığı pozisyonlarda meydana gelir. İkincisi, basit manada, bilek merkezi noktasının operasyon uzayının dış sınırlarına ulaştığı pozisyonlarda meydana gelir. Genel olarak konuşmak gerekirse, verilen iş erişilebilir ve çalışma uzayı sınırlarına yeteri kadar uzak ise, belirtilen tekillikleri atlatmanız için yeterlidir. Diğer yandan, bilek tekilliği için önceden belirlenebilir bir oluşma pozisyonu bulunmamaktadır. Dejenere dönme yönü, robot duruşunun oluşturduğu her farklı uç-uzuv oryantasyonu için özgündür. Dahası, robot duruşu aynı TCP oryantasyonu için farklı konfigürasyonların mümkün olması sebebiyle değişkenlik gösterebilir. Bu yüzden, karmaşıklık düzeyinde, bilek tekilliği diğer tekilliklere göre daha üstündür ve verilen bir iş için herhangi bir robot çalışma uzayı pozisyonunda karşılaşılabilir.

Bu tezin amacı, on yıllardır bilinen bilek tekilliğine analitik bir çözüm sunmaktır. Genel yaklaşım, basitçe, kinematik artıklıktan yararlanılarak bu bilek tekil konfigürasyonlardan kaçınmaktır. Ancak kaçınmak, çalışma uzayını daraltıp robot konfigürasyonlarının değişmediği daha küçük bir bölgeye sınırlandırmaktan başka bir şey değildir. İstenilen görev çalışma uzayı içinde olsa dahi bu sınırlandırılan bölgeyi aşabilir. Dolayısıyla, bu yöntem probleme tam bir çare oluşturmamaktadır. Belli başlı yöntemler bilek tekilliği içeren yörünge takibi için robot kontrölü ve yörünge oluşturma alanlarında geliştirilmiştir. Ancak bunların da hiçbiri robot kol tekil koşuldayken uygulanamaz yönün etkisini ortadan kaldıramamıştır. Bu tez çalışmasında farklı bir bakış açısı ile bu problemin analitik olarak çözümünün mümkün olduğu gösterilecektir. Bunun için tekil yönlerin operasyon uzayında tanımlanması ve istenilen işin tekillikler civarında bu yönlere göre tekrar inşa edilmesi kullanılacaktır.

Eğer bilek konfigürasyonları arasında yumuşak bir geçiş mümkün olabilirse, bütün çalışma uzayı istenilen sürekli bir yörünge için kullanılabilir hale gelicektir. Bu da çeşitli uygulamalarda robotun iş gerçekleştirebilmesini önemli bir şekilde arttıracaktır. Ark kaynağı, boyama, lazer kesme, su jeti, robotlu imalat, yapıştırma gibi sürekli prosesler buna örnektir. Bu tezin yoğunlaştığı, endüstriyel robotların kullanıldığı 5 serbestlik derecesi gerektiren bu sürekli proseslere katkı sunmaktır. Yörüngeler belirtilen kıstaslarda oluşturulduktan sonra bir endüstriyel robota kolayca yüklenip takip edilecek şekilde düşünülmüştür. Bu da bilek tekilliği içeren sürekli yörüngelerin hassas şekilde takibini mümkün kılacaktır.

Problemi ve çözümü açıkca anlatabilmek için, hem robot kinematiği tarafında hem de yörünge planlama tarafında matematiksel modeller sunulacaktır. Mevcut yöntemlerle probleme yaklaşımlar gösterilecektir. Sonrasında tezin kendi ortaya koyduğu çözüm farklı yörüngelerde kıyaslanacaktır.

1. INTRODUCTION

Robotics is a highly interested study field that most likely shapes the future of mankind which is very exciting for scientists and engineers. It is a relatively newer engineering area starting from the late 60s and consisting of different disciplines. From there to now, robots are used almost everywhere, on Mars to Earth, on air to land, in deep space to deep oceans, in factories to houses, in laboratories to hospitals. Surely, this is because of their capabilities in three-dimensional space. They can carry out desired tasks like living beings do and become more intelligent as the time goes on.

The definition of the robot term appears to be changing according to the time it belongs. It came to the English language with the novel of playwright Karel Capek's "Rossum's Universal Robots" which is released in 1920. In that literal, a new machine that is similar to human is mentioned. This ideal machine was working at unpleasant jobs, obeying their human creators and named robot referring word "robota" which means forced labor or drudgery in Slavic languages. However, the underlying idea has deeper meanings philosophically and belongs to earlier times. We can see that people were inventing machines or "automatas" that are similar to living-beings ever since the middle ages [1]. The "robot band" of Al-Jazari (1136-1206) and the "child writer" of Jacquet Droz (1752-1791) are some interesting examples. Moreover, even in ancient texts, creating artificial life can be noticed. Humans have been talented at creating tools and have made autonomous machines, still, have not been able to make a conscious being yet. Possibly, that makes robots fascinating in our popular culture [2]. Illustration art for Capek's play can be seen in **Figure 1.1**.

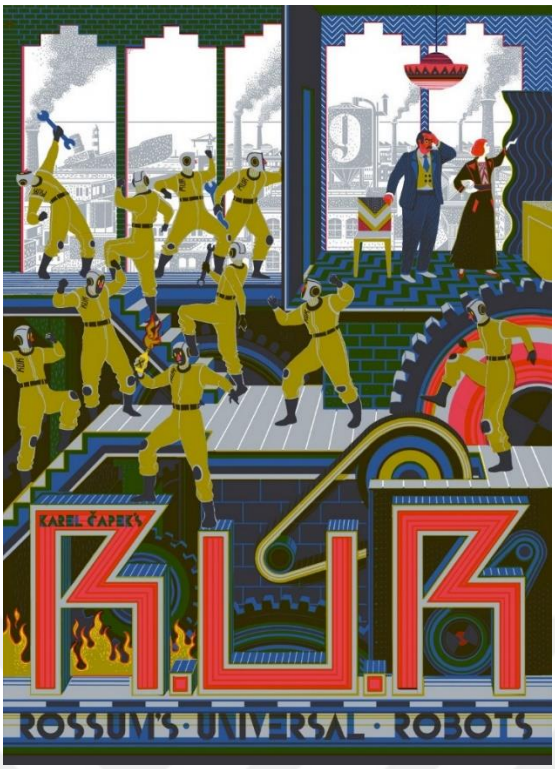

Figure 1.1 : Illustration art for Capek's play [6].

Our intentions are not the fictional meaning of robots but industrial ones. Still, motivation factors may relate to the same underlying idea regarding the arm structure of industrial robots. To deeply conceive industrial robots, it is better to be looked into the historical development of robot technology.

1.1 Development of Robot Technology

Foundations of modern robotics are highly related to the industrial requirements of quality, efficiency, and increasing production rates. Since the industrial revolution, there is a desire for increasing product quality and reducing manufacturing costs [3]. Surely, that is because of market competition and also creating welfare in the general population.

Firstly, we see that products are made by craftsmen whose skills are decisive for production rate and product quality. We call this type of production as manual production. In the year 1905, Henry Ford introduced the concept of mass production which separates works of production into successive stations. In each station, there were specialized machines whose task was pre-determined and generally working with the "bang-bang" principle. These work stations form an assembly line and improve production rates. Mass production had leaded automobiles and other industrial

products to become affordable to the general population by reducing manufacturing costs. Still, it requires high investment costs for establishment. Furthermore, whenever a newer version of a product is going to be produced, there is a necessity for retooling some of the stations and a shut-down for the whole assembly line for a period of time. Today, this type of automation is called hard automation by referring to being inert to the product revision. In response, a new approach had appeared as flexible automation with robot manipulators. They are reprogrammable for a variety of jobs and having a wide workspace with high degrees of freedom. If there is a requirement for reconditioning stations of an assembly line, one is possible to use the same robot in different tasks or may make the same work done with more or less robot count. This provides flexibility in assembly lines which can be re-designed for demands. Robots can be bought off the shelf whenever there is a necessity for them and offer standardized repeatability and accuracy. However, the acknowledgment of industrial robots in assembly lines had taken a long period because manufacturing was an utmost conservative activity in earlier times [4]. They were firstly accepted by the automotive industry and then others.

After Kapek's play, most of the mechanical systems which are generally computer-controlled and owning some degrees of freedom are called robots. However, first industrial robots, in the proper meaning, were appeared after the marriage of two older technologies: CNCs and teleoperators [5]. CNCs or computer-controlled milling machines highly developed in World War II. That is because of the manufacturing quality requirement of certain items, such as warplane components. At the same time, for manipulating nuclear materials in facilities, servo-mechanisms were used which are called teleoperators. The user was remoting the system behind lead-glass protection while mechanism follows the master-slave principle.

In the year 1954, inventor George C. Devol, Jr. thought technology had reached a sufficient level to develop a robot in the strict sense that is capable of doing works as humans do. He applied for a patent on "Programmed Article Transfer" in the U.S. which is the first digitally operating and the programmable robotic arm. Thereafter, he met with the engineer Joseph F. Endelberger and they founded the first robotic company of the world, Unimation. They had created their robot brand, called Unimate, and installed it on General Motors factory in 1961 for the application of die casting. It was a remarkable point of industrial robots as a unification of CNC and teleoperator

subsystems and also for successful integration into a manufacturing process. Still, Engelberger states that finding customers for implementing industrial robots was not easy at that time because no one needed a robot actually, considering everything a robot does is also possible to be done by humans [4]. So, they gave more attention to institutional activities to propagate industrial robots around the world, not only for industry but also academy. Laboratories at Stanford University and Massachusetts Institute of Technology started to develop their robot manipulators in these years. Also, the Japanese were highly interested in the robot concept. They firstly awaken to the significance of using robots in industry and they founded the first national robot community, JIRA, which consists of 46 industrial incorporations. In the year 1968, Japanese giant company, Kawasaki Heavy Industries, made a license agreement with Unimation Inc. to produce the Unimate robots in Japan [7]. The robot can be seen in **Figure 1.2**.

Figure 1.2 : Japan's first domestically manufactured industrial robot (courtesy of Kawasaki Robotics).

In the 1970s, the industrial world had realized the future of robotics. Other robot manufactures and users had appeared in many countries. Cincinnati-Milacron Inc. developed the first hydraulic heavy-load serial robot, T3, in 1973 [8]. IRB-6 of ASEA (now ABB) was the first electrically driven serial robot arm that was produced in 1974. PUMA robot of Unimation Inc. was developed for a project of General Motors in 1979. It was designed with the inspiration of human arm dexterity and used in many academic studies later on [9]. These robots can be seen in **Figure 1.3** and **Figure 1.4**.

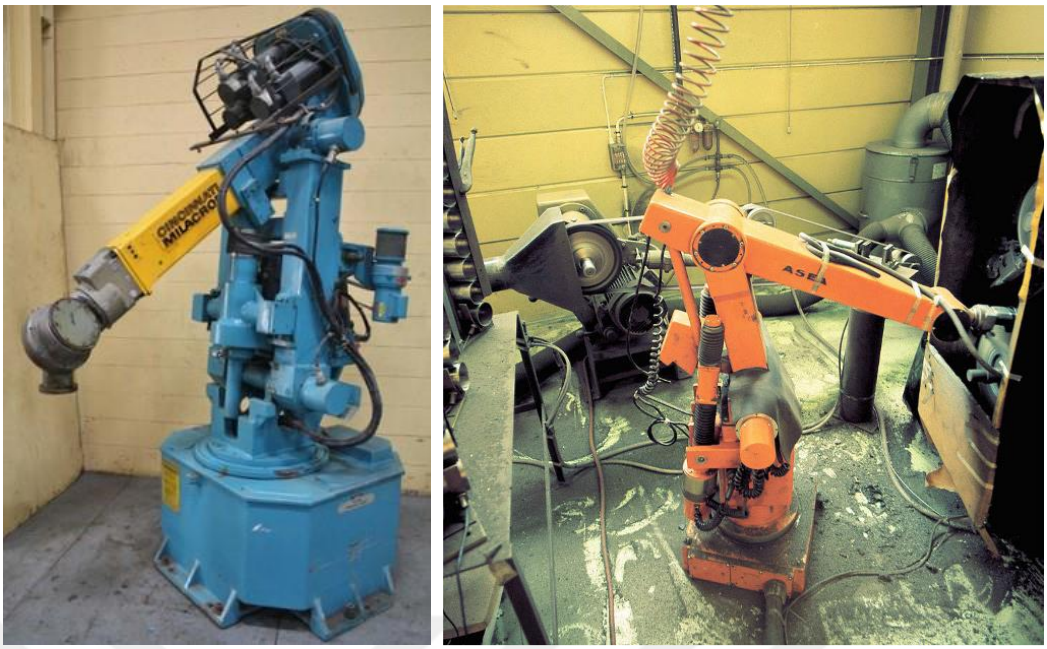

Figure 1.3 : Cincinnati-Milacron T3 [8] and ASEA IRB-6 (courtesy of ABB Automation).

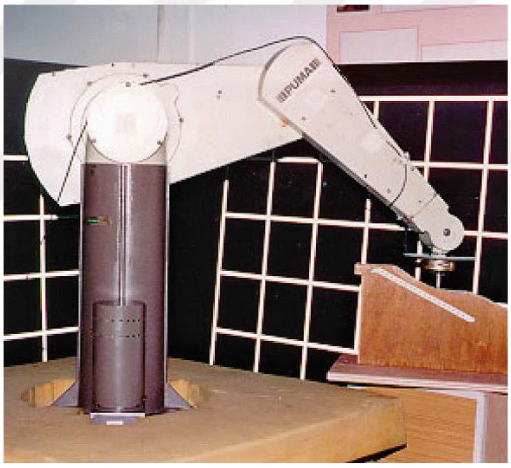

Figure 1.4 : PUMA (Programmable Universal Machine for Assembly) robot [10].

These primitive robots began to find places in many industries. The motivation was simply replacing human workers at first. However, their capabilities were rather insufficient to compete with any human worker [2]. Some key technologies, such as compliance and machine vision, were lacking at the beginning. They were not able to carry out tasks that are not pre-determined. Programming them was difficult. Moreover, they were prohibited to work together with humans because they could be fatal if the safety conditions were not provided. These made robots to be used mostly in hazardous and non-ergonomic environments which human workers cannot work. Ultimately, replacing human workers directly was not an easy issue to accomplish and more research and development studies were needed.

1.2 Today's Industrial Robots

Today we see that serial industrial robot designs are not very different from the ones developed in the 70s. Yet, they are highly specialized according to applications, categorized to the purpose of realizing industry jobs, e.g. painting robots, welding robots, handling robots. Accuracy and repeatability of industrial robots were already enormous, providing quality and consistency in products. On the other hand, performance parameters such as speed, payload, and mean time between failures (MTBF) have dramatically enhanced which are very significant for productivity and efficiency in manufacturing. Furthermore, the average robot price in 2014 had become one-third of its equivalent price in 1990 which leads to automation becoming a more affordable, faster return of investment [10].

Industrial robots bear the main workload of the agile-manufacturing jobs of today. In this purpose, stations in assembly lines are specially designed for robots and other auxiliary types of equipment. Many engineering corporations have appeared to establish efficient robotic turn-key systems in OEMs. According to IFR, the annual shipment of industrial robots is increasing unless there is a global economic crisis [11]. A chart for the annual shipments of industrial robots given in **Figure 1.5**. Most probably, industrial robots will remain significant in the intelligent factories of the future.

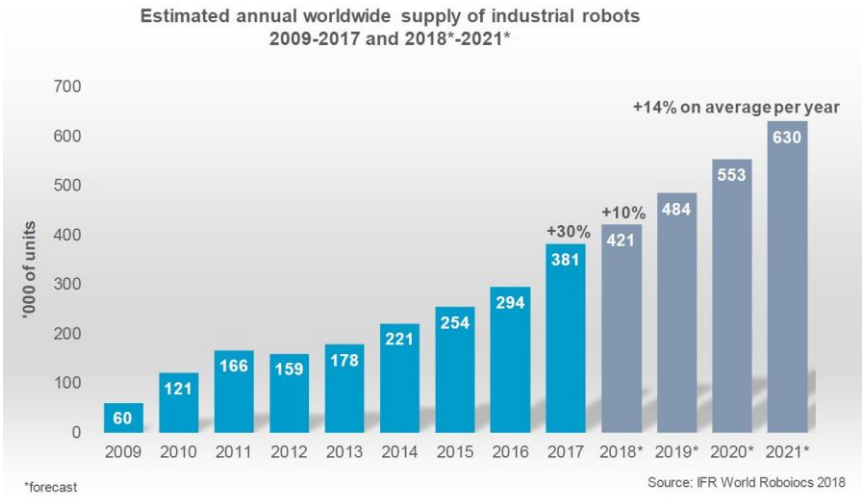

Figure 1.5 : The increasing supply of industrial robots (IFR) [11].

Industrial robots span a fundamental part of robotics but not all. As a result of historical development, contributions to the robotic research were mainly done for industrial robots at first. Whereas, in the course of events, numerous types of robotic systems were discovered holding features that industrial robots are lacking or incapable. Mobile robots appeared in different environments with challenging tasks, such as drones, sea voyagers, and mars explorers. Furthermore, service robots arose in houses and hospitals assisting people in daily life problems. Even non-mechanical systems can be classified in today's robotics with the advance of artificial intelligence. Consequently, for the sake of this study, it is required to define a scope for the type of robot that is going to be considered.

This thesis aims a kinematic study for singularity, especially on the wrist side of industrial robots. Therefore, only a specific type of industrial robot is going to be taken in the scope that is called articulated or serial industrial robots. On the contrary, parallel industrial robots are going to be taken out of the scope because they require a special kinematic approach. Moreover, it will be assumed that serial robots contain only one DOF rotary joints. Classification of industrial robot types will be done concerning axes count and differing wrist types because the shoulder part of the system is already standardized for years. Robots that are already recognized and highly used in industry will be presented in the next sub-topic.

1.2.1 Six successive axis industrial robots

1.2.1.1 Ortho-parallel shoulder with spherical wrist

Over ninety percent of robots used in today's industry can be classified in the kinematic design of ortho-parallel shoulder with a spherical wrist. It is one of the simplest design that is developed for industrial needs. It satisfies varying applications via the large workspace it has. This large workspace is provided by the first three rotary joints called the shoulder. By comparison to spanning areas, rotary joints are more advantageous rather than linear joints because of their sizes. Linear joints require guides through limits of their workspace but rotary joints use links directly. With this shoulder design, it is possible to span nearly a spherical space. An example of a six-axis industrial robot can be seen in **Figure 1.6**.

Figure 1.6 : ABB IRB-6700 robot [12].

For satisfying orientation requirements, the last three joints are decisive. Therefore, they are designed as an equivalent of a spherical joint. Whereas, in serial kinematic designs, it is not possible to get the exact equivalent of spherical joint because in some configurations serial joints lose degrees of freedom. This is the point where wrist singularity is encountered. A representation of spherical joints is given in **Figure 1.7**.

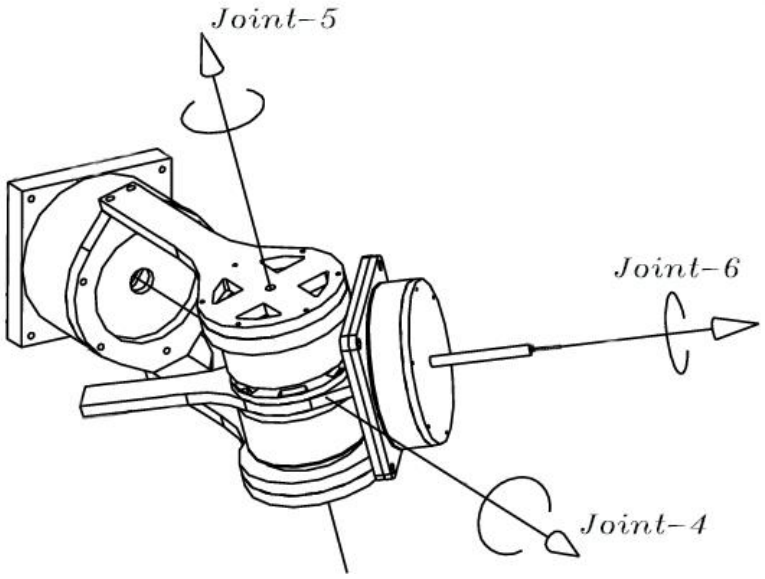

Figure 1.7 : Wrist joints (modified from the reference [32]).

Despite the prevalence in most of the applications, ortho-parallel shoulder with spherical wrist robots are barely suitable for continuous path tracking applications because satisfying high orientation change may not be possible. The reason is that joint-5 has mechanical limits that do not allow turning 360° which causes collision

through the robot itself. Moreover, wrist singular configuration stands in the middle of joint-5 limits but it would be discussed in another topic.

1.2.1.2 Other wrist types

Alternatively, non-spherical wrist types have appeared in some robots that are used at continuous applications such as painting and welding. The wrist in **Figure 1.8** contains 3 angled axes that can rotate more than 360°. However, as the wrist axes do not coincide at one point, these wrist type requires a more complex kinematic approach to obtain closed-form inverse solution. One can visit the reference [38] for additional information.

Figure 1.8 : MPX3500 painting robot (courtesy of Yaskawa Motoman).

Universal Robot company produces robots that owning shifted wrist structures. **Figure 1.9** shows one of its robots. This allows joint-5 to be turning more than 360° and they have a closed-loop inverse geometric solution. Yet, robots can collide itself while there is a tool at the end-effector.

Figure 1.9 : UR5 robot (courtesy of Universal Robots).

Increased joint limits are advantageous for applications. Still, they do not provide a remedy for singularity.

1.2.2 Industrial robots with higher than six axes

To increase kinematic capacities of robot manipulators, industrial robots with higher than six axes are produced by manufacturers which we call intrinsically redundant manipulators. They are preferred for somewhat more specific and more difficult continuous applications such as painting or deburring. Further, the extra axis allows changing robot configuration internally. This redundancy can be used for actively controlling robot posture without changing end-effector position and orientation which assists in avoiding singular configurations and collisions.

1.2.2.1 Four-axes wrist

For avoiding wrist singularity in a continuous application, one of the solutions is adding another axis to the manipulator at the wrist side. An example robot can be seen in **Figure 1.10**. The additional axis allows controlling joint-5 actively for the given path. Yet, it does not simply solve the wrist singularity. The degenerate rotation axis still exists in the kinematics of the manipulator. One can avoid this configuration by using redundancy in wrist axes.

Figure 1.10 : IRB-5510 robot (courtesy of ABB).

As shown in **Figure 1.11**, in some configurations 4 axis wrist joints can lie on a common plane. In those cases, it is not possible to rotate EE in the axis perpendicular to this plane. However, by using redundancy, this configuration can be tried to be avoided.

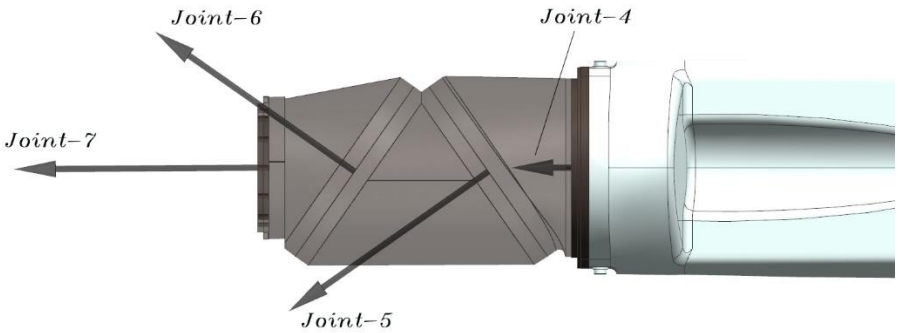

Figure 1.11 : 4 axis wrist joints of IRB-5510 on the common plane.

1.2.2.2 Spherical-rotational-spherical

One of the most important redundant robot types is a 7-axis robot whose kinematic chain called Spherical-Rotational-Spherical (S-R-S). They are referred to as robots of the new generation. **Figure 1.12** shows a renowned example. Actually, the arm structure is anthropomorphic and generally used for arms of humanoids. The approach for the kinematic solution of these robots is a bit different than classical industrial

robots. Still, they also suffer from wrist singularity. Applications of this type of robots are generally material handling with force control. Unfortunately, their kinematic structure is out of our scope.

Figure 1.12 : KUKA LBR iiwa robot (courtesy of KUKA Gmbh).

1.2.3 Industrial robots with lesser than six axes

For tasks that are not requiring rotating through roll and pitch axis such as palletizing and basic assembling, robots can be specially designed with lesser axes. One of the most used industrial 4 axis robots is given in the figures below. They may be more effective at loading capacity or speed in some applications rather than classical 6 axis industrial robots. Whereas, their operational space limited by only one DOF for orientation. For satisfying orientation requirements, they use only one joint. That is why they do not suffer from wrist singularity.

Palletizing robots consist of special mechanisms for maintaining the wrist axis always vertical which is shown in **Figure 1.13**.

Figure 1.13 : FANUC M-410*i*B/700 robot [15].

SCARA robots are suitable for fast assembly tasks. By their kinematic design, their last two joints stay always vertical with respect to the base frame. Generally, they are used in the electronic industry. An example can be seen in **Figure 1.14**.

Figure 1.14 : TP80 robot (courtesy of Stäubli).

1.3 Robot Kinematics

Manipulators contain joints that describe the capability of motion between the links they are attached. Joints can be both translational or rotational. For serial industrial robots, joints are generally rotational through an axis. Joints are successive which is causing a chain of motion through links. For a joint, the previous link remains fixed and the successive link or links are driven by the motion of this joint. There are seven links for a six axes robot where the zeroth link is always stationary. The first link is

only driven by the first joint. The sixth link is affected by all of the robot joints. An illustration of this open kinematic chain is given in **Figure 1.15**.

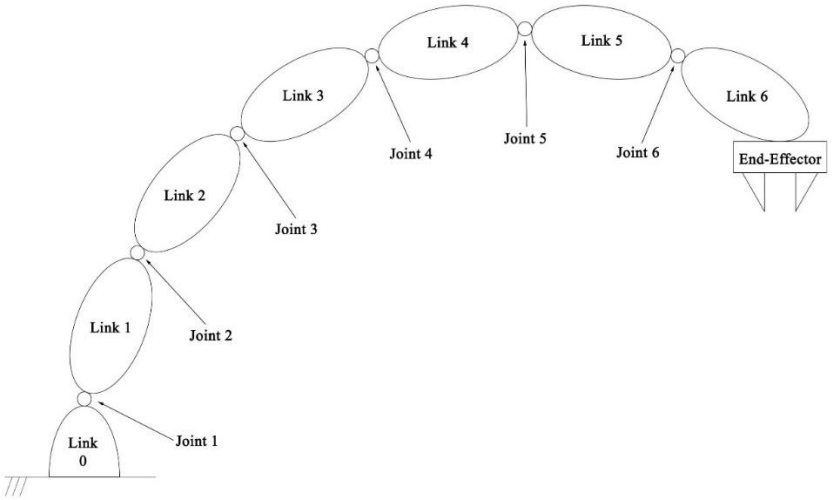

Figure 1.15 : Kinematic chain of a six successive axes robot.

Industrial robots generally have six successive joints. The reason is that any free body in three-dimensional space has six degrees of freedom, three for translation through any orthogonal frame axes, and three for rotation through any three-dimensional vector. To manipulate an object freely in space, one must have six independent motion parameters like a six axes serial industrial robot which manipulates its end-effector.

If position and orientation (also called *pose)* of the end-effector are termed as a vector, $\boldsymbol{x}$, in cartesian space and joint displacement parameters are defined as also a vector, $\boldsymbol{q}$, the relation between them must be a function as below.

$$\boldsymbol{x} = f(\boldsymbol{q}) \tag{1.1}$$

The operated function here is called *forward geometrical model* (FGM) of the robot. The forward term is used regarding the kinematic chain direction. The vector size of the $\boldsymbol{q}$ is only related to the joint count of the manipulator. If the robot has n joints the vector would be:

$$\boldsymbol{q} = [q_1 \dots q_n]^T \in \mathfrak{J} \tag{1.2}$$

Where $\mathfrak{J}$ is called as *joint space* and describes the space of its independent joint parameters. For serial manipulators, all of the joints are independent. Hence, its

dimension is *n*. Any value that joint parameters corresponded forms a different *robot posture*.

For end-effector motion, $\boldsymbol{x}$ vector can be arranged according to interest because any coordinate system can be used. Cartesian coordinates are best suited for representing the position. Whereas, representing the orientation through Cartesian axes is a bit challenging. Euler angles or roll-pitch-yaw angles can be used for minimum parameter representation but they may mathematically singular in special cases. Orientation representation will be entirely discussed in Section 2.2. Here, we can represent the pose of the EE by six independent parameters as below.

$$\boldsymbol{x} = [x\ y\ z\ \phi\ \theta\ \psi]^T \in \mathfrak{O} \tag{1.3}$$

Where $\mathfrak{O}$ is denoted for *operational space* of the end-effector which exists as a result of the FGM. Because of any free body in space being able to move in DOF of maximum six, hence, the maximum dimension of the $\mathfrak{O}$ can be six. Yet, it is possible to be reduced for the joint space dimension. For the dimension of $\mathfrak{O}$, we use the symbol, *o*.

Furthermore, for a specified task, the motion of the end-effector may span the whole of the operational space or be its subset. In any case, the space of undergoing motion for the task can be defined as *task space*, $\mathfrak{T}$, whose dimension referred to as *m*. It must be noted that most of the robotic applications require less than six DOF.

A robot must satisfy the following condition to carry out any task in its operational space:

$$n \geq o \geq m \tag{1.4}$$

If this condition is satisfied, it is possible to say that a solution exists for the inverse geometrical model of the robot manipulator as below.

$$\boldsymbol{q} = f^{-1}(\boldsymbol{\chi}) \tag{1.5}$$

For a typical industrial robot, the first three axes or shoulder axes are responsible for the translation part of the manipulation. Besides, the last three axes or wrist axes are dedicated to orientation. This is caused by the special kinematic design of this type of robot. Wrist axes coincide at one point which is called wrist center point (WCP).

Shoulder axes are assembled as the ortho-parallel structure that is able to span a spherical volume if there are no joint limits specified. The reason for this decoupled kinematic design of the robot is related to the simplicity of its inverse geometric model of the mechanical system.

1.4 Definition of Kinematic Redundancy

Different types of serial robots have been shown with their kinematic structure. The reason for this contrast is related to their operational space. Robots are dedicated to carrying out tasks that are subsets of their operational capacity. By considering spaces that we have introduced, it is possible to define three types of redundancy as Huo (2005) stated:

- **Intrinsic redundancy:** For a serial manipulator whose dimension of the joint space, n, is greater than its dimension of the operational space, o; it is said to be an intrinsically redundant manipulator. Hence, in the case of $n > o$, redundancy can be computed as follows.

$$r_I = n - o \tag{1.6}$$

- **Functional redundancy:** For a task that is going to be realized by a manipulator, it is said to be functionally redundant if the EE dimension of operational space is greater than the dimension of task space and only if the task is placed in the operational space of the EE. Hence, in the case of $o > m$, redundancy can be computed as follows.

$$r_F = o - m \tag{1.7}$$

- **Kinematic redundancy:** For a task that is going to be realized by a manipulator, it is said to be kinematically redundant if the manipulator has a greater dimension of joint space than the dimension of the task space of the task and only if the task is placed in the operation space of the manipulator EE. Hence, in the case of $n > m$, redundancy can be computed as follows.

$$r_K = n - m \tag{1.8}$$

Moreover, we can define kinematic redundancy as a summation of intrinsic (manipulator related) redundancy and functional (task-related) redundancy:

$$r_K = r_I + r_F \tag{1.9}$$

Kinematic redundancies cause the infinite solution to inverse problems, however, they can be used for the optimization of the following features:

- minimizing the norm of the joint velocities [14],
- task prior path tracking [17],
- avoiding obstacles [13],
- avoiding singular configurations [15][16][17][18][19],
- avoiding joint limits [20],
- minimizing driving joint torques [16].

1.5 Functional Redundant Robotic Applications

Any serial product competing in the global market requires a specified manufacturing quality which is significant for its added-value. If the manufacturing process contains only basic rotations, it can be carried out by specialized machines. Whereas, for a complex process that contains spatial rotations, industrial robots could be the only solution. As a result of that, robotic applications take place in various industries and different processes.

Most of the robotic applications do not require six degrees of freedom to be accomplished. There may be symmetric axes or planes in tasks which let manipulator carry out the same job with different EE orientations. This assists in finding a solution more conveniently in dense or unreachable areas. As the robots are at the center of flexible automation, functional redundancy is very significant in practice. Let us view the most used robotic applications.

1.5.1 Welding

Welding is a joining process of materials via heating them on the surface. For heating, several energy types can be used e.g. chemical, electrical, laser, or mechanical. However, for robotic applications electrical energy is generally preferred. Two types

of electrical power supplied welding are become prominent in industrial robotics: resistance welding and arc welding. These are the most widespread applications in robotic welding, besides, they are essential processes for automobile assembly manufacture.

1.5.1.1 Resistance welding

Resistance Welding is a process in which the heat required for welding is produced by the electrical resistance between parts at a specified location. The process requires clamping the parts with pressure in a region. Welding current that flowing through parts in this region causes the creation of a welding nugget on lap joint. An illustration of the process is given in **Figure 1.16**.

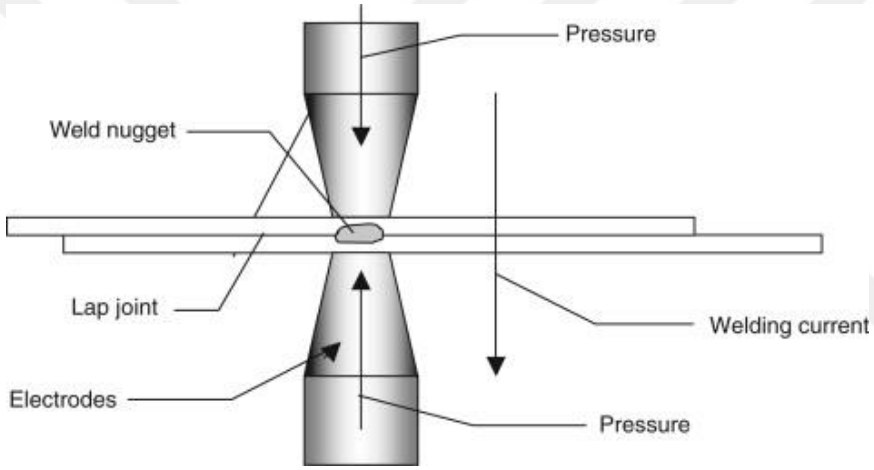

Figure 1.16 : Resistance welding [37].

Generally, robots carry welding guns through specified points to be welded. Welding guns carry out the jobs of clamping and electric flow. The process does not require movement while the welding is happening. Hence, it is not a continuous process but redundancy along the symmetry axis is very essential for approaching from different angles. The collision of welding gun through parts can be avoided by using this redundancy. Besides, for decreasing the cycle time of the robot, approach angles are specially designed by process planners. An example process can be seen in **Figure 1.17**.

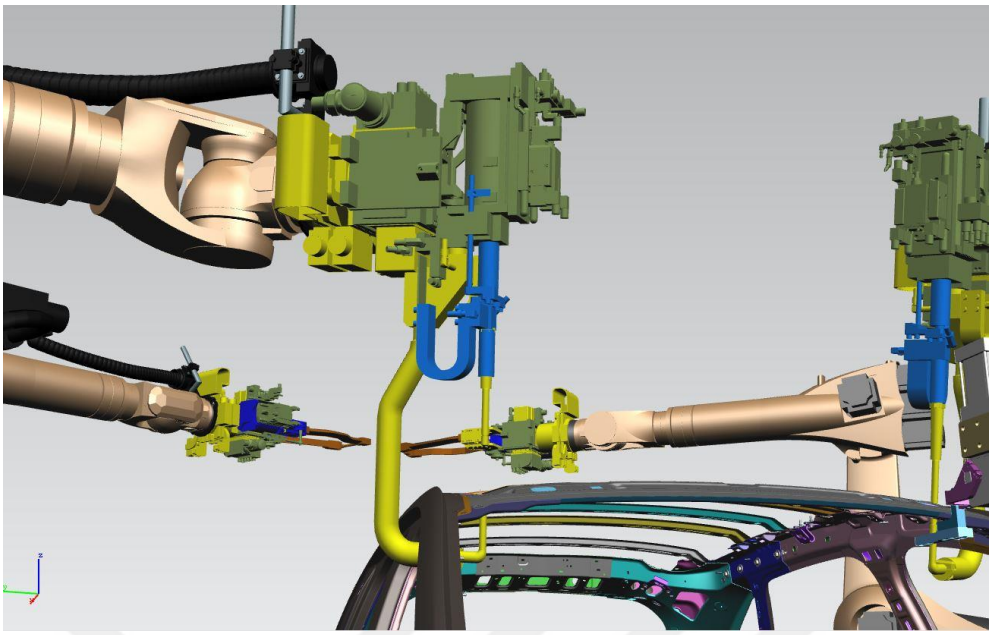

Figure 1.17 : A spot welding application to the car frame (courtesy of Altinay).

1.5.1.2 Arc welding

Arc welding is another popular process in industrial robotics that uses the electric arc to weld parts. There are two prominent subtypes of this process according to method e.g. Gas Tungsten Arc Welding (also known as TIG welding) and Gas Metal Arc Welding (also known as MIG/MAG welding). The main difference between them is the GTAW process uses a non-consumable tungsten electrode while GMAW uses consumable electrode wire for welding. In assembly lines, especially in automobile manufacturing, GMAW is more widely used concerning GTAW because of economic reasons. A schematic of the GMAW process can be seen in **Figure 1.18.** The heat of the electric arc melts both electrode wire and metal components which carry-outs welding. Protection gas required for the fusion to prevent pernicious contamination with some gases of the atmosphere. Electrode wire is fed by an automatically controlled coil where its speed is crucial for the process. In general, the magnitude of electric current and voltage, wire diameter and feed speed, used protection gas are important input parameters of the welding process to meet requirements of weld penetration [43].

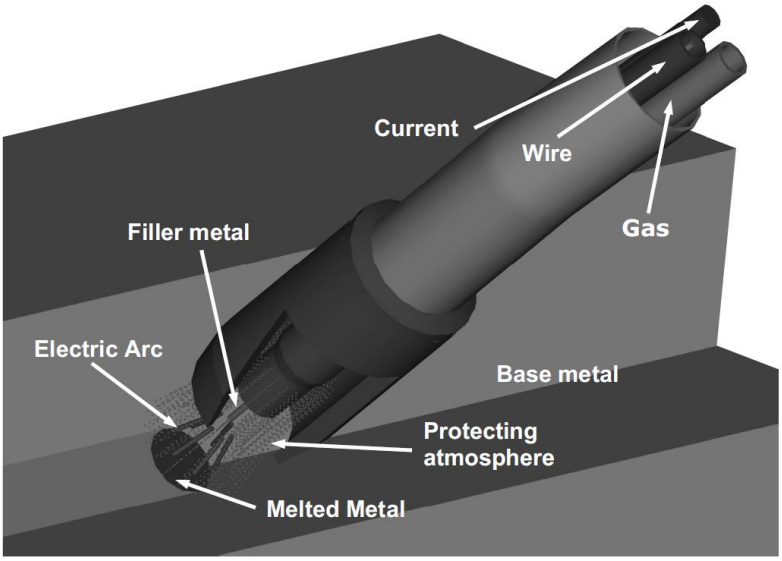

Figure 1.18 : Arc welding process [43].

In the robotics point of view, arc welding is a five DOF operation, where the end-effector is the arc welding torch. Its distance to the parts being welded and orientation angles to the surfaces of the parts should not change or the change must be small through the process because they affect welding quality. Holding the same pose along the whole process can be difficult when part geometries are complex. In that purpose, welding torches are angled for the robot to access easily. An example can be seen in **Figure 1.19**. Arc welding machine which is the source of required electric current and voltage generally stays out of robot cell. Thus, it stays out of operation. Whereas cable package and wire feeder unit are dressed on robot, and sometimes they can collide to the robot or the other equipment.

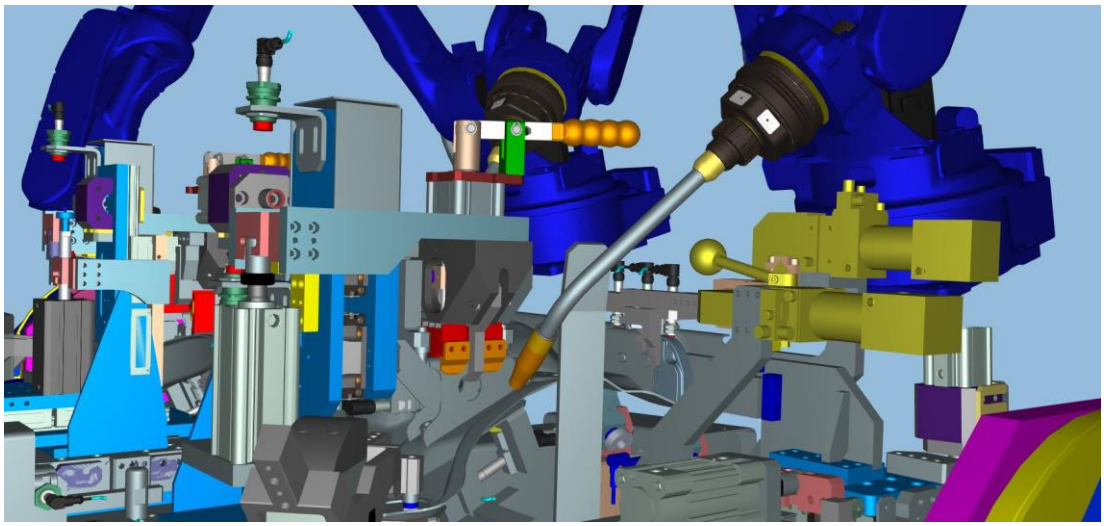

Figure 1.19 : An arc welding application (courtesy of Altinay).

There are two arc welding angles that should be specified with respect part surfaces: travel angle and work angle which are shown in **Figure 1.20**. If they are held at desired values for the whole process, weld quality generally would meet requirements. To achieve this in complex geometries, CAD software and simulations can be useful. Furthermore, there is a symmetry axis through the electrode wire axis, where rotation of EE is irrelevant for welding quality. That is why process requires 5 degrees of freedom and redundant DOF are useful for any other purpose, e.g. singularity avoidance, collision avoidance, minimum displacement for joints, etc.

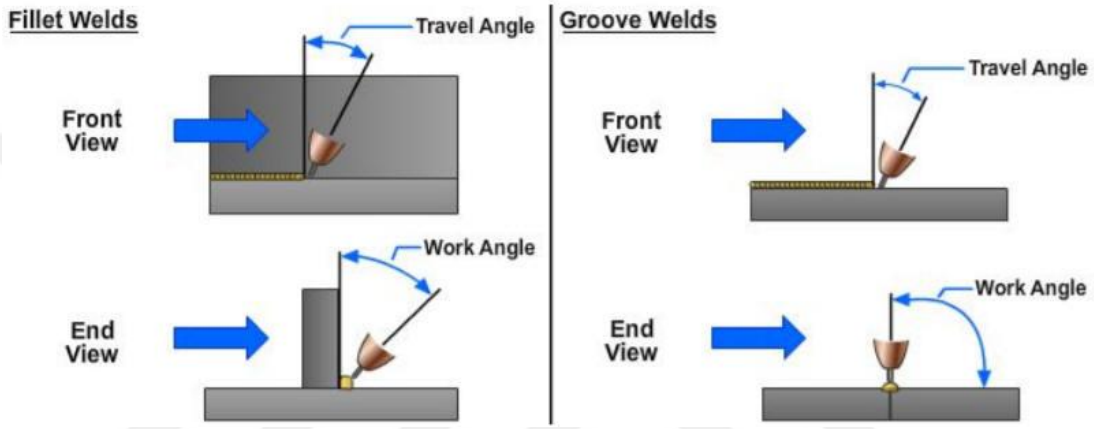

Figure 1.20 : Travel and work angle of welding torch [36].

1.5.2 Other continuous processes

Laser Cutting and Welding, Painting, Sealing, Hemming, and Machining are the other continuous processes owning functional redundancy in which industrial robots are used. Typically, similar to the arc welding process, all these processes have a special process tool to be manipulated by the robot as an end-effector. Occasionally, robots do not carry process tools as end-effector but parts and process tool stands on peripheral equipment. In that case, robot manipulates part or parts that are going in the process and carry out the process by reaching the process tool. Examples can be seen in **Figure 1.21** and **Figure 1.22**.

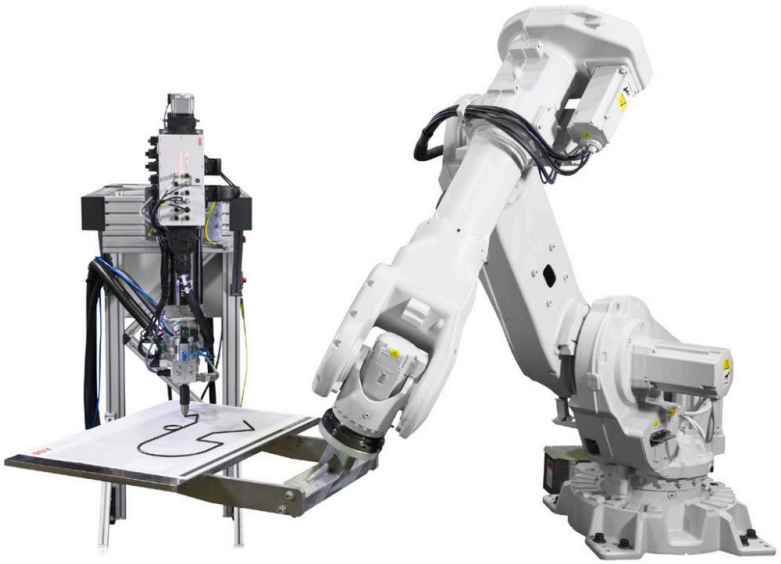

Figure 1.21 : Sealing process application (courtesy of ABB).

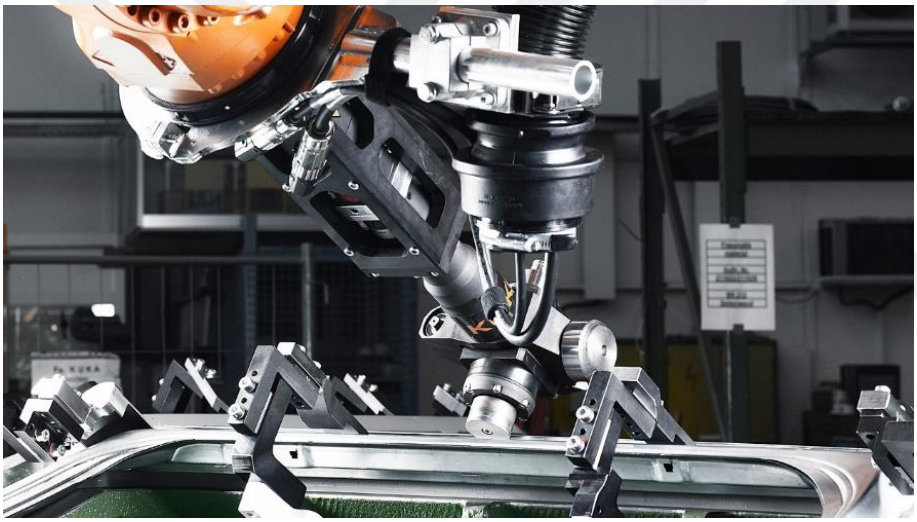

Figure 1.22 : Hemming process application (courtesy of KUKA Gmbh.).

These processes have assorted difficulties according to the quality requirements, cost, production cycle time, etc. Henceforth, their margins for manipulation errors can differ for each other just like all manufacturing processes. Whereas, there are predictable process speeds which we can compare them for kinematic analyses. A process speeds table is given in **Table 1.1** showing general intervals for the common applications.

Process Type	Process Speed [mm/s]
Resistance Welding	-
Arc Welding	5 - 50
Laser Welding	30 - 100
Painting	600 - 1000
Sealing	500 - 1000
Hemming	500 - 1400
Machining	0 - 30

Table 1.1 : Comparison of process speeds.

1.6 Definition of Kinematic Singularities

For articulated robots, kinematic singularities are inevitable phenomenons that are caused by the nonlinear relationship between joint space and operational space. At singularities, the instantaneous motion of the end-effector becomes infeasible at least in a spatial direction which is called degenerate direction. Singularities should be considered as natural constraints of manipulators and had better be handled specially in any task-prior robotic application. Otherwise, whenever the robot arm is at or near singularities, desired end-effector motion in task space may correspond to very high joint speeds and joint torques which are generally impractical. Fundamentally, kinematic singularities for a 6-DOF ortho-parallel manipulator with the spherical wrist can be classified into three types: shoulder, elbow, and wrist singularities. Shoulder and elbow singularities occur at determined positions of operational space which can be seen in **Figure 1.23** and **Figure 1.24**. The former occurs whenever WCP is on the line of the first joint axis and the latter occurs, in simple terms, whenever WCP is on limits of workspace. These singularities can be sensed by controlling the position of WCP for a given task and they are easily avoidable by changing robot base position or

using different tool connections. Generally speaking, if the task is reachable and far enough to the workspace boundaries, it would be sufficient to avoid these singularities.

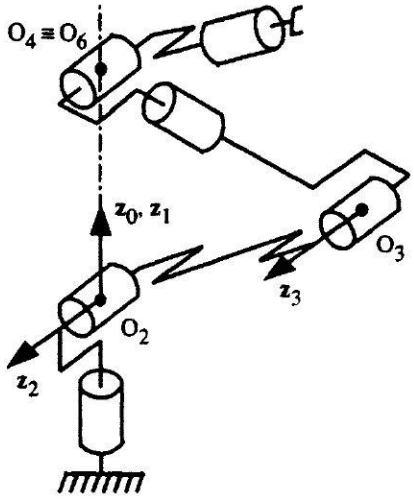

Figure 1.23 : Shoulder singularity [21].

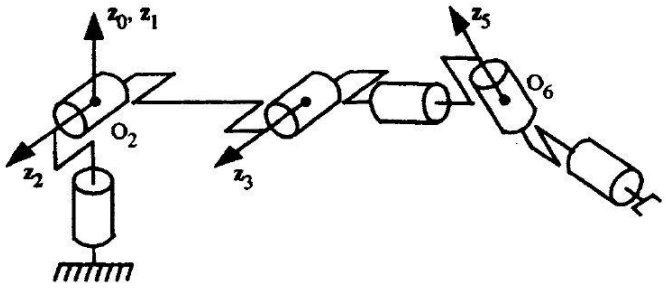

Figure 1.24 : Elbow singularity [21].

On the other hand, there is not any pre-determined occurrence position for wrist singularity. Please see **Figure 1.25** for visualization. The degenerate rotation direction is unique for every different end-effector orientation that robot posture forms. Moreover, robot posture may differ with selected shoulder configuration for the same WCP. Therefore, in the level of complexity, wrist singularity is superior to other singularities and can be encountered at any position of robot workspace for a given task.

Mathematically, there are sixteen different configurations for a six-successive-axis robot. If it is an ortho-parallel robot with a spherical wrist, like a typical decoupled industrial manipulator, the possible configuration count is eight. Each singularity is placed in the transition point of the configuration types. Thus, configurations are classified with respect to the singularities and each has two possible choices which

make total possibility eight. In a continuous application, one of these configurations is selected as an initial condition by considering mechanical joint limits and robot manipulability. The selected configuration is held along the path to carry out the task. However, selecting a configuration, actually, constraints the workspace to a smaller region. The path may exceed that region even though it is in the workspace. In such a manner, the robot may try to pass through the singularity and change its configuration. Yet, at and near singularities desired movements might not be possible.

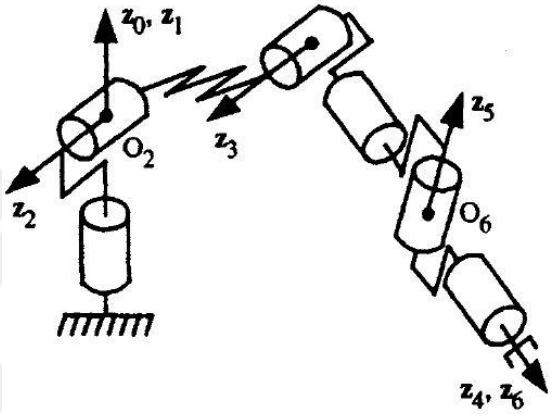

Figure 1.25 : Wrist singularity [21].

Wrist singularity might not be easily detected because it is a function of end-effector position and robot configuration. Actually, it occurs when the 2 of the wrist axes get aligned and become kinematically identical. This problem is also known as gimbal lock in other study fields. Thus, for every different position and configuration, there is a unique wrist singular orientation that constraints the limits of the selected configuration. By considering robot should track more than one path in an application, every path should be controlled by a designer for the wrist singularities of the robot. Without using simulations, it is very difficult to detect whether the selected configuration is going to make robot wrist singular, along the path. If it does in a path, using the solutions above for non-wrist singularities might cause encountering wrist singularity in other paths. Furthermore, it is not sufficient to avoid just exact singular orientations but also a specified closeness to that orientations. Because robot movement is still effected near wrist singular orientations by the robot Jacobian that has grown into ill-conditioned. There are numerical methods based on manipulating

Jacobian in a non-singular way but if the singular area is going to be passed by these methods, there would be orientation errors and reduced TCP speeds that are not desired in the applications. Hence, the wrist singularity restrains robot capabilities in its workspace and may make desired tasks inapplicable.

1.7 Purpose of the Thesis

The intention of this thesis is to conceive a new analytical solution to the wrist singularity of industrial robots which is known for decades. The overall approach is simply avoiding these wrist singular configurations in applications by regarding functional redundancy. Still, avoidance is nothing but the reduction of the operational space to a smaller region that robot configurations are not varying. The task may exceed that region even if it is in the workspace. Thus, it is not an exact remedy for this problem. The general solution should be passing through singularity somehow. Certain methods had been applied in both robot control and trajectory generation for the paths containing singular configurations. However, none of these wiped out the infeasible motion of the robot arm whenever it is at singular state. Generally, those methods can be described as a trade-off between tracking error and robot speed. There will be a different viewpoint in this thesis to prove this problem can be solved analytically by defining singular directions on the operational space and re-constructing the task in the neighborhood of singularity with respect to these singular directions.

1.8 Hypothesis

If a smooth transition between wrist configurations were possible, all of the workspaces would become accessible to a continuous path which is greatly increasing robot capabilities in various applications such as painting, arc welding, laser cutting, water jet, robot machinery, sealing applications et cetera. This thesis is mainly aiming contribution to these 5-DOF required continuous processes in which industrial robots are used. To achieve this aim, the robot trajectory can be specially organized in the manner of compensating degenerate direction while passing through wrist singularity. It requires a study for kinetostatic performance of the robot in the whole path and re-construct task by using redundancy regarding wrist singularity which is possible by using an off-line programming algorithm that is going to be expressed in this thesis.

After trajectory generated in specified conditions, one can simply load desired joint angles and derivatives to a typical industrial robot and use it in a continuous application in presence of wrist singular configurations that do not cause any extraordinary path-tracking error.

2. MATHEMATICAL MODELS

Any robotic system requires a suitable mathematical model that defines its motion capabilities in Euclidean space. The model can vary depending on the intention, but in this thesis, rigid body mechanics are going to be used. Assuming robot links rigid is quite valid in literature for motion representation. We are going to compose models containing vectors and matrices which will be shown in the bold case for ease of articulation. Vectors would be represented in lowercase letters and matrices in uppercase. Further, quaternion vectors are going to be shown with an overbar.

2.1 Vectors and Frames

Vectors and frames are essential tools for forming a basis of kinematic representations in three-dimensional space. For determining the position of a point in space, it is required to have a coordinate system. In robotics and also other disciplines, Cartesian coordinate systems or frames are very useful because they contain three orthogonal unit vectors being coincident at origin which directly gives us values in three orthogonal directions of any created vector in the given Cartesian coordinate system. By convention, right-hand sided frames will be used.

For the representation of a point, we use free vectors and determined frames which are illustrated in the figure below. By assuming that we are knowing origin locations of given frames, position vectors can be created from origins through a specified point. Those vectors can be represented in three components by the frames they belong to. We do require vectors to be projected into frame basis vectors to find vector components. By doing so, point position vectors in **Figure 2.1** are mathematically represented below by indicating the frame they belong to. We use dot product (direction cosine) for projection.

$$[\boldsymbol{p}]_0 = \begin{bmatrix} \boldsymbol{v}_0 \cdot \boldsymbol{x}_0 \\ \boldsymbol{v}_0 \cdot \boldsymbol{y}_0 \\ \boldsymbol{v}_0 \cdot \boldsymbol{z}_0 \end{bmatrix}, \qquad [\boldsymbol{p}]_1 = \begin{bmatrix} \boldsymbol{v}_1 \cdot \boldsymbol{x}_1 \\ \boldsymbol{v}_1 \cdot \boldsymbol{y}_1 \\ \boldsymbol{v}_1 \cdot \boldsymbol{z}_1 \end{bmatrix} \tag{2.1}$$

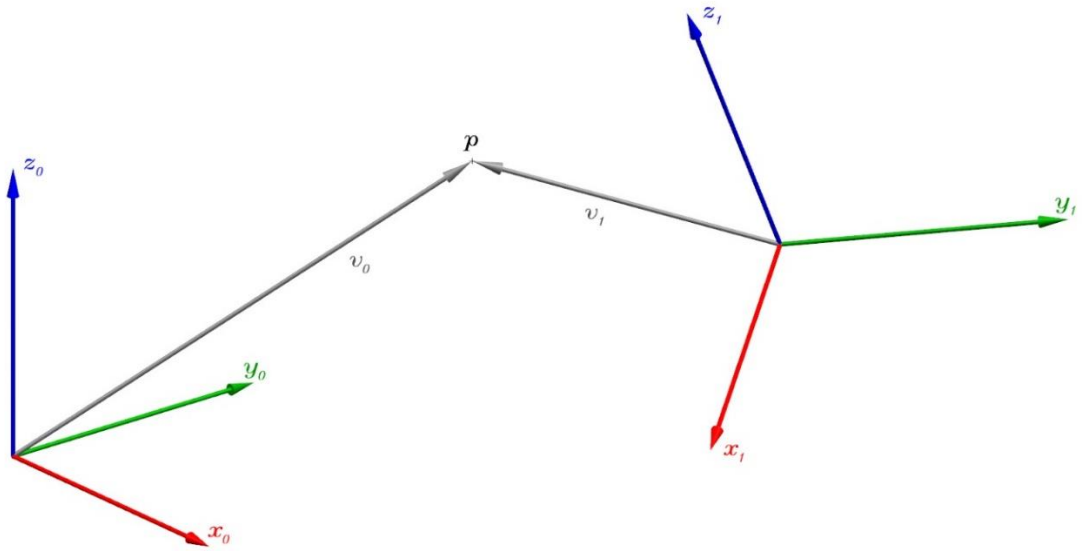

Figure 2.1 : Representation of the same point in different frames.

Frames directly give components of a vector in XYZ directions. However, it must be noted that we are using free vectors which means it can be translated through space freely. Its starting point is not determined unless the frame is specified. Hence, the only way to make free vectors meaningful is the determining frame that we want to represent.

The establishment of these frames is an important issue in robotics. Frames can be inertial or moving; for instance, one can assign a frame to the robot base and another frame to the end-effector. Their orientation would differ eventually and this causes an obstruction for adding or subtraction of vectors because basis vectors of represented frames are also different. Thus, it is required to define the orientation relationship between frames. An example is shown in **Figure 2.2**.

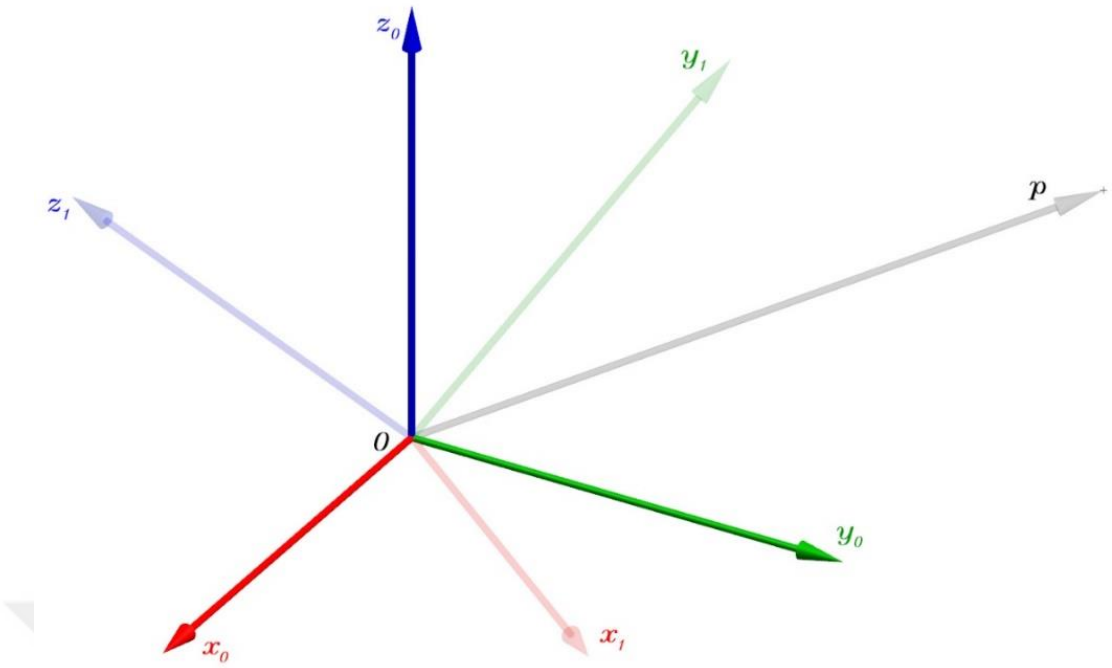

Figure 2.2 : Orientation difference between frames.

Frames are nothing but three orthogonal unit vectors that are possible to be represented in other frames. We think in the same manner of free vectors and coincide origins of different frames in order to represent orientation relation, regarding their origin may be at different locations. We obtain three vectors that contain three components. By vertically concatenating them, we get the 3x3 matrix called *rotation matrix*, $\boldsymbol{R}$, as follows.

$$[\boldsymbol{R}_1]_0 = \begin{bmatrix} \boldsymbol{x}_1 \cdot \boldsymbol{x}_0 & \boldsymbol{y}_1 \cdot \boldsymbol{x}_0 & \boldsymbol{z}_1 \cdot \boldsymbol{x}_0 \\ \boldsymbol{x}_1 \cdot \boldsymbol{y}_0 & \boldsymbol{y}_1 \cdot \boldsymbol{y}_0 & \boldsymbol{z}_1 \cdot \boldsymbol{y}_0 \\ \boldsymbol{x}_1 \cdot \boldsymbol{z}_0 & \boldsymbol{y}_1 \cdot \boldsymbol{z}_0 & \boldsymbol{z}_1 \cdot \boldsymbol{z}_0 \end{bmatrix} \tag{2.2}$$

We have obtained orientation information of a frame by taking direction cosines of its basis vectors. To represent the vector $\boldsymbol{p}$ in **Figure 2.2** with respect to frame zero, which is known for frame one, we use this rotation matrix:

$$[\boldsymbol{p}]_1 = [\, u \;\; v \;\; w \,]^T \tag{2.3}$$

$$[\boldsymbol{p}]_1 = u\boldsymbol{x}_1 + v\boldsymbol{y}_1 + w\boldsymbol{z}_1 \tag{2.4}$$

$$[\boldsymbol{p}]_0 = \begin{bmatrix} [\boldsymbol{p}]_1 \cdot \boldsymbol{x}_0 \\ [\boldsymbol{p}]_1 \cdot \boldsymbol{y}_0 \\ [\boldsymbol{p}]_1 \cdot \boldsymbol{z}_0 \end{bmatrix} \tag{2.5}$$

$$[\boldsymbol{p}]_0 = \begin{bmatrix} u\boldsymbol{x}_1 \cdot \boldsymbol{x}_0 + v\boldsymbol{y}_1 \cdot \boldsymbol{x}_0 + w\boldsymbol{z}_1 \cdot \boldsymbol{x}_0 \\ u\boldsymbol{x}_1 \cdot \boldsymbol{y}_0 + v\boldsymbol{y}_1 \cdot \boldsymbol{y}_0 + w\boldsymbol{z}_1 \cdot \boldsymbol{y}_0 \\ u\boldsymbol{x}_1 \cdot \boldsymbol{z}_0 + v\boldsymbol{y}_1 \cdot \boldsymbol{z}_0 + w\boldsymbol{z}_1 \cdot \boldsymbol{z}_0 \end{bmatrix} \tag{2.6}$$

$$[\boldsymbol{p}]_0 = \begin{bmatrix} \boldsymbol{x}_1 \cdot \boldsymbol{x}_0 & \boldsymbol{y}_1 \cdot \boldsymbol{x}_0 & \boldsymbol{z}_1 \cdot \boldsymbol{x}_0 \\ \boldsymbol{x}_1 \cdot \boldsymbol{y}_0 & \boldsymbol{y}_1 \cdot \boldsymbol{y}_0 & \boldsymbol{z}_1 \cdot \boldsymbol{y}_0 \\ \boldsymbol{x}_1 \cdot \boldsymbol{z}_0 & \boldsymbol{y}_1 \cdot \boldsymbol{z}_0 & \boldsymbol{z}_1 \cdot \boldsymbol{z}_0 \end{bmatrix} \begin{bmatrix} u \\ v \\ w \end{bmatrix} \tag{2.7}$$

$$[\boldsymbol{p}]_0 = [\boldsymbol{R}_1]_0 [\boldsymbol{p}]_1 \tag{2.8}$$

Where u, v, and w are arbitrary values.

The rotation matrix is an orthogonal matrix which satisfies the following equation:

$$\boldsymbol{R}^T = \boldsymbol{R}^{-1} \tag{2.9}$$

Rotation term describes motion in the meaning, however, we are using it for the static orientation relationship between a frame with respect to a known frame. Still, the rotation matrix can also be used for rotating vectors in the frame they have been represented. It is actually the same definition with the representation of a vector whose frame is already rotated. An illustration is shown in **Figure 2.3**.

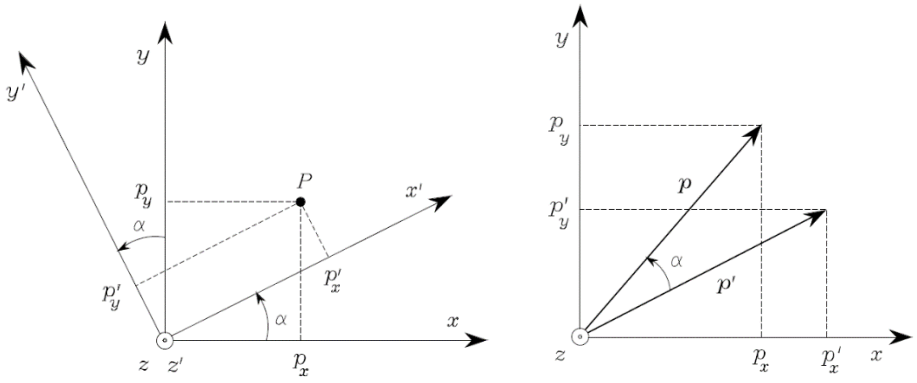

Figure 2.3 : Illustration for rotated frame vector and vector rotating in its frame [20].

$$\boldsymbol{p} = \boldsymbol{R}(\alpha, \boldsymbol{z})\boldsymbol{p}' \tag{2.10}$$

The rotation matrix is a linear operator for rotating vectors and it is a very fast computation method because it contains 9 parameters in the memory for 3 independent parameters. Nevertheless, it is not the only one to represent rotation. There are other useful methods for defining rotation with fewer parameters.

Before investigating rotation representations, we require to express another two matrix operators called line and plane projectors. Projecting vectors on lines and planes are going to be used frequently in the next chapters. Those projector matrices assist us to get feasible and infeasible directions of a movement vector. Firstly, we need to define a line in a frame. To do that, the easiest method is using a unit vector on the line. The direction of the vector is irrelevant for projector matrices. It is sufficient for the vector to stay on that line. Let us represent this unit vector as $\boldsymbol{e}$ and then, we define the line projector and the plane projector respectively as follows.

$$\boldsymbol{P}_L = \boldsymbol{e}\boldsymbol{e}^T \tag{2.11}$$

$$\boldsymbol{P}_P = \boldsymbol{I} - \boldsymbol{e}\boldsymbol{e}^T \tag{2.12}$$

These projector matrices decompose any vector on the specified line and the plane which is perpendicular to the line. Let us assume, arbitrary free vector $\boldsymbol{p}$ consists of two decomposed vectors on line and plane:

$$\boldsymbol{p} = \boldsymbol{p}_L + \boldsymbol{p}_P \tag{2.13}$$

To obtain these sub-vectors, we are going to use projectors above:

$$\boldsymbol{p}_L = \boldsymbol{P}_L\, \boldsymbol{p} = \boldsymbol{e}\boldsymbol{e}^T\boldsymbol{p} \tag{2.14}$$

$$\boldsymbol{p}_P = \boldsymbol{P}_P\, \boldsymbol{p} = (\boldsymbol{I} - \boldsymbol{e}\boldsymbol{e}^T)\boldsymbol{p} \tag{2.15}$$

The illustration is given in **Figure 2.4**.

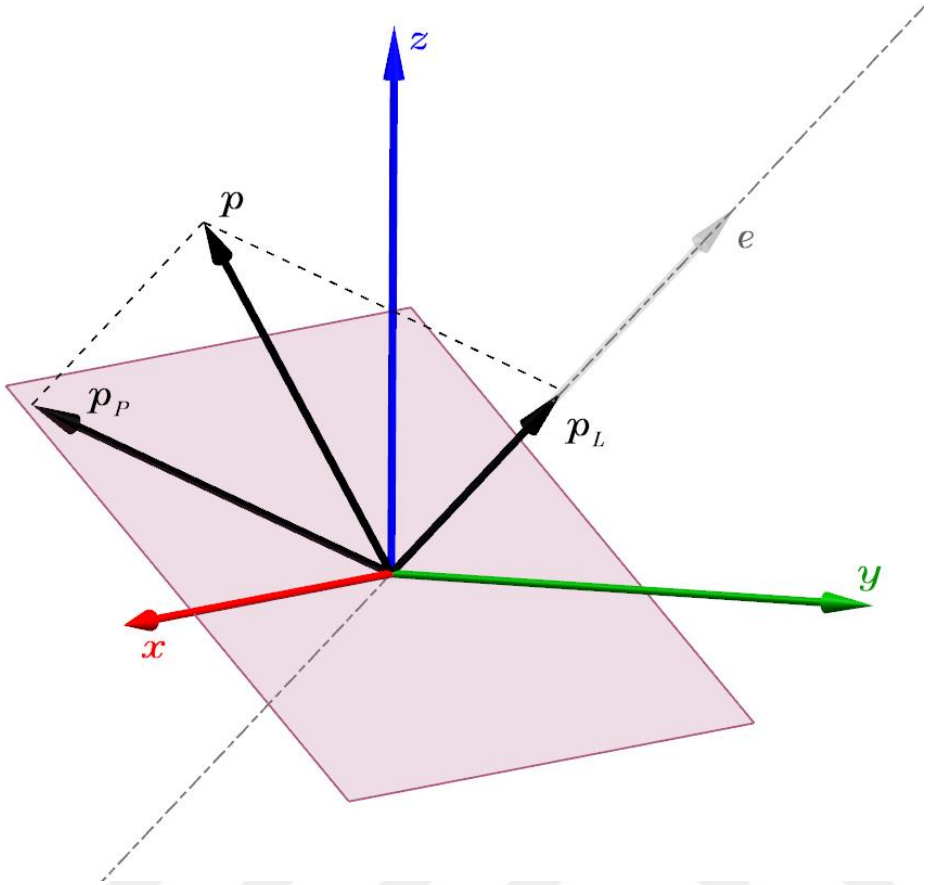

Figure 2.4 : Projecting a vector onto a line and a plane.

2.2 Rotation Representations

2.2.1 Euler-Rodrigues formula and invariance concept

In the year 1775, Euler proved that any rotation in three-dimensional space can be represented with a specified axis and angle [22]. By right-hand convention, the axis can be thought of as a unit vector, $\boldsymbol{e}$, and finite rotation displacement as an angle, θ, which can be seen in **Figure 2.5**. The study will be done here is mostly taken from J. Angeles [40].

For a finite rotation, unit vector $\boldsymbol{e}$, aligned to the rotation axis, would not be affected by the rotation itself and stay invariant:

$$\boldsymbol{e} = \boldsymbol{R}\boldsymbol{e} \tag{2.16}$$

By considering **Figure 2.6**, rotation can be specified in a geometric sense for the two-dimensional case:

$$\overrightarrow{QP'} = (\cos\theta)\overrightarrow{QP} + (\sin\theta)\overrightarrow{QP''} \tag{2.17}$$

Where:

$$\overrightarrow{QP} = (\boldsymbol{I} - \boldsymbol{e}\boldsymbol{e}^T)\boldsymbol{p} \tag{2.18}$$

$$\overrightarrow{QP''} = \boldsymbol{e} \times \boldsymbol{p} \equiv \boldsymbol{E}\boldsymbol{p} \tag{2.19}$$

Here, we are using $\boldsymbol{I}$ symbol for 3x3 identity matrix and $\boldsymbol{E}$ for the cross-product matrix of vector $\boldsymbol{e}$. Thereafter, we can define 3D rotated $\boldsymbol{p}'$ as follows:

$$\boldsymbol{p}' = (\boldsymbol{e}\boldsymbol{e}^T)\boldsymbol{p} + \overrightarrow{QP'} \tag{2.20}$$

$$\boldsymbol{p}' = \boldsymbol{e}\boldsymbol{e}^T\boldsymbol{p} + \cos\theta\,(\boldsymbol{I} - \boldsymbol{e}\boldsymbol{e}^T)\boldsymbol{p} + \sin\theta\,\boldsymbol{E}\boldsymbol{p} \tag{2.21}$$

$$\boldsymbol{p}' = [\,\boldsymbol{e}\boldsymbol{e}^T + \cos\theta\,(\boldsymbol{I} - \boldsymbol{e}\boldsymbol{e}^T) + \sin\theta\,\boldsymbol{E}]\boldsymbol{p} \tag{2.22}$$

$$\boldsymbol{p}' = \boldsymbol{R}(\theta, \boldsymbol{e})\boldsymbol{p} \tag{2.23}$$

We have obtained a linear relationship between initial and rotated vector as we obtained from direction cosines of frame representation. Besides, we have shown we can create a rotation matrix by using four parameters, three for unit rotation vector and one for the angle.

$$\boldsymbol{R}(\theta, \boldsymbol{e}) = \boldsymbol{e}\boldsymbol{e}^T + \cos\theta\,(\boldsymbol{I} - \boldsymbol{e}\boldsymbol{e}^T) + \sin\theta\,\boldsymbol{E} \tag{2.24}$$

These parameters, $\boldsymbol{e}$, and θ are called *natural invariants* and form following rotation matrix.

$$\boldsymbol{e} = [e_1\ e_2\ e_3]^T \tag{2.25}$$

$$\boldsymbol{R}(\theta, \boldsymbol{e}) = \begin{bmatrix} c_\theta + e_1^2 v_\theta & e_1 e_2 v_\theta - e_3 s_\theta & e_1 e_3 v_\theta + e_2 s_\theta \\ e_1 e_2 v_\theta + e_3 s_\theta & c_\theta + e_2^2 v_\theta & e_2 e_3 v_\theta - e_1 s_\theta \\ e_1 e_3 v_\theta - e_2 s_\theta & e_2 e_3 v_\theta + e_1 s_\theta & c_\theta + e_3^2 v_\theta \end{bmatrix} \tag{2.26}$$

Where $c_\theta = \cos\theta$, $s_\theta = \sin\theta$, $v_\theta = (1 - \cos\theta)$. By using this formulation, we can create any arbitrary orientation representation by given $\boldsymbol{e}$ and θ.

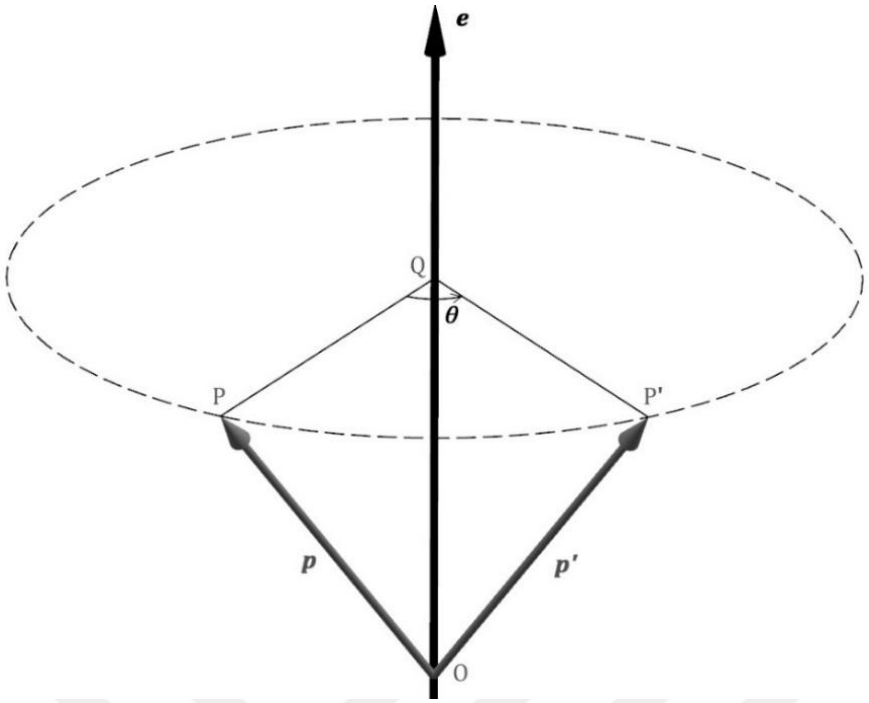

Figure 2.5 : Rotating a 3D vector through an axis.

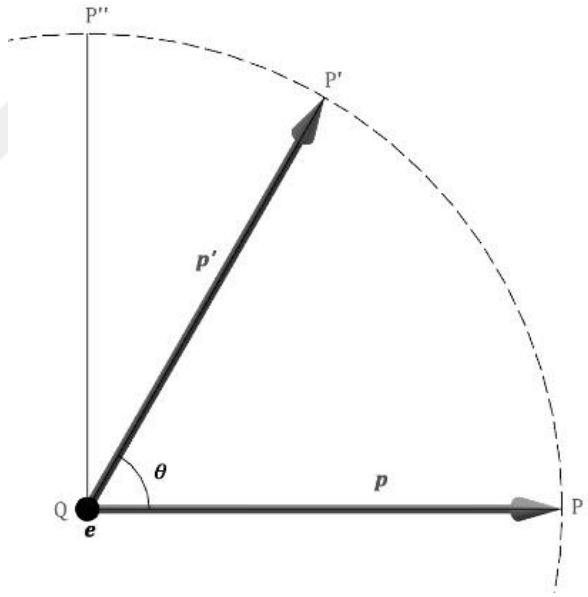

Figure 2.6 : Rotation of the 3D vector, top view.

2.2.1.1 Linear invariants of a matrix

If arbitrary 3x3 matrix **A** is a linear transformation of arbitrary vector $\boldsymbol{u} \in SE(3)$, resulting vector is an image of $\boldsymbol{u}$, namely, $\boldsymbol{v}$:

$$\boldsymbol{v} = \boldsymbol{A}.\boldsymbol{u} \tag{2.27}$$

$$\boldsymbol{A} = \begin{bmatrix} a_{11} & a_{12} & a_{13} \\ a_{21} & a_{22} & a_{23} \\ a_{31} & a_{32} & a_{33} \end{bmatrix} \tag{2.28}$$

We can counterpart symmetric and skew-symmetric part of the matrix as follows which is called *Cartesian decomposition*.

$$\boldsymbol{A_s} = \frac{1}{2}(\boldsymbol{A} + \boldsymbol{A}^T), \qquad \boldsymbol{A_{ss}} = \frac{1}{2}(\boldsymbol{A} - \boldsymbol{A}^T) \tag{2.29}$$

Where A_s and A_{ss} are denoted for symmetric and skew-symmetric matrices respectively.

It is possible to introduce two functions which are called vector of a matrix, $vect(*)$, and the trace of a matrix, $tr(*)$. Trace is only related to the symmetric part of the matrix while the vector of a matrix is only related to skew-symmetric as shown below.

$$\text{vect}(\boldsymbol{A}) \equiv \text{vect}(\boldsymbol{A_{ss}}) \equiv \boldsymbol{a} \equiv \frac{1}{2}\begin{bmatrix} a_{32} - a_{23} \\ a_{13} - a_{31} \\ a_{21} - a_{12} \end{bmatrix} \tag{2.30}$$

$$\text{tr}(\boldsymbol{A}) \equiv \text{tr}(\boldsymbol{A_s}) \equiv a_{11} + a_{22} + a_{33} \tag{2.31}$$

We can also use the skew-symmetric part of a matrix for replacing cross-product vector operation.

$$\boldsymbol{a} \times \boldsymbol{v} \equiv \boldsymbol{A_{ss}}.\boldsymbol{v} \tag{2.32}$$

2.2.1.2 Linear invariants of rotation

If we apply the concept above through the rotation matrix, we are able to obtain *linear invariants* of rotation.

Symmetric part of the rotation matrix in Eq. (2.24) would be:

$$[\boldsymbol{R}(\theta, \boldsymbol{e})]_s = \boldsymbol{e}\boldsymbol{e}^T + \cos\theta\,(\boldsymbol{I} - \boldsymbol{e}\boldsymbol{e}^T) \tag{2.33}$$

And skew-symmetric part of the rotation matrix in Eq. (2.24) would be:

$$[\boldsymbol{R}(\theta, \boldsymbol{e})]_{ss} = \sin\theta\,\boldsymbol{E} \tag{2.34}$$

Henceforth, we can use matrix functions separately which we have introduced:

$$\text{vect}(\boldsymbol{R}(\theta, \boldsymbol{e})) = \sin\theta\, \boldsymbol{e} \tag{2.35}$$

$$\text{tr}\big(\boldsymbol{R}(\theta, \boldsymbol{e})\big) = \text{tr}(\boldsymbol{e}\boldsymbol{e}^T) + \cos\theta * \text{tr}(\boldsymbol{I} - \boldsymbol{e}\boldsymbol{e}^T) \tag{2.36}$$

$$= \boldsymbol{e}^T\boldsymbol{e} + \cos\theta\,(3 - \boldsymbol{e}^T\boldsymbol{e}) \tag{2.37}$$

By using fact that $\boldsymbol{e}$ is a unit vector and $\boldsymbol{e}^T\boldsymbol{e}$ is equivalent to its dot product with itself, we obtain:

$$\text{tr}\big(\boldsymbol{R}(\theta, \boldsymbol{e})\big) = 1 + 2\cos\theta \tag{2.38}$$

This representation allows us to obtain unit rotation vector $\boldsymbol{e}$ and displacement angle θ, by using linear functions namely, *vect* and *tr*, from any rotation matrix that parameters are unknown. Hence, one is able to represent orientation between two frames with the following 4x1 vector, $\bar{\boldsymbol{\lambda}}$.

$$\bar{\boldsymbol{\lambda}} = \begin{bmatrix}\lambda_0 \\ \boldsymbol{\lambda}\end{bmatrix} = \begin{bmatrix}\cos\theta \\ \sin\theta\, \boldsymbol{e}\end{bmatrix} = \begin{bmatrix}\cos\theta \\ \sin\theta\, e_1 \\ \sin\theta\, e_2 \\ \sin\theta\, e_3\end{bmatrix} \tag{2.39}$$

This representation solves the uniqueness issue of natural invariants which we have eliminated with the convention using the right-hand rule in rotation. One can represent the rotation with any pair $(\theta, \boldsymbol{e})$ is also possible with the pair $(-\theta, -\boldsymbol{e})$ which causes uncertainty when taking inverse. Whereas, in linear invariants, they correspond the same $\boldsymbol{\lambda}$ and it solves this issue.

However, linear invariants suffer from a mathematical singularity when θ is at π and -π and causes error in results around those values. The reason is that we cannot obtain a skew-symmetric part of the rotation matrix when the sine function goes to zero. Hence, for a rotation matrix that is only numerically given, parameters unknown; linear invariants become a bit useless because the frame that is going to be represented can be placed around π of -π in an arbitrary axis for a frame.

A remedy for this problem was founded by changing angles into half-angles in the rotation matrix. The replacing parameters are called Euler-Rodrigues parameters and

have a special algebra called quaternion algebra which was invented by Sir William Rowan Hamilton.

2.2.2 Euler-Rodrigues parameters and quaternions

If we change parameters in trigonometric function as follows:

$$\cos\theta = 2\cos^2\frac{\theta}{2} - 1, \qquad \sin\theta = 2\sin\frac{\theta}{2}\cos\frac{\theta}{2} \tag{2.40}$$

We can obtain Eq. (2.24) in form of half-angles.

$$\boldsymbol{R}\left(\frac{\theta}{2}, \boldsymbol{e}\right) = 2\sin^2\frac{\theta}{2}\boldsymbol{e}\boldsymbol{e}^T + \left(2\cos^2\frac{\theta}{2} - 1\right)\boldsymbol{I} + 2\cos\frac{\theta}{2}\sin\frac{\theta}{2}\boldsymbol{E} \tag{2.41}$$

Then, we describe Euler-Rodrigues parameters or *quaternions* as follows, a scalar and a vector quantity.

$$q_0 = \cos(\theta/2) \tag{2.42}$$

$$\boldsymbol{q} = \sin(\theta/2)\,\boldsymbol{e} = \begin{bmatrix} q_1 \\ q_2 \\ q_3 \end{bmatrix} = \begin{bmatrix} \sin(\theta/2)\,e_1 \\ \sin(\theta/2)\,e_2 \\ \sin(\theta/2)\,e_3 \end{bmatrix} \tag{2.43}$$

By replacing Eq. (2.41) with these parameters, we obtain a linear rotation operator as a function of Euler-Rodrigues parameters:

$$\boldsymbol{R}(q_0, \boldsymbol{q}) = 2\boldsymbol{q}\boldsymbol{q}^T + (2q_0^2 - 1)\boldsymbol{I} + 2q_0\boldsymbol{Q} \tag{2.44}$$

Where $\boldsymbol{Q}$ is the skew-symmetric matrix form of vector $\boldsymbol{q}$.

Furthermore, the rotation matrix can be formed with quaternions:

$$\boldsymbol{R}(q_0, \boldsymbol{q}) = \begin{bmatrix} 2(q_0^2 + q_1^2) - 1 & 2(q_1q_2 - q_0q_3) & 2(q_1q_3 + q_0q_2) \\ 2(q_1q_2 + q_0q_3) & 2(q_0^2 + q_2^2) - 1 & 2(q_2q_3 - q_0q_1) \\ 2(q_1q_3 - q_0q_2) & 2(q_2q_3 + q_0q_1) & 2(q_0^2 + q_3^2) - 1 \end{bmatrix} \tag{2.45}$$

This matrix formulation is very important because it is robust to mathematical singularities which we have encountered in linear invariance concept. Moreover, they also solve the uniqueness issue of natural invariants as linear invariants solve. However, there are several methods for obtaining quaternions from an unknown

rotation matrix. A survey of methods is written by Sarabandi and Thomas [23]. After the comparison, they stated that the most useful method is Cayley's method with computational robustness and speed. It would be described below.

An unknown rotation matrix is shown below:

$$\boldsymbol{R} = \begin{bmatrix} r_{11} & r_{12} & r_{13} \\ r_{21} & r_{22} & r_{23} \\ r_{31} & r_{32} & r_{33} \end{bmatrix} \tag{2.46}$$

If we equate this with the matrix in Eq. (2.45) and try to solve quaternions, we obtain 10 equations as follows:

$$f_{a0} = 4q_0^2 = 1 + r_{11} + r_{22} + r_{33} \tag{2.47}$$

$$f_{a1} = 4q_1^2 = 1 + r_{11} - r_{22} - r_{33} \tag{2.48}$$

$$f_{a2} = 4q_2^2 = 1 - r_{11} + r_{22} - r_{33} \tag{2.49}$$

$$f_{a3} = 4q_3^2 = 1 - r_{11} - r_{22} + r_{33} \tag{2.50}$$

$$f_{b1} = 4q_2q_3 = r_{23} + r_{32} \tag{2.51}$$

$$f_{b2} = 4q_1q_3 = r_{13} + r_{31} \tag{2.52}$$

$$f_{b3} = 4q_1q_2 = r_{12} + r_{21} \tag{2.53}$$

$$f_{c1} = 4q_0q_1 = r_{32} - r_{23} \tag{2.54}$$

$$f_{c2} = 4q_0q_2 = r_{13} - r_{31} \tag{2.55}$$

$$f_{c3} = 4q_0q_3 = r_{21} - r_{12} \tag{2.56}$$

Henceforth, Cayley's method suggests that obtaining quaternions from the rightmost hand side of equations, we are showing here them with equation symbols, f, for simplicity.

$$q_0 = \frac{1}{4}\sqrt{f_{a0}^2 + f_{c1}^2 + f_{c2}^2 + f_{c3}^2} \tag{2.57}$$

$$q_1 = \frac{1}{4}\text{sign}(f_{c1})\sqrt{f_{a1}^2 + f_{c1}^2 + f_{b2}^2 + f_{b3}^2} \tag{2.58}$$

$$q_2 = \frac{1}{4}\text{sign}(f_{c2})\sqrt{f_{a2}^2 + f_{b1}^2 + f_{c2}^2 + f_{b3}^2} \tag{2.59}$$

$$q_3 = \frac{1}{4}\text{sign}(f_{c3})\sqrt{f_{a3}^2 + f_{b1}^2 + f_{b2}^2 + f_{c3}^2} \tag{2.60}$$

By convention, we assume q_0 is non-negative. Also, sign function gives output 1for the non-negative inputs including zero and gives output -1 for negative inputs.

In consequence, we obtain the quaternions with non-linear functions namely; sign, square, and square-root; which causing more computation cost. However, the reason we are using quaternions is they have their own algebra for rotation comparison and some advantageous features.

2.2.2.1 Quaternion algebra

For the sake of the thesis, we will basically introduce the quaternion algebra here. We can define two arbitrary quaternions as follows, they can be thought of as two different frame orientations with respect to a known frame.

$$\bar{\boldsymbol{r}} = [r_0\ r_1\ r_2\ r_3]^T, \qquad \bar{\boldsymbol{p}} = [p_0\ p_1\ p_2\ p_3]^T. \tag{2.61}$$

- *The quaternion product* is given below. The result is also a quaternion, a 4x1 vector.

$$\bar{\boldsymbol{r}} \star \bar{\boldsymbol{p}} = \begin{bmatrix} r_0 p_0 - \boldsymbol{r} \cdot \boldsymbol{p} \\ r_0 \boldsymbol{p} + p_0 \boldsymbol{r} + \boldsymbol{r} \times \boldsymbol{p} \end{bmatrix}$$

$$= \begin{bmatrix} r_0 p_0 - r_1 p_1 - r_2 p_2 - r_3 p_3 \\ r_1 p_0 + r_0 p_1 + r_2 p_3 - r_3 p_2 \\ r_2 p_0 + r_0 p_2 + r_3 p_1 - r_1 p_3 \\ r_3 p_0 + r_0 p_3 + r_1 p_2 - r_2 p_1 \end{bmatrix}. \tag{2.62}$$

Quaternion product is non-commutative, $\bar{\boldsymbol{r}} \star \bar{\boldsymbol{p}} \neq \bar{\boldsymbol{p}} \star \bar{\boldsymbol{r}}$, as rotation matrices. It has an analogy with the successive rotations, e.g. one can possibly define $\boldsymbol{R}_1\boldsymbol{R}_2\boldsymbol{R}_3$with also $\overline{\boldsymbol{r}_1} \star \overline{\boldsymbol{r}_2} \star \overline{\boldsymbol{r}_3}$ too. Moreover, they are more useful for rotation comparison rather than rotation matrices because they require less computation and also use fewer parameters.

- *Pure Quaternion*, is defined when the scalar part of the quaternion is equal to zero.

$$\overline{\boldsymbol{r}_p} = \begin{bmatrix} 0 \\ \boldsymbol{r} \end{bmatrix} \tag{2.63}$$

- *Quaternion inverse,* for unit quaternions, is equivalent to its conjugate or itself with a negative vector.

$$\bar{\boldsymbol{r}} = \begin{bmatrix} r_0 \\ \boldsymbol{r} \end{bmatrix}, \qquad \bar{\boldsymbol{r}}^{-1} = \begin{bmatrix} r_0 \\ -\boldsymbol{r} \end{bmatrix} \tag{2.64}$$

- *Quaternion rotation* is provided by firstly defining a 3D vector in quaternion form and then, multiplying it (quaternion product) with the determined quaternion from left and with the quaternion inverse from the right. $\bar{\boldsymbol{p}}$ is a quaternion form of position vector $\boldsymbol{p}$. We have given a rotated $\boldsymbol{p}'$ vector with the rotation matrix approach. Now, we are going to show it can be also founded in the quaternion form.

$$\boldsymbol{p} = [p_1\ p_2\ p_3]^T \tag{2.65}$$

$$\bar{\boldsymbol{p}} = \begin{bmatrix} 0 \\ \boldsymbol{p} \end{bmatrix} \tag{2.66}$$

$$\bar{\boldsymbol{p}}' = \begin{bmatrix} 0 \\ \boldsymbol{p}' \end{bmatrix} = \bar{\boldsymbol{q}} \star \bar{\boldsymbol{p}} \star \bar{\boldsymbol{q}}^{-1} \tag{2.67}$$

$$\bar{\boldsymbol{p}}' = \begin{bmatrix} q_0 \\ \boldsymbol{q} \end{bmatrix} \star \begin{bmatrix} 0 \\ \boldsymbol{p} \end{bmatrix} \star \begin{bmatrix} q_0 \\ -\boldsymbol{q} \end{bmatrix} \tag{2.68}$$

$$\bar{\boldsymbol{p}}' = \begin{bmatrix} -\boldsymbol{q} \cdot \boldsymbol{p} \\ q_0\boldsymbol{p} + \boldsymbol{q} \times \boldsymbol{p} \end{bmatrix} \star \begin{bmatrix} q_0 \\ -\boldsymbol{q} \end{bmatrix} \tag{2.69}$$

$$\bar{\boldsymbol{p}}' = \begin{bmatrix} 0 \\ [2\boldsymbol{q}\boldsymbol{q}^T + (2q_0^2 - 1)\boldsymbol{I} + 2q_0\boldsymbol{Q}]\boldsymbol{p} \end{bmatrix} \tag{2.70}$$

This operation is computationally much costly than the rotation matrix. So, we will use rotation matrices to rotate vectors instead of quaternions.

2.2.2.2 Quaternion Visualization

Both linear invariants and quaternions satisfy the following condition.

$$q_0^2 + q_1^2 + q_2^2 + q_3^2 = \lambda_0^2 + \lambda_1^2 + \lambda_2^2 + \lambda_3^2 = 1 \tag{2.71}$$

Hence, both can be thought of as a four-dimensional vector that defines the position of a point moving on a unit sphere. If we are going to move from one orientation to another, there must be a path laying on the unit sphere that describes the minimum possible turn. Actually, that path would be provided by an axis that is perpendicular to both quaternion vectors. An illustration is given in **Figure 2.7**.

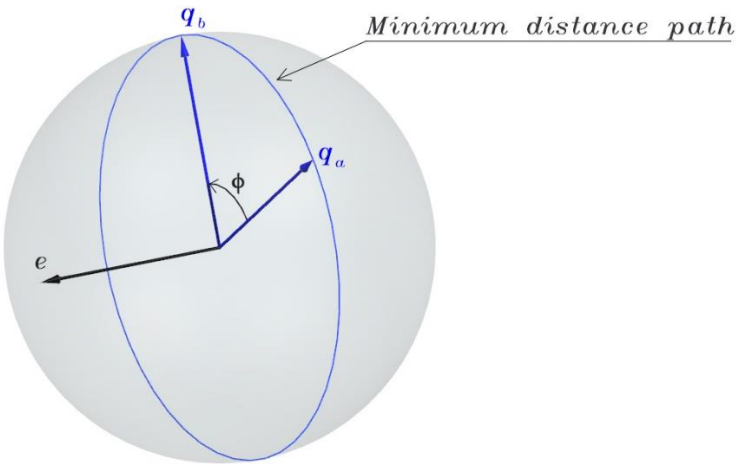

Figure 2.7 : Two quaternions on the unit sphere and minimum distance path between them.

2.2.2.3 Quaternion interpolation

It is possible to create the path between two quaternions on the unit sphere by using spherical linear interpolation or in short, *slerp*. It would be a parametric orientation curve which is evenly separated for every equal increment. We will use it in the path planning side for interpolation of an amount of given orientation information in order to generate it as a continuous curve that robot can track smoothly. Let us define the initial point as $\overline{\boldsymbol{q}}_a$, final point as $\overline{\boldsymbol{q}}_b$ and rotation quaternion between them as $\overline{\boldsymbol{q}}$:

$$\overline{\boldsymbol{q}}_b = \overline{\boldsymbol{q}}_a \star \overline{\boldsymbol{q}} \tag{2.72}$$

If we are wishing to find half-angle between them, the easiest way would be finding scalar part of the rotation quaternion:

$$\overline{\boldsymbol{q}} = \overline{\boldsymbol{q}}_a^{-1} \star \overline{\boldsymbol{q}}_b \tag{2.73}$$

$$\cos\phi = q_0 = q_{a0}q_{b0} + \boldsymbol{q}_a \cdot \boldsymbol{q}_b \tag{2.74}$$

$$\phi = \cos^{-1}(q_{a0}q_{b0} + \boldsymbol{q}_a \cdot \boldsymbol{q}_b) \tag{2.75}$$

After the half-angle, ϕ, is founded, slerp function can be defined as follows which is firstly given by Shoemake (1985) [24]:

$$\overline{\boldsymbol{q}}(u) = \frac{\sin(1-u)\phi}{\sin\phi}\overline{\boldsymbol{q}}_a + \frac{\sin u\phi}{\sin\phi}\overline{\boldsymbol{q}}_b \tag{2.76}$$

The interpolation parameter, *u*, takes any value from 0 to 1 and evenly spans path on the unit sphere. An illustration is given in **Figure 2.8**.

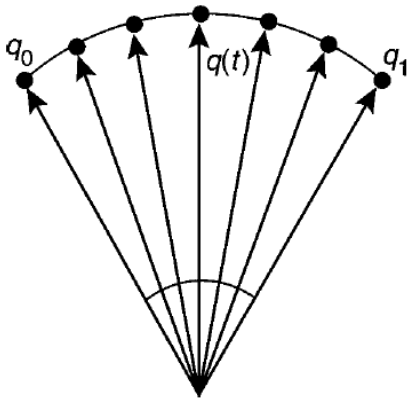

Figure 2.8 : SLERP spacing [25].

2.2.3 Successive rotations and Euler angles

By taking aside the discussed orientation representations, there is also another basic method for defining arbitrary orientations, *Euler Angles*. We have already stated that any orientation in Euclidean space can be represented by three parameters. If we select these parameters as a rotation through Cartesian coordinate axes that we are using, it would be possible to create representation by simply successively multiplying them. There is an illustration in **Figure 2.9** that shows three successive rotations. The key point here is in every stage one of the axes stays invariant.

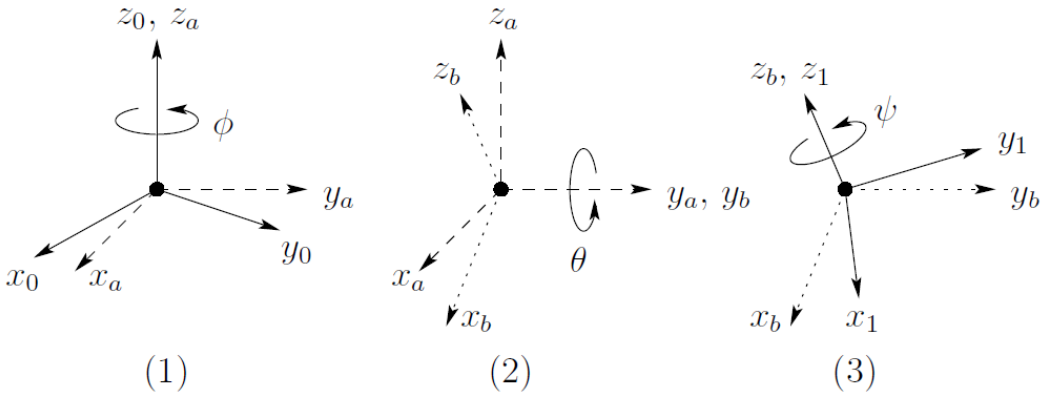

Figure 2.9 : Euler angles [26].

The generally preferred selection for the axes is ZYZ but any successively varying axes can be used. We will show ZYZ first and then others. Rotation matrices obtained from Eq. (2.26).

$$\boldsymbol{R}_{zyz} = \boldsymbol{R}(\phi, \boldsymbol{z})\boldsymbol{R}(\theta, \boldsymbol{y})\boldsymbol{R}(\psi, \boldsymbol{z})$$

$$= \begin{bmatrix} c_\phi & -s_\phi & 0 \\ s_\phi & c_\phi & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} c_\theta & 0 & s_\theta \\ 0 & 1 & 0 \\ -s_\theta & 0 & c_\theta \end{bmatrix} \begin{bmatrix} c_\psi & -s_\psi & 0 \\ s_\psi & c_\psi & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

$$= \begin{bmatrix} c_\phi c_\theta c_\psi - s_\phi s_\psi & -c_\phi c_\theta s_\psi - s_\phi c_\psi & c_\phi s_\theta \\ s_\phi c_\theta c_\psi - c_\phi s_\psi & -s_\phi c_\theta s_\psi + c_\phi c_\psi & s_\phi s_\theta \\ -s_\theta c_\psi & s_\theta s_\psi & c_\theta \end{bmatrix} \tag{2.77}$$

If we choose our axes as XYZ, they have a special name as roll-pitch-yaw and formulated as follows:

$$\boldsymbol{R}_{xyz} = \boldsymbol{R}(\phi, \boldsymbol{z})\boldsymbol{R}(\theta, \boldsymbol{y})\boldsymbol{R}(\psi, \boldsymbol{x})$$

$$= \begin{bmatrix} c_\phi c_\theta & -s_\phi c_\psi + c_\phi s_\theta s_\psi & s_\phi s_\psi + c_\phi s_\theta c_\psi \\ s_\phi c_\theta & c_\phi c_\psi + s_\phi s_\theta s_\psi & -c_\phi s_\psi + s_\phi s_\theta c_\psi \\ -s_\theta & c_\theta s_\psi & c_\theta c_\psi \end{bmatrix} \tag{2.78}$$

Moreover, it is possible to select XYX axes which can be used for wrist joint axes that are in XYX direction with respect to base frame:

$$\boldsymbol{R}_{xyx} = \boldsymbol{R}(\theta_1, \boldsymbol{x})\boldsymbol{R}(\theta_2, \boldsymbol{y})\boldsymbol{R}(\theta_3, \boldsymbol{x})$$

$$= \begin{bmatrix} c_2 & s_2 s_3 & s_2 c_3 \\ s_1 s_2 & -s_1 c_2 s_3 + c_1 c_3 & -s_1 c_2 c_3 - c_1 s_3 \\ -c_1 s_2 & c_1 c_2 s_3 + s_1 c_3 & c_1 c_2 c_3 - s_1 s_3 \end{bmatrix} \quad (2.79)$$

If we are up to solve these angles from given an arbitrary, numerical rotation matrix likely in Eq. (2.46), then we are going to encounter the following issues:

- If θ_2 is not equal to zero or π, and not even close to them:

Then, we can solve the angles with two possibilities, θ_2 can be positive or negative.

$$\theta_{1,p} = \text{atan2}(\text{r}_{21}, -r_{31}) \quad (2.80)$$

$$\theta_{2,p} = \text{atan2}\left(\sqrt{1 - \text{r}_{11}^2}, r_{11}\right) \quad (2.81)$$

$$\theta_{3,p} = \text{atan2}(\text{r}_{12}, -r_{13}) \quad (2.82)$$

$$\theta_{1,n} = \text{atan2}(-\text{r}_{21}, r_{31}) \quad (2.83)$$

$$\theta_{2,n} = \text{atan2}\left(-\sqrt{1 - \text{r}_{11}^2}, r_{11}\right) \quad (2.84)$$

$$\theta_{3,n} = \text{atan2}(-\text{r}_{12}, -r_{13}) \quad (2.85)$$

- If θ_2 is equal or close to zero:

In that case, we obtain the following rotation matrix.

$$\boldsymbol{R} = \begin{bmatrix} 1 & 0 & 0 \\ 0 & -s_1 s_3 + c_1 c_3 & -s_1 c_3 - c_1 s_3 \\ 0 & c_1 s_3 + s_1 c_3 & c_1 c_3 - s_1 s_3 \end{bmatrix}$$

$$= \begin{bmatrix} 1 & 0 & 0 \\ 0 & c_{13} & -s_{13} \\ 0 & s_{13} & c_{13} \end{bmatrix} \quad (2.86)$$

So, we can only solve the summation of $\theta_1$and θ_3 because the $r_{11} = 1$ ($\theta_2 = 0$). There are infinite solutions.

$$\theta_1 + \theta_3 = \text{atan2}(\text{r}_{32}, r_{33}) \quad (2.87)$$

- If θ_2 is equal or close to π:

Then, $r_{11} = -1$ ($\theta_2 = \pi$), we can only solve the difference of $\theta_1$and θ_3 and there are again infinite solutions.

$$\boldsymbol{R} = \begin{bmatrix} -1 & 0 & 0 \\ 0 & s_1 s_3 + c_1 c_3 & s_1 c_3 - c_1 s_3 \\ 0 & -c_1 s_3 + s_1 c_3 & -c_1 c_3 - s_1 s_3 \end{bmatrix}$$

$$= \begin{bmatrix} -1 & 0 & 0 \\ 0 & \cos(\theta_1 - \theta_3) & \sin(\theta_1 - \theta_3) \\ 0 & \sin(\theta_1 - \theta_3) & -\cos(\theta_1 - \theta_3) \end{bmatrix} \quad (2.88)$$

$$\theta_1 - \theta_3 = \text{atan2}(\mathrm{r}_{32}, r_{23}) \quad (2.89)$$

Consequently, Euler angles are problematic for taking inverse. Furthermore, they are useless for interpolation because portions of angles do not correspond to an intermediate orientation that laying on minimum rotation path, e.g. if we use half of the angles we would not end up with the middle of the orientation but in an irrelevant orientation. Nevertheless, they can be used for finding wrist joint parameters in IGM when the robot is not at the wrist singular configurations.

2.3 Robot Modelling

The robot model that is going to represented here would be a kinematically decoupled, ortho-parallel shoulder with a spherical wrist, manipulator which is the most commonly used kinematic design for industrial robots. A simplified 3D model can be seen in **Figure 2.10**.

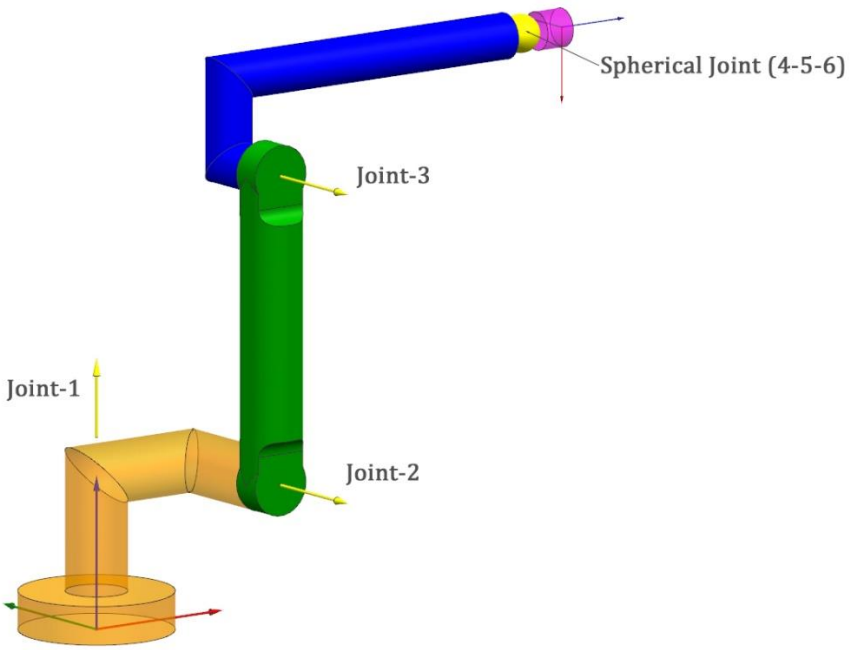

Figure 2.10 : Simplified decoupled industrial robot links and joints.

Manipulating position and orientation of the end-effector is the main objective of the robot to carry out. The motion of the end-effector is generated by the six successive joints that robots possess. To describe the motion of the end-effector, it requires at least two coordinate frames, one is at end-effector and the other one is at somewhere stationary. Robot base location is generally preferred for establishing a stationary coordinate frame which is a very meaningful reference for calculation of the end-effector motion. Still, the robot base can be movable e.g. it can be on a slider or on a mobile platform. In such cases, the robot base frame is not stationary and another inertial frame or world frame is needed to define the motion. In our study, we assume the robot base is not moving but only end-effector. Thus, the world frame and robot base frame are identical or coincident. Moreover, defining robot base and end-effector frames closes the kinematic chain of the model and it corresponds to specified values for every joint which will be discussed in this section.

For serial manipulators, kinematic models generally formed on the basis of Denavit-Hartenberg method which is well-known for everyone related to robotics. The main advantage of the DH method is the capability of defining any one degree-of-freedom motion between two links with minimum parameters (called DH parameters). However, it requires creating and assigning frames to each robot joint. Moreover, determining DH parameters for industrial robots can vary according to the convention

which makes it a user-defined phenomenon. Some improvements have been proposed to this method for better usage e.g. standard conventions or modified DH parameters. However, nonuniqueness leads to different frame assignments to result in the same end-effector motion in robot manipulator. Thus, from an industrial manipulator datasheet, one does not simply get DH parameters directly because there are multiple possibilities. Even if the parameters have been got, understanding assigned frames is a bit challenging.

Instead of the DH method, Brandstötter et al. suggest using link parameters for all of the manipulators owning ortho-parallel shoulder with spherical wrist [27]. By taking inspiration from their work, we defined 7 link parameters that are going to be used in both forward and inverse geometrical modeling in **Figure 2.11**. Furthermore, we are going to establish our models consisting of vectors with respect to the base frame.

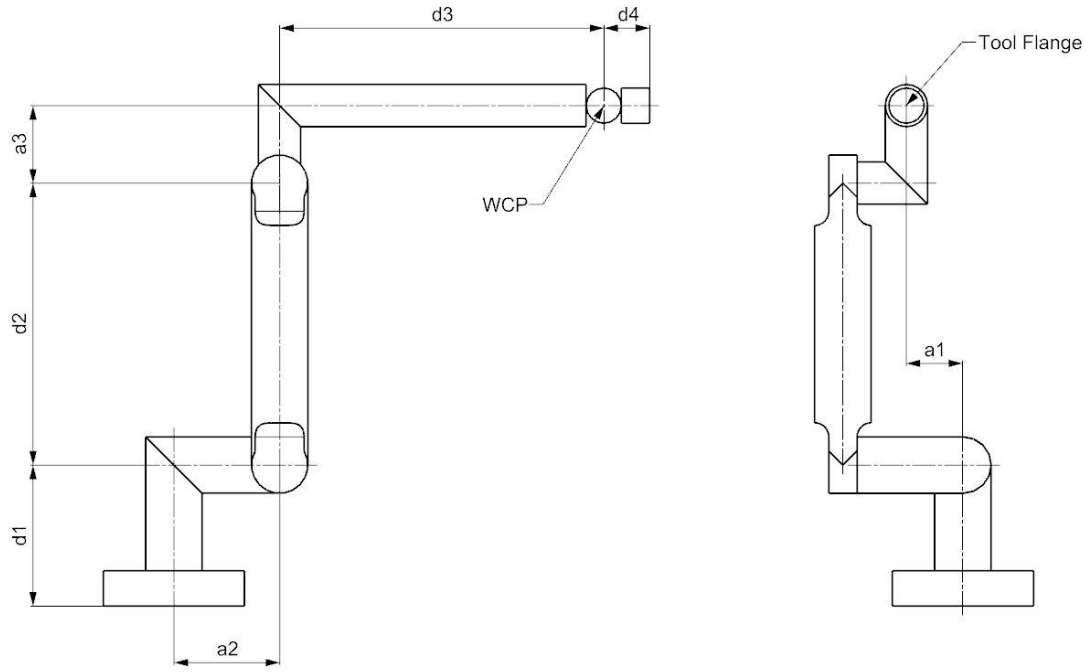

Figure 2.11 : Link parameters of industrial robots.

2.3.1 Forward geometrical model

As we discussed in the introduction, the forward geometrical model defines the end-effector pose for the given robot posture. Firstly, let us define joint parameters as follows.

$$\boldsymbol{q} = [q_1\ q_2\ q_3\ q_4\ q_5\ q_6]^T \tag{2.90}$$

Secondly, we are going to define all of the joint axes as a unit vector in the initial robot posture and with respect to the base frame as below.

$$\boldsymbol{e}_1 = [0\ 0\ 1]^T$$
$$\boldsymbol{e}_2 = [0\ 1\ 0]^T$$
$$\boldsymbol{e}_3 = [0\ 1\ 0]^T$$
$$\boldsymbol{e}_4 = [1\ 0\ 0]^T$$
$$\boldsymbol{e}_5 = [0\ 1\ 0]^T$$
$$\boldsymbol{e}_6 = [1\ 0\ 0]^T \tag{2.91}$$

Then, rotation matrices can be obtained by using Eq. (2.26). All of them are defined with respect to the base frame as follows.

$$\boldsymbol{R}_1 = \boldsymbol{R}(q_1, \boldsymbol{e}_1)$$
$$\boldsymbol{R}_2 = \boldsymbol{R}(q_2, \boldsymbol{e}_2)$$
$$\boldsymbol{R}_3 = \boldsymbol{R}(q_3, \boldsymbol{e}_3)$$
$$\boldsymbol{R}_4 = \boldsymbol{R}(q_4, \boldsymbol{e}_4)$$
$$\boldsymbol{R}_5 = \boldsymbol{R}(q_5, \boldsymbol{e}_5)$$
$$\boldsymbol{R}_6 = \boldsymbol{R}(q_6, \boldsymbol{e}_6) \tag{2.92}$$

Now, let us define successive position vectors between joint axes in the initial robot posture by regarding **Figure 2.11** in Eq. (2.93). It can also be thought as we have created frames, owning the same orientation with the base frame, at the center of each joint axis.

$$[\boldsymbol{o}_1]_0 = \begin{bmatrix} 0 \\ 0 \\ \mathrm{d1} \end{bmatrix}, \qquad [\boldsymbol{o}_2]_1 = \begin{bmatrix} \mathrm{a2} \\ 0 \\ 0 \end{bmatrix}, \qquad [\boldsymbol{o}_3]_2 = \begin{bmatrix} 0 \\ 0 \\ \mathrm{d2} \end{bmatrix},$$
$$[\boldsymbol{o}_4]_3 = \begin{bmatrix} \mathrm{d3} \\ -\mathrm{a1} \\ \mathrm{a3} \end{bmatrix}, \qquad [\boldsymbol{o}_5]_4 = [\boldsymbol{o}_6]_5 = \begin{bmatrix} 0 \\ 0 \\ 0 \end{bmatrix}. \tag{2.93}$$

Then, we are going to define position vectors of the joint centers as a function of joint parameters in the following equations. Successive matrix multiplication will be used here which can be thought of as defining vectors that are known for another frame or

alternatively, it also can be thought of as rotating free-vectors in the same frame with successive rotations.

$$\boldsymbol{p}_1 = [\boldsymbol{o}_1]_0 \tag{2.94}$$

$$\boldsymbol{p}_2 = [\boldsymbol{o}_1]_0 + \boldsymbol{R}_1[\boldsymbol{o}_2]_1 \tag{2.95}$$

$$\boldsymbol{p}_3 = [\boldsymbol{o}_1]_0 + \boldsymbol{R}_1[\boldsymbol{o}_2]_1 + \boldsymbol{R}_1\boldsymbol{R}_2[\boldsymbol{o}_3]_2 \tag{2.96}$$

$$\boldsymbol{p}_4 = [\boldsymbol{o}_1]_0 + \boldsymbol{R}_1[\boldsymbol{o}_2]_1 + \boldsymbol{R}_1\boldsymbol{R}_2[\boldsymbol{o}_3]_2 + \boldsymbol{R}_1\boldsymbol{R}_2\boldsymbol{R}_3[\boldsymbol{o}_4]_3 \tag{2.97}$$

$$\boldsymbol{p}_{WCP} = \boldsymbol{p}_4 = \boldsymbol{p}_5 = \boldsymbol{p}_6 \tag{2.98}$$

For decoupled manipulators, finding the position of WCP is sufficient to determine the end-effector position because wrist axes are coincident at that point. They do not contribute to position displacement by having zero distance between robot joint axes, however, their contribution to orientation affects the end-effector position by the distance between selected tool point and WCP. We call the control point of the end-effector as *tool center point*, TCP, and define its position as follows.

$$\boldsymbol{p}_{TCP} = \boldsymbol{p}_{WCP} + \boldsymbol{R}_1\boldsymbol{R}_2\boldsymbol{R}_3\boldsymbol{R}_4\boldsymbol{R}_5\boldsymbol{R}_6[\boldsymbol{o}_t]_6 \tag{2.99}$$

Where $\boldsymbol{o}_t$ is defined according to arbitrary tool position and last link distance, d4, in the orientation of base frame:

$$[\boldsymbol{o}_t]_6 = \begin{bmatrix} p_{tx} \\ p_{ty} \\ p_{tz} \end{bmatrix} + \begin{bmatrix} \mathrm{d4} \\ 0 \\ 0 \end{bmatrix} \tag{2.100}$$

Inherently, the desired TCP pose which is determined by the task is time-varying. The TCP frame defining the orientation of the end-effector with respect to the task can be assigned arbitrarily at both the path planning side or the robot side. In path planning, the general approach is orienting the z-axis in direction of task symmetry axis and placing the origin of the TCP frame to the task control point. Hence, for the robot side, we are going to refer orientation of the TCP frame with respect to base frame as an invariant rotation matrix, $\boldsymbol{R}_t$, for the initial robot configuration which is given likely in **Figure 2.10** and we define TCP frame orientation as follows.

$$\boldsymbol{R}_{TCP} = \boldsymbol{R}_1\boldsymbol{R}_2\boldsymbol{R}_3\boldsymbol{R}_4\boldsymbol{R}_5\boldsymbol{R}_6\boldsymbol{R}_t \tag{2.101}$$

The pose of the end-effector would be a function of this parametrization. We will definitely use quaternions while determining the task, whereas, on the robot side, rotation matrices may be advantageous for rotating vectors and finding inverses, etc. However, at the end or in one stage, we need to transform them into each other.

$$\boldsymbol{R}_{TCP} \rightarrow \overline{\boldsymbol{q}}_{TCP} \tag{2.102}$$

Where $\overline{\boldsymbol{q}}_{TCP}$ can be transformed from through Eq. (2.57-2.60) after creating a numerical value $\boldsymbol{R}_{TCP}$. Alternatively, it can be obtained by using quaternions instead of rotation matrices as in Eq. (2.42) and (2.43), then, multiplying them by using Eq. (2.62).

Finally, we define the pose of the EE, a 7x1 vector, by referring $\boldsymbol{s}$:

$$\boldsymbol{s} = \begin{bmatrix} \boldsymbol{p}_{TCP} \\ \overline{\boldsymbol{q}}_{TCP} \end{bmatrix} \tag{2.103}$$

On the other hand, by utilizing this formulation, $\boldsymbol{p}_{WCP}$ can be obtained easily in open form as below with few parameters while $\boldsymbol{p}_{TCP}$ requires much more term and complexity.

$$\boldsymbol{p}_{WCP} = \begin{bmatrix} c_1(\mathrm{a2} + s_2\mathrm{d2} + c_{23}\mathrm{d3} + s_{23}\mathrm{a3}) + s_1(\mathrm{a1}) \\ s_1(\mathrm{a2} + s_2\mathrm{d2} + c_{23}\mathrm{d3} + s_{23}\mathrm{a3}) + c_1(\mathrm{a1}) \\ \mathrm{d1} + c_2\mathrm{d2} + c_{23}\mathrm{a3} - s_{23}\mathrm{d3} \end{bmatrix} \tag{2.104}$$

Where joint parameters referred to as $c_i = \cos q_i$, $s_j = \sin q_j$ and $c_{ij} = \cos(q_i + q_j)$, $s_{ij} = \sin(q_i + q_j)$.

2.3.2 Inverse geometrical model

The model for finding joint parameters for any reachable, joint-limits allowing, EE position and orientation; is said to be inverse geometrical model, IGM, as follows.

$$\boldsymbol{s} \rightarrow \boldsymbol{q} \tag{2.105}$$

For a spatial six-successive axes robot, if there is not any coincidence with joint axes, it is possible to say that there would be 16 different robot posture correspond the same

end-effector pose, mathematically. To solve these parameters from the proposed forward model, there is not any known analytic solution. Yet, numerical methods exist, however, they can be troubled whenever singularities occur.

To preclude this issue, decoupled manipulators owning analytical solutions are preferred. IGM solutions of decoupled manipulators are described by the configuration it belongs to. There are three configuration types for the IGM solutions of decoupled manipulators: shoulder, elbow, and wrist. Each of them has two possible choices and the total possibility is eight. As the meaning of term decoupled, we are going to separate the shoulder and wrist sides of the manipulator and show solutions respectively. The solutions were firstly given by Donald Lee Pieper in 1968.

2.3.2.1 IGM of shoulder side

Firstly, we need to calculate the WCP position from the given TCP pose. For the orientation, we can transform quaternion to a rotation matrix by using Eq. (2.45) and position can be directly obtained as follows.

$$\boldsymbol{p}_{WCP} = \boldsymbol{p}_{TCP} - \boldsymbol{R}_{TCP}\boldsymbol{R}_t^T[\boldsymbol{o}_t]_6 \tag{2.106}$$

Then, we are going to obtain components of arbitrary WCP position vector with respect to two frames that are given in **Figure 2.12**:

$$\boldsymbol{p}_{WCP} = \begin{bmatrix} p_{x_0} \\ p_{y_0} \\ p_{z_0} \end{bmatrix} = \begin{bmatrix} c_1 p_{x_1} - s_1 p_{y_1} \\ c_1 p_{y_1} + s_1 p_{x_1} \\ p_{z_1} + d1 \end{bmatrix} \tag{2.107}$$

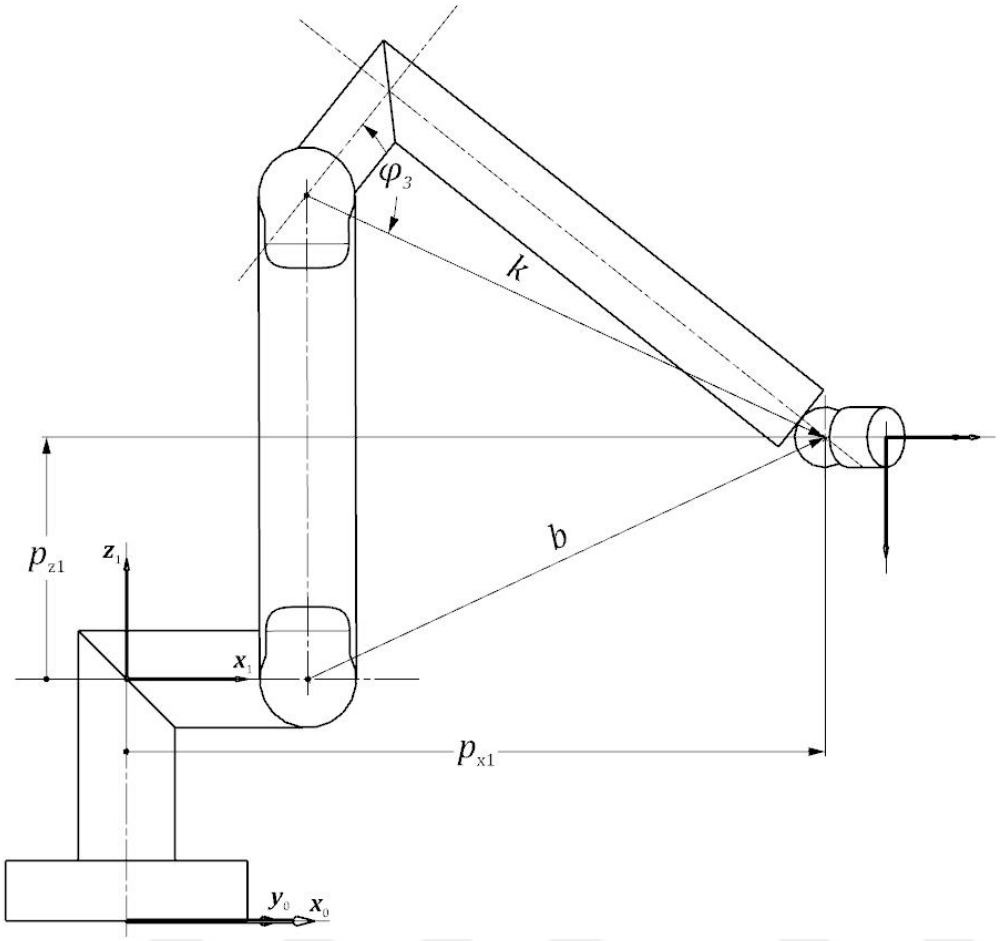

Figure 2.12 : Normal view with respect to joint-2 and joint-3.

Furthermore, we are going to define the following parameters for simplicity from link parameters:

$$k = \sqrt{(\mathrm{d3})^2 + (\mathrm{a3})^2} \tag{2.108}$$

$$\varphi_3 = \mathrm{atan2}(\mathrm{d3}, \mathrm{a3}) \tag{2.109}$$

$$b = \sqrt{(\mathrm{d2})^2 + \mathrm{k}^2} \tag{2.110}$$

These parameters are also shown in **Figure 2.12**. By regarding Eq. (2.104), the position of the WCP, with respect to the first frame, can be defined with these parameters:

$$[\boldsymbol{p}_{WCP}]_1 = \begin{bmatrix} p_{x_1} \\ p_{y_1} \\ p_{z_1} \end{bmatrix} = \begin{bmatrix} \sin(q_2)\,\mathrm{d2} + \sin(q_2 + q_3 + \varphi_3)\,k + \mathrm{a2} \\ -\mathrm{a1} \\ \cos(q_2)\mathrm{d2} + \cos(q_2 + q_3 + \varphi_3)\,k \end{bmatrix} \tag{2.111}$$

The trigonometric equations above can be solved for joint parameters when the coefficients are known, namely, $\boldsymbol{p}_{WCP}$ and link parameters. Assuming they are given

by the task and robot design, the solution would be 4 different combinations of joint parameters called configurations which can be seen in **Figure 2.13**. These are caused by firstly Eq. (2.107), having $q_1$defined by the 2 coefficients equations for $p_{x_0}$and p_{y_0}, and secondly Eq. (2.111), having q_2 and q_3 defined by the 2 equations for $p_{x_1}$and p_{z_1}.

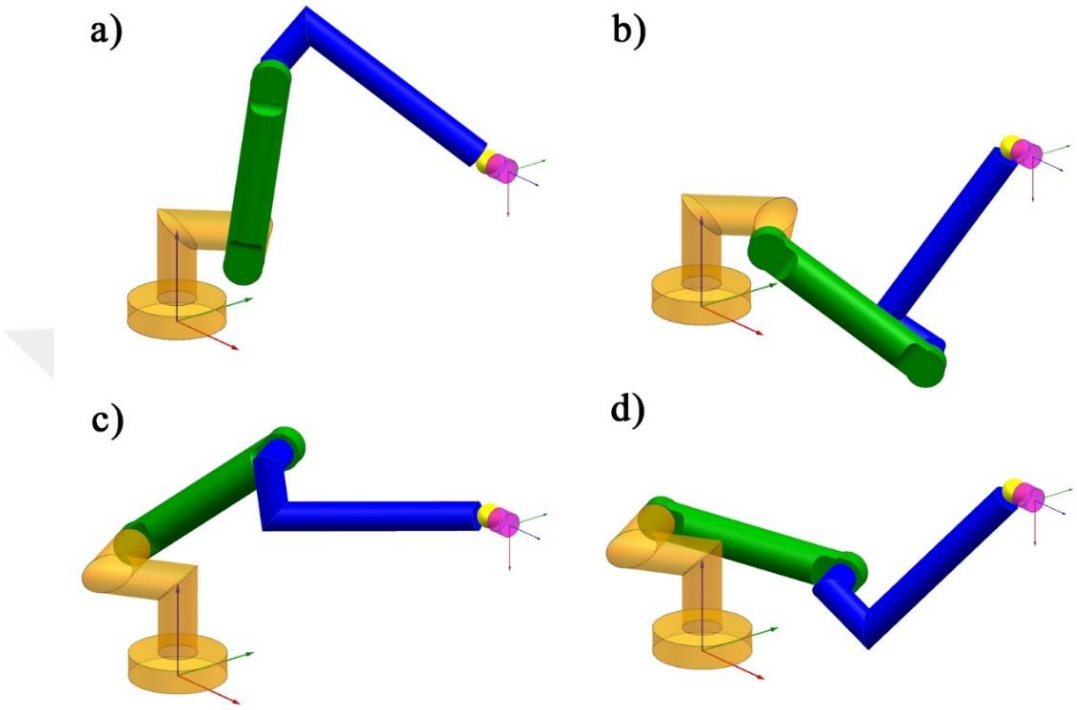

Figure 2.13 : Shoulder configurations: a) direct-shoulder elbow-up, b) direct-shoulder elbow-down, c) reverse-shoulder elbow-up, d) reverse-shoulder elbow-down.

All of these shoulder postures satisfy Eq. (2.107) and Eq. (2.101) and be solved by the following equations. We will create some intermediate parameters for ease of understanding.

Solving q_1:

By square summing p_{x_0} and p_{y_0} at Eq. (2.107), we can obtain $p_{x_1}^2 + p_{y_1}^2$ while $p_{y_1}$is just related to a link parameter a1. Then, p_{x_1} can be obtained as follows.

$$p_{x_1} = \sqrt{p_{x_0}^2 + p_{y_0}^2 - (\mathrm{a1})^2} \tag{2.112}$$

Hence, we are going to define φ_1 and then solve q_1 which can be seen in **Figure 2.14**.

$$\varphi_1 = \text{atan}\left({}^{\text{a1}}/_{p_{x_1}}\right) \tag{2.113}$$

$$q_{1,i} = \text{atan2}\left(p_{y_0}, p_{x_0}\right) + \varphi_1 \tag{2.114}$$

However, there is another possibility that robot posture may be reverse-shoulder. In that case, we are going to define q_1 as follows.

$$q_{1,ii} = q_{1,i} - 2\varphi_1 - \text{sign}\left(q_{1,i}\right)\pi \tag{2.115}$$

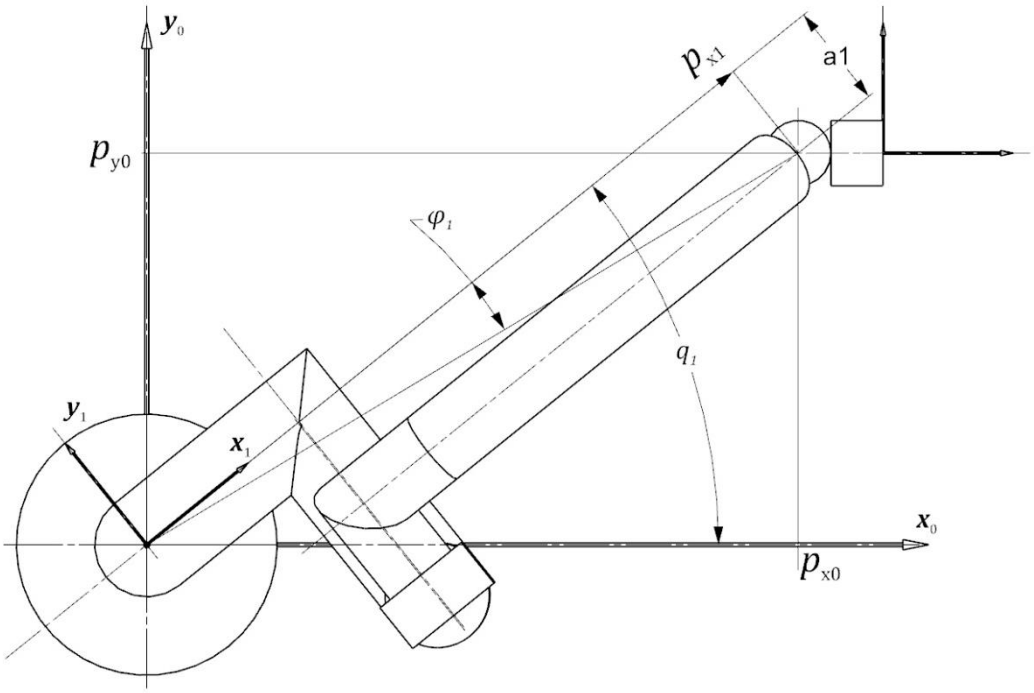

Figure 2.14 : Normal view with respect to joint-1.

Solving q_2 and q_3:

For solving q_2 and q_3, firstly, we need to define x-direction distance between axes joint-2 and WCP in the first frame. There are two possibilities according to joint-1 configuration namely, $q_{1,i}$ and $q_{1,ii}$, and the distances represented as follows.

$$n_1 = p_{x_1} - \text{a2} \tag{2.116}$$

$$n_2 = p_{x_1} + \text{a2} \tag{2.117}$$

We have already obtained p_{x_1} from the WCP. Moreover, the other significant direction is z for solving q_2 and q_3, we can directly obtain it as follows.

$$p_{z_1} = p_{z_0} - \mathrm{d1} \tag{2.118}$$

Hence, we get 2 two equations from Eq. (2.111):

$$n_1 = \sin(q_2)\,\mathrm{d2} + \sin(q_2 + q_3 + \varphi_3)\,k \tag{2.119}$$

$$p_{z_1} = \cos(q_2)\mathrm{d2} + \cos(q_2 + q_3 + \varphi_3)\,k \tag{2.120}$$

The solution for this system of equations is given by Khalil et al. (2002) [28], we are going to directly give them without describing detail.

$$\cos(q_3 + \varphi_3)_1 = \frac{n_1^2 + p_{z_1}^2 - b^2}{2k(\mathrm{d2})} \tag{2.121}$$

$$\sin(q_3 + \varphi_3)_1 = \sqrt{1 - \cos^2(q_3 + \varphi_3)_1} \tag{2.122}$$

$$q_{3,i} = \mathrm{atan2}(\sin(q_3 + \varphi_3)_1\,, \cos(q_3 + \varphi_3)_1) - \varphi_3 \tag{2.123}$$

Here, we have solved $q_{3,i}$ for elbow-up robot posture. However, when the robot is at elbow down posture, we need to make it negative and compensate φ_3.

$$q_{3,ii} = -q_{3,i} - 2\varphi_3 \tag{2.124}$$

After finding $q_{3,i}$ and $q_{3,ii}$ we are going to find the corresponding q_2 for each of them separately by using the formulation given by Khalil et al.

$$m_1 = \mathrm{d2} + k\cos(q_3 + \varphi_3)_1 \tag{2.125}$$

$$m_2 = k\sin(q_3 + \varphi_3)_1 \tag{2.126}$$

$$\sin(q_{2,i}) = \frac{m_1 n_1 - m_2 p_{z_1}}{b^2} \tag{2.127}$$

$$\cos(q_{2,i}) = \frac{m_1 p_{z_1} + m_2 n_1}{b^2} \tag{2.128}$$

$$\sin(q_{2,ii}) = \frac{m_1 n_1 + m_2 p_{z_1}}{b^2} \tag{2.129}$$

$$\cos(q_{2,ii}) = \frac{m_1 p_{z_1} - m_2 n_1}{b^2} \tag{2.130}$$

$$q_{2,i} = \text{atan2}\left(\sin\left(q_{2,i}\right), \cos\left(q_{2,i}\right)\right) \tag{2.131}$$

$$q_{2,ii} = \text{atan2}\left(\sin\left(q_{2,ii}\right), \cos\left(q_{2,ii}\right)\right) \tag{2.132}$$

We have finished the solution to $q_{2,i}$ and $q_{2,ii}$ for the robot postures direct-shoulder elbow-up and elbow-down. Still, there is also a reverse-shoulder robot posture and also two possibilities for the elbow. For q_3, it is possible to use the same formulation with only changing n_1 to n_2.

$$\cos(q_3 + \varphi_3)_2 = \frac{n_2^2 + p_{z_1}^2 - b^2}{2k(\text{d2})} \tag{2.133}$$

$$\sin(q_3 + \varphi_3)_2 = \sqrt{1 - \cos^2(q_3 + \varphi_3)} \tag{2.134}$$

$$q_{3,iii} = \text{atan2}(\sin(q_3 + \varphi_3)_2, \cos(q_3 + \varphi_3)) - \varphi_3 \tag{2.135}$$

$$q_{3,iv} = -q_{3,iii} - 2\varphi_3 \tag{2.136}$$

For q_2, we need to define new intermediate parameters for the reverse-shoulder q_3 values.

$$m_3 = \text{d2} + \cos(q_3 + \varphi_3)_2 \tag{2.137}$$

$$m_4 = \sin(q_3 + \varphi_3)_2 \tag{2.138}$$

$$\sin\left(q_{2,iii}\right) = \frac{-m_3 n_2 - m_4 p_{z_1}}{b^2} \tag{2.139}$$

$$\cos\left(q_{2,iii}\right) = \frac{m_3 p_{z_1} - m_4 n_2}{b^2} \tag{2.140}$$

$$\sin\left(q_{2,iv}\right) = \frac{-m_3 n_2 + m_4 p_{z_1}}{b^2} \tag{2.141}$$

$$\cos\left(q_{2,iv}\right) = \frac{m_3 p_{z_1} + m_4 n_2}{b^2} \tag{2.142}$$

$$q_{2,iii} = \text{atan2}\left(\sin\left(q_{2,iii}\right), \cos\left(q_{2,iii}\right)\right) \tag{2.143}$$

$$q_{2,iv} = \text{atan2}\left(\sin\left(q_{2,iv}\right), \cos\left(q_{2,iv}\right)\right) \tag{2.144}$$

We have finished all of the joint parameter solutions for the shoulders side of the decoupled robot. **Table 2.1** is given below that summarizes the proposed solutions with the configuration they belong to.

Robot Shoulder Configuration	Direct-Shoulder Elbow-Up	Direct-Shoulder Elbow-Down	Reverse-Shoulder Elbow-Up	Reverse-Shoulder Elbow-Down
Joint-1	$q_{1,i}$	$q_{1,i}$	$q_{1,ii}$	$q_{1,ii}$
Joint-2	$q_{2,i}$	$q_{2,ii}$	$q_{2,iii}$	$q_{2,iv}$
Joint-3	$q_{3,i}$	$q_{3,ii}$	$q_{3,iii}$	$q_{3,iv}$

Table 2.1 : Robot shoulder configurations and joint parameters.

2.3.2.2 IGM of wrist side

After the shoulder joint parameters are found, we can solve wrist joint parameters. Firstly, it is required to obtain the frame orientation as a consequence of the founded shoulder angles.

$$\boldsymbol{R}_{sh} = \boldsymbol{R}_1\boldsymbol{R}_2\boldsymbol{R}_3 \tag{2.145}$$

Then, we can found the orientation difference between the third link and the sixth link from the given TCP frame. This would correspond to a rotation matrix that wrist angles forms.

$$\boldsymbol{R}_{wr} = \boldsymbol{R}_4\boldsymbol{R}_5\boldsymbol{R}_6 = \boldsymbol{R}_3^T\boldsymbol{R}_2^T\boldsymbol{R}_1^T\boldsymbol{R}_{TCP}\boldsymbol{R}_t^T \tag{2.146}$$

For solving these successive rotation matrices, Euler angles are very useful in the view of calculation speed. Although, we will not use IGM at wrist singular configurations where Euler angles cannot be solved. Thus, there are two possibilities for the solution where the joint-5 may be positive or negative. Hence, for these two configurations, wrist angles can be found as follows.

$$\boldsymbol{R}_{wr} = \begin{bmatrix} r_{11} & r_{12} & r_{13} \\ r_{21} & r_{22} & r_{23} \\ r_{31} & r_{32} & r_{33} \end{bmatrix} \tag{2.147}$$

$$q_{4,p} = \text{atan2}(\text{r}_{21}, -r_{31}) \tag{2.148}$$

$$q_{5,p} = \text{atan2}\left(\sqrt{1 - \text{r}_{11}^2}, r_{11}\right) \tag{2.149}$$

$$q_{6,p} = \text{atan2}(\text{r}_{12}, -r_{13}) \tag{2.150}$$

$$q_{4,n} = \text{atan2}(-\text{r}_{21}, r_{31}) \tag{2.151}$$

$$q_{5,n} = \text{atan2}\left(-\sqrt{1 - \text{r}_{11}^2}, r_{11}\right) \tag{2.152}$$

$$q_{6,n} = \text{atan2}(-\text{r}_{12}, -r_{13}) \tag{2.153}$$

2.3.3 Velocity analysis

To begin the velocity analysis of a robot manipulator, we firstly discuss how to represent the velocity of a free body in three-dimensional space. It is required to separate the motion in types as translational and rotational because each can be thought of as independent with respect to a known frame. There are alternate representations such as screw motion with respect instantaneous lines but we are not interested in them.

For a finite translational motion, the position vector of a free body can be thought of as a time function starting from the initial state.

$$\boldsymbol{r}(t) = \boldsymbol{p}(t) + \boldsymbol{r}_0 \tag{2.154}$$

If the right-hand side of the equation above is differentiated, the translational velocity vector having components of Cartesian coordinate axes can be found which is natural and useful as described below.

$$\frac{d}{dt}\boldsymbol{p}(t) = \boldsymbol{v}(t) = \begin{bmatrix} \dot{p}_x(t) \\ \dot{p}_y(t) \\ \dot{p}_z(t) \end{bmatrix} \tag{2.155}$$

For the pure and finite rotational motion, let us define a body frame whose origin is coincident with the world frame as given in **Figure 2.15**.

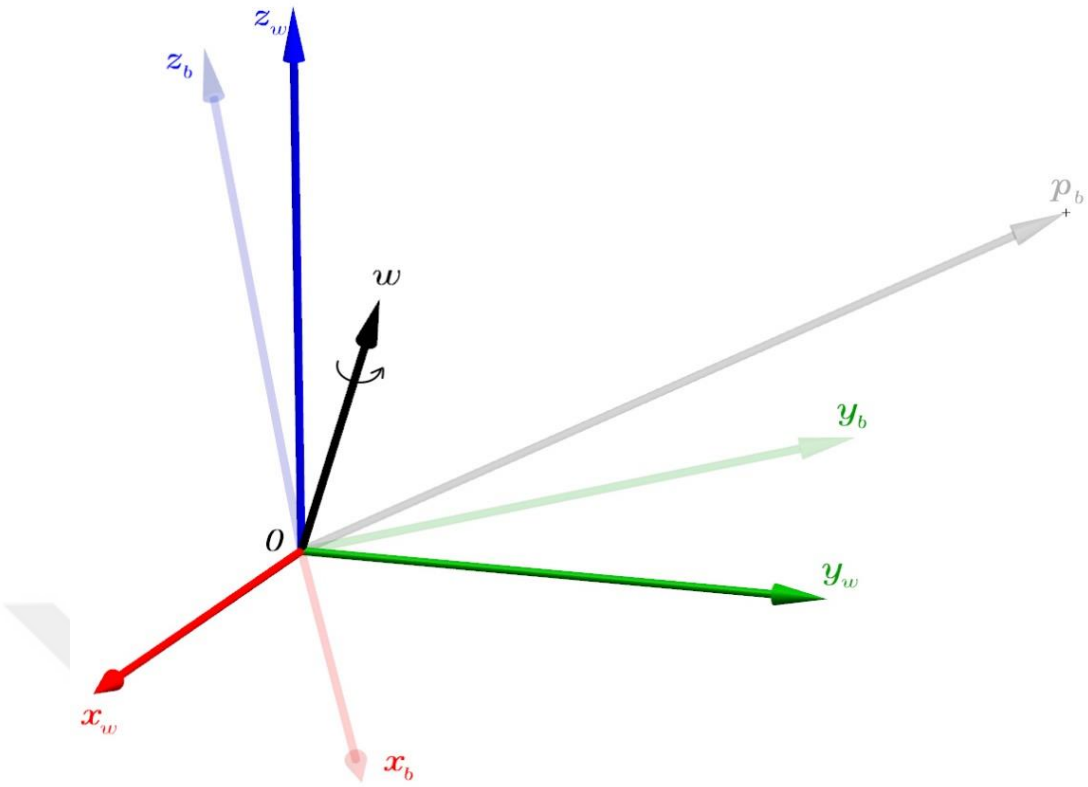

Figure 2.15 : Pure rotation.

Any determined point in the body frame having a rotational motion can be expressed with a time-varying rotation matrix that represents the position of the point with respect to the world frame as follows.

$$\boldsymbol{p}_w(t) = \boldsymbol{R}(t)\boldsymbol{p}_b \tag{2.156}$$

Where $\boldsymbol{p}_w$ is denoted position vector of the body point $\boldsymbol{p}_b$. We have assumed there is only rotational motion in the body frame. If we are willing to investigate the velocity of this body point, it is required to differentiate both sides.

$$\dot{\boldsymbol{p}}_w(t) = \dot{\boldsymbol{R}}(t)\boldsymbol{p}_b \tag{2.157}$$

$\boldsymbol{p}_b$ is not time-varying according to the body frame, however, it can also be expressed in the world frame by using Eq. (2.156):

$$\dot{\boldsymbol{p}}_w(t) = \dot{\boldsymbol{R}}(t)\boldsymbol{R}(t)^T\boldsymbol{p}_w(t) \tag{2.158}$$

This expression is important because we have found the velocity of the body point with respect to the world frame by using a coefficient matrix which we define as *instantaneous angular velocity matrix*:

$$\boldsymbol{\Omega}(\mathrm{t}) = \dot{\boldsymbol{R}}(t)\boldsymbol{R}(t)^T \tag{2.159}$$

We have used the orthogonality principle of the rotation matrix,

$$\boldsymbol{R}(t)\boldsymbol{R}(t)^T = \boldsymbol{I} \tag{2.160}$$

And its derivative tells us, instantaneous angular velocity matrix is skew-symmetric.

$$\dot{\boldsymbol{R}}(t)\boldsymbol{R}(t)^T + \boldsymbol{R}(t)\dot{\boldsymbol{R}}(t)^T = \boldsymbol{0} \tag{2.161}$$

That means we can also express the instantaneous angular velocity matrix as a vector by using vect function which we have discussed in subsection 2.2.

$$\mathrm{vect}\big(\boldsymbol{\Omega}(\mathrm{t})\big) = \boldsymbol{\omega}(t) \tag{2.162}$$

Hence, we can express derivative of any pure rotating vector in world frame by cross-production with time-varying *angular velocity vector*:

$$\frac{d}{dt}\boldsymbol{p_w} = \boldsymbol{\omega} \times \boldsymbol{p}_w \tag{2.163}$$

The fact is also valid for orthogonal unit vectors of the body frame:

$$\frac{d}{dt}[\boldsymbol{x}_b]_w = \boldsymbol{\omega} \times [\boldsymbol{x}_b]_w \tag{2.164}$$

$$\frac{d}{dt}[\boldsymbol{y}_b]_w = \boldsymbol{\omega} \times [\boldsymbol{y}_b]_w \tag{2.165}$$

$$\frac{d}{dt}[\boldsymbol{z}_b]_w = \boldsymbol{\omega} \times [\boldsymbol{z}_b]_w \tag{2.166}$$

Because of each term is time-dependent, we have eliminated time symbols for ease of articulation.

The angular velocity vector is fundamental for velocity analysis of rigid body motion, such that one can determine any finite instantaneous rotation by only using this vector. We use the term finite because it is parallel to the rotation axis that Euler's theorem proposed. In other words, rotational motion, starting from any orientation to another orientation, has been formulated by an invariant rotation axis, e, and time-varying angle, theta. In that way, the rotation matrix becomes a smooth function of time and

the angular velocity vector becomes parallel to the rotation axis. Let us show it by differentiated Euler-Rodrigues formula as Reza (2018) stated [39]:

$$\boldsymbol{R}(\theta, \boldsymbol{e})^T = \boldsymbol{e}\boldsymbol{e}^T + \cos\theta\,(\boldsymbol{I} - \boldsymbol{e}\boldsymbol{e}^T) - \sin\theta\,\boldsymbol{E} \tag{2.167}$$

$$\dot{\boldsymbol{R}}(\theta, \boldsymbol{e}) = \dot{\theta}\big(\cos\theta\,\boldsymbol{E} - \sin\theta\,(\boldsymbol{I} - \boldsymbol{e}\boldsymbol{e}^T)\big) \tag{2.168}$$

$$\boldsymbol{\Omega} = \lim_{\theta\to 0}\big[\dot{\boldsymbol{R}}(\theta, \boldsymbol{e})\boldsymbol{R}(\theta, \boldsymbol{e})^T\big] \tag{2.169}$$

$$\boldsymbol{\Omega} = \dot{\theta}\boldsymbol{E} \tag{2.170}$$

$$\boldsymbol{\omega} = \mathrm{vect}(\boldsymbol{\Omega}) = \dot{\theta}\boldsymbol{e} \tag{2.171}$$

It is also possible to obtain an angular velocity vector by using quaternions instead of a rotation matrix. Let us think in the same manner as the rotating position vector of the body frame and then taking its derivative. We require transforming vectors in pure quaternion form, 4x1 vectors.

$$\overline{\boldsymbol{p}}_w(t) = \overline{\boldsymbol{q}}(t) \star \overline{\boldsymbol{p}}_b \star \overline{\boldsymbol{q}}(t)^{-1} \tag{2.172}$$

$$\dot{\overline{\boldsymbol{p}}}_w(t) = \dot{\overline{\boldsymbol{q}}}(t) \star \overline{\boldsymbol{p}}_b \star \overline{\boldsymbol{q}}(t)^{-1} + \overline{\boldsymbol{q}}(t) \star \overline{\boldsymbol{p}}_b \star \dot{\overline{\boldsymbol{q}}}(t)^{-1} \tag{2.173}$$

$$\dot{\overline{\boldsymbol{p}}}_w(t) = \overline{\boldsymbol{q}}(t) \star \overline{\boldsymbol{q}}(t)^{-1} \star \overline{\boldsymbol{p}}_w(t) + \overline{\boldsymbol{p}}_w(t) \star \overline{\boldsymbol{q}}(t) \star \dot{\overline{\boldsymbol{q}}}(t)^{-1} \tag{2.174}$$

The equation above can be simplified, for proof, one can visit Graf (2007) [29]:

$$\dot{\overline{\boldsymbol{p}}}_w(t) = 2 * \dot{\overline{\boldsymbol{q}}}(t) \star \overline{\boldsymbol{q}}(t)^{-1} \star \overline{\boldsymbol{p}}_w(t) \tag{2.175}$$

We have already know cross production $\boldsymbol{\omega}$ with $\boldsymbol{p}_w$ gives us its derivative. Thus, we can equate the founded quaternion coefficient to $\boldsymbol{\omega}$, whereas, we need to transform $\boldsymbol{\omega}$ into pure quaternion form as follows.

$$\overline{\boldsymbol{\omega}}(t) = 2 * \dot{\overline{\boldsymbol{q}}}(t) \star \overline{\boldsymbol{q}}(t)^{-1} \tag{2.176}$$

The right-hand side of the equation is also a pure quaternion because:

$$\dot{q}_0 q_0 + \dot{q}_1 q_1 + \dot{q}_2 q_2 + \dot{q}_3 q_3 = 0 \tag{2.177}$$

For obtaining quaternion derivative we can reverse the equation as:

$$\dot{\overline{\boldsymbol{q}}}(t) = \frac{1}{2} * \overline{\boldsymbol{\omega}}(t) \star \overline{\boldsymbol{q}}(t) \tag{2.178}$$

Alternatively, one can use a newton method approach for quaternion derivative for the cases not requiring an exact solution and owning short computation time.

$$\dot{\overline{\boldsymbol{q}}}(t) \approx \frac{\overline{\boldsymbol{q}}(t + \Delta t) - \overline{\boldsymbol{q}}(t)}{\Delta t} \tag{2.179}$$

Furthermore, if the minimum path between two quaternions is going to be used, then we can solve the rotation angle and axis analytically. In order to do that, we are going to create a rotation quaternion by using two successive orientations in form of quaternions:

$$\overline{\boldsymbol{q}}_{rot} = \overline{\boldsymbol{q}}(t)^{-1} \star \overline{\boldsymbol{q}}(t + \Delta t) \tag{2.180}$$

Then, solve angle and axis:

$$\frac{\theta_{rot}}{2} = \cos^{-1}(q_{rot_0}) \tag{2.181}$$

$$\sin\frac{\theta_{rot}}{2} = \sqrt{1 - \left(q_{rot_0}\right)^2} \tag{2.182}$$

$$\boldsymbol{e}_{rot} = \boldsymbol{R}(q_0, \boldsymbol{q}) * \frac{1}{\sin\frac{\theta_{rot}}{2}} * \begin{bmatrix} q_{rot_1} \\ q_{rot_2} \\ q_{rot_3} \end{bmatrix} \tag{2.183}$$

After that, we can directly obtain $\boldsymbol{\omega}$ by using Eq. (2.171).

$$\boldsymbol{\omega}_{rot} = \frac{\theta_{rot}}{\Delta t} * \boldsymbol{e}_{rot} \tag{2.184}$$

2.3.3.1 The velocity of the rigid body

We have discussed the pure translational and pure rotational motion. Now, we are going to combine them and show the possibility of representing both with a 6x1 *twist* vector, $\boldsymbol{t}$. As given in **Figure 2.16**, let us define a freely moving body frame whose origin position is defined as $\boldsymbol{p}$ for a given configuration. Any point in this body moves

freely according to the world frame, yet, it can be thought of as a combination of rotation and translation with respect to the body frame.

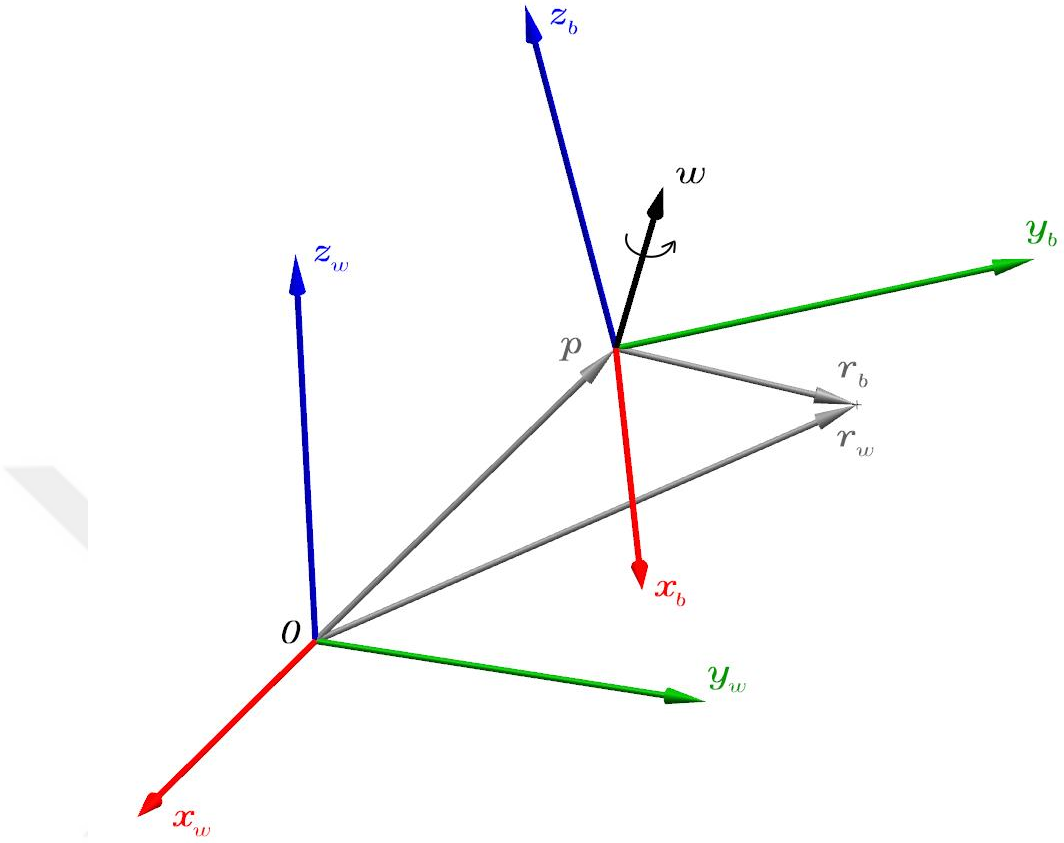

Figure 2.16 : The velocity of the rigid body.

At the origin of the body frame, there would exist an angular velocity vector according to the rotational motion that a rigid body possesses. Firstly, let us define the position of the point with respect to the world frame for the given configuration.

$$\boldsymbol{r}_w(t) = \boldsymbol{p}(t) + \boldsymbol{R}(t)\boldsymbol{r}_b \tag{2.185}$$

By remarking, any body point, $\boldsymbol{r}_b$, would remain constant in any arbitrary value of $\boldsymbol{p}$ and $\boldsymbol{R}$.

$$\boldsymbol{r}_b = \boldsymbol{R}(t)^T\big(\boldsymbol{r}_w(t) - \boldsymbol{p}(t)\big) \tag{2.186}$$

If we differentiate both sides of the Eq. (2.185):

$$\dot{\boldsymbol{r}}_w(t) = \dot{\boldsymbol{p}}(t) + \dot{\boldsymbol{R}}(t)\boldsymbol{r}_b \tag{2.187}$$

Then, replacing $\boldsymbol{r}_b$ with world frame parameters:

$$\dot{\boldsymbol{r}}_w(t) = \boldsymbol{v}(t) + \dot{\boldsymbol{R}}(t)\boldsymbol{R}(t)^T\big(\boldsymbol{r}_w(t) - \boldsymbol{p}(t)\big) \tag{2.188}$$

We end up with a familiar equation that we have discussed in pure rotational motion. Now, we can place the angular velocity vector into the equation as follows.

$$\dot{\boldsymbol{r}}_w(t) = \boldsymbol{v}(t) + \boldsymbol{\omega}(t) \times \big(\boldsymbol{r}_w(t) - \boldsymbol{p}(t)\big) \tag{2.189}$$

Hence, the velocity of any point of the rigid body can be formulated with an instantaneous angular velocity and translational velocity of the body frame. If the position of the body frame is known we require only two arguments for defining velocity which are $\boldsymbol{\omega}$ and $\boldsymbol{v}$. They are both determined in the world frame and can be used very widely for further investigation, e.g. velocity, acceleration analysis of robot manipulator, and its end-effector. We combine these two vectors as one which is called a twist vector.

$$\boldsymbol{t} = \begin{bmatrix} \boldsymbol{v} \\ \boldsymbol{\omega} \end{bmatrix} \tag{2.190}$$

In the forward geometrical model, we have used the pose of the end-effector by the vector 7x1. We can transform derivative of end-effector pose vector into 6x1 twist vector by the formulation that we have shown obtaining $\boldsymbol{\omega}$ from quaternions, Eq. (2.180-2.184).

$$\frac{d}{dt}\boldsymbol{s} \rightarrow \boldsymbol{t} \tag{2.191}$$

2.3.3.2 Velocity analysis of robot manipulator

In forward geometrical modeling of the robot manipulator, we have guided an effective approach for obtaining a relation between joint parameters, $\boldsymbol{q}$, and end-effector pose, $\boldsymbol{s}$. Despite the approach is convenient and easily programmable, the resulting equations are complex and include trigonometric functions which leads to multiple solutions while taking inverse. Nevertheless, if the robot configuration is determined, it is possible to solve all of the joint parameters in closed-form excluding singularities from the forward relation equation below.

$$\boldsymbol{s} = f(\boldsymbol{q}) \tag{2.192}$$

Where q is an n-dimensional vector and pose vector contains 7 parameters while only six of them are independent. For controlling manipulator in velocity level and also for our intensions in kinematic capabilities of the manipulator, it is required to obtain a velocity level equation which is likely below.

$$\frac{d\boldsymbol{s}}{dt} = \frac{\partial f(\boldsymbol{q})}{\partial \boldsymbol{q}} \frac{d\boldsymbol{q}}{dt} \tag{2.193}$$

Derivatives of the vectors are vectors as follows.

$$\frac{d\boldsymbol{s}}{dt} = \begin{bmatrix} \frac{d\boldsymbol{p}_{TCP}}{dt} \\ \frac{d\bar{\boldsymbol{q}}_{TCP}}{dt} \end{bmatrix}, \qquad \frac{d\boldsymbol{q}}{dt} = \begin{bmatrix} \dot{q}_1 \\ \dot{q}_2 \\ \vdots \\ \dot{q}_6 \end{bmatrix} \tag{2.194}$$

Further, the partially differentiated matrix of coefficients is a 7x6 matrix for our case and called as *analytical Jacobian matrix* of the robot manipulator.

$$\frac{\partial f(\boldsymbol{q})}{\partial \boldsymbol{q}} = \begin{bmatrix} \frac{\partial f_1(\boldsymbol{q})}{\partial q_1} & \frac{\partial f_1(\boldsymbol{q})}{\partial q_2} & \cdots & \frac{\partial f_1(\boldsymbol{q})}{\partial q_6} \\ \frac{\partial f_2(\boldsymbol{q})}{\partial q_1} & \frac{\partial f_2(\boldsymbol{q})}{\partial q_2} & \cdots & \frac{\partial f_2(\boldsymbol{q})}{\partial q_6} \\ \vdots & \vdots & \ddots & \vdots \\ \frac{\partial f_7(\boldsymbol{q})}{\partial q_1} & \frac{\partial f_7(\boldsymbol{q})}{\partial q_2} & \cdots & \frac{\partial f_7(\boldsymbol{q})}{\partial q_6} \end{bmatrix} \tag{2.195}$$

However, this approach is a bit challenging because of the constraint functions which is not determined directly in our FGM, and also it is not preferred nowadays. In 1972, D. E. Whitney proposed a geometrical approach for the Jacobian matrix. The equation is given below.

$$\boldsymbol{t} = \boldsymbol{\mathcal{J}}(\boldsymbol{q}) \frac{d\boldsymbol{q}}{dt} \tag{2.196}$$

Where $\boldsymbol{\mathcal{J}}$ is denoted for the *geometrical Jacobian matrix* of the manipulator, or in short called its *Jacobian* which gives us a relation between end-effector twist and joint rates using the instantaneous configuration of the robot. Jacobian can also be used for determining feasible motions of the end-effector.

To obtain Jacobian, firstly, it is required to define joint axes positions and orientations which we have discussed in FGM for the manipulator configuration. Then, creating angular velocity vectors of the joints whose contributions are fundamentals for end-effector twist.

Firstly, joint axis vectors are going to be rotated by the affecting robot joints successively:

$$\boldsymbol{e}_{r_1} = \boldsymbol{e}_1 \tag{2.197}$$

$$\boldsymbol{e}_{r_2} = \boldsymbol{R}_1 \boldsymbol{e}_2 \tag{2.198}$$

$$\boldsymbol{e}_{r_3} = \boldsymbol{R}_1 \boldsymbol{R}_2 \boldsymbol{e}_3 \tag{2.199}$$

$$\boldsymbol{e}_{r_4} = \boldsymbol{R}_1 \boldsymbol{R}_2 \boldsymbol{R}_3 \boldsymbol{e}_4 \tag{2.200}$$

$$\boldsymbol{e}_{r_5} = \boldsymbol{R}_1 \boldsymbol{R}_2 \boldsymbol{R}_3 \boldsymbol{R}_4 \boldsymbol{e}_5 \tag{2.201}$$

$$\boldsymbol{e}_{r_6} = \boldsymbol{R}_1 \boldsymbol{R}_2 \boldsymbol{R}_3 \boldsymbol{R}_4 \boldsymbol{R}_5 \boldsymbol{e}_6 \tag{2.202}$$

The angular velocity vector of the i'th joint would equal to multiplication of these vector with the i'th joint rate and the summation of all would correspond to EE angular velocity vector which is given below.

$$\boldsymbol{\omega}_{EE} = \sum_{i=1}^{6} \boldsymbol{e}_{r_i} \dot{q}_i \tag{2.203}$$

We have already determined joint position vectors for the robot configuration by denoting $\boldsymbol{p}_i$ vectors with respect to the base frame, Eq. (2.94-2.98). For translational velocity contributions, we require distance vectors between these position vectors and TCP position vector:

$$\boldsymbol{r}_i = \boldsymbol{p}_{TCP} - \boldsymbol{p}_i \tag{2.204}$$

And the translational velocity of each joint can be formulated as follows.

$$\boldsymbol{v}_i = \boldsymbol{e}_{r_i} \dot{q}_i \times \boldsymbol{r}_i \tag{2.205}$$

If all of the is going to be summed, one can obtain instantaneous translational velocity of the end-effector for its tool center point:

$$\boldsymbol{v}_{TCP} = \sum_{i=1}^{6} \boldsymbol{v}_i \tag{2.206}$$

Now, it is possible to obtain a 6x6 Jacobian matrix as the equation below.

$$\boldsymbol{t}_{TCP} = \begin{bmatrix} \boldsymbol{v}_{TCP} \\ \boldsymbol{\omega}_{EE} \end{bmatrix} = \begin{bmatrix} \boldsymbol{e}_{r_1} \times \boldsymbol{r_1} & \boldsymbol{e}_{r_2} \times \boldsymbol{r_2} & \cdots & \boldsymbol{e}_{r_6} \times \boldsymbol{r_6} \\ \boldsymbol{e}_{r_1} & \boldsymbol{e}_{r_2} & \cdots & \boldsymbol{e}_{r_6} \end{bmatrix} \dot{\boldsymbol{q}} \tag{2.207}$$

$$\boldsymbol{\mathcal{J}}(\boldsymbol{q}) = \frac{\partial \boldsymbol{t}_{TCP}}{\partial \dot{\boldsymbol{q}}} \tag{2.208}$$

Each column of the Jacobian matrix is formed by the contribution of each joint. Hence, one may represent the columns as follows.

$$\boldsymbol{j}_i = \begin{bmatrix} \boldsymbol{e}_{r_i} \times \boldsymbol{r_i} \\ \boldsymbol{e}_{r_i} \end{bmatrix} \tag{2.209}$$

2.3.3.3 Solving joint rates from the Jacobian matrix

Jacobian matrix is a milestone for robot control because it provides a linear relationship between EE twist and joint rates as long as joint parameters are known. In this purpose, generally, sensors exist in robot manipulators to collect joint parameters. After robot Jacobian evaluated, any desired path tracking applications owning high velocities becomes feasible in a possible success criterion. Evaluated Jacobian is required to be taken inverse in a determined sample rate. This control method is called *the resolved motion rate* and might be the only solution for manipulators that do not possess any inverse geometric solution.

For decoupled manipulators, we do not need to take the inverse of the 6x6 matrix. It is possible to create Jacobian not for TCP twist but WCP which cancels out the translational velocity contributions of the wrist joints. We have shown getting TCP to WCP in the IGM section for the given pose. The same transformation can be done for the twists and the solution can be simplified.

As it can be seen in **Figure 2.17**, WCP and TCP can be assumed as two points on the same rigid body. In reality, WCP stays out of the end-effector body. However, for

decoupled manipulators, as the wrist axes coincide at one point, its position does not vary with respect to tool frame and can be thought of as a body point.

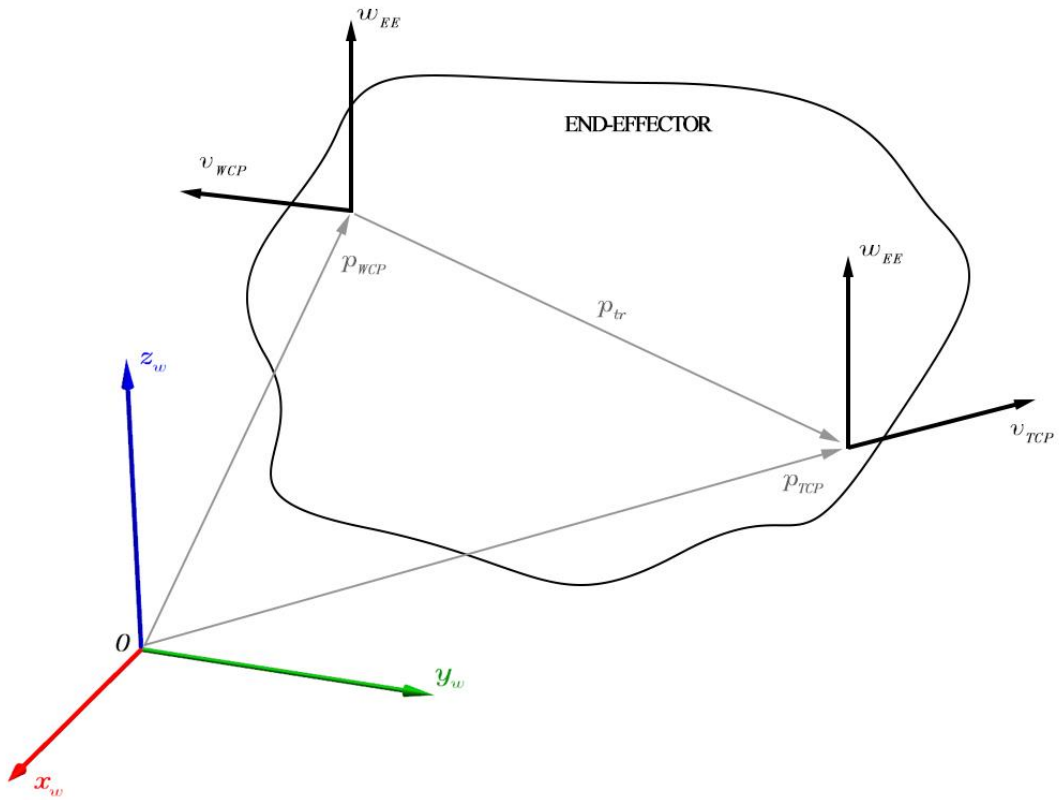

Figure 2.17 : Twist transfer TCP to WCP.

Henceforth, instantaneous angular velocity remains the same for any selected position. That is why we parametrize angular velocities of both WCP and TCP as $\boldsymbol{\omega}_{EE}$. Whereas, the translational velocity of a rotating rigid body varies according to the selected position. In order to find the translational velocity of an arbitrary position, it is required to be known translational velocity of another point on the body and transform it with respect to the instantaneous angular velocity.

We define the position vector between WCP and TCP in the base frame as:

$$\boldsymbol{p}_{tr} = \boldsymbol{p}_{TCP} - \boldsymbol{p}_{WCP} \tag{2.210}$$

The derivative of this vector is going to be found by cross producing with angular velocity from the right-hand side which eliminates negative sign in the equation:

$$\dot{\boldsymbol{p}}_{tr} = -\boldsymbol{p}_{tr} \times \boldsymbol{\omega}_{EE} \tag{2.211}$$

And then, the translational velocity of WCP can be found as follows.

$$\boldsymbol{v}_{WCP} = \boldsymbol{v}_{TCP} + \boldsymbol{p}_{tr} \times \boldsymbol{\omega}_{EE} \tag{2.212}$$

We can define a cross-product matrix of $\boldsymbol{p}_{tr}$ for easiness.

$$\boldsymbol{P}_{tr} = \begin{bmatrix} 0 & -p_{tr_3} & p_{tr_2} \\ p_{tr_3} & 0 & -p_{tr_1} \\ -p_{tr_2} & p_{tr_1} & 0 \end{bmatrix} \tag{2.213}$$

Thus, we can define Eq. (2.211) without cross-production.

$$\dot{\boldsymbol{p}}_{tr} = -\boldsymbol{P}_{tr}\, \boldsymbol{\omega}_{EE} \tag{2.214}$$

After obtaining the translational velocity of WCP, we can define a new twist vector as follows.

$$\boldsymbol{t}_{WCP} = \begin{bmatrix} \boldsymbol{v}_{WCP} \\ \boldsymbol{\omega}_{EE} \end{bmatrix} \tag{2.215}$$

Now, we are going to re-create the Jacobian matrix for the wrist center point. For that purpose, we re-create distance vectors in Eq. (2.204) with respect to WCP.

$$\boldsymbol{r}_i = \boldsymbol{p}_{WCP} - \boldsymbol{p}_i \tag{2.216}$$

Now the Jacobian for WCP becomes as follows:

$$\boldsymbol{\mathcal{J}}_{WCP} = \begin{bmatrix} \boldsymbol{\mathcal{J}}_{11} & \mathbf{0} \\ \boldsymbol{\mathcal{J}}_{21} & \boldsymbol{\mathcal{J}}_{22} \end{bmatrix} \tag{2.217}$$

Where $\mathbf{0}$ is a 3x3 zeros matrix and other sub-matrix are as follows.

$$\boldsymbol{\mathcal{J}}_{11} = [\boldsymbol{e}_{r_1} \times \boldsymbol{r}_1 \quad \boldsymbol{e}_{r_2} \times \boldsymbol{r}_2 \quad \boldsymbol{e}_{r_3} \times \boldsymbol{r}_3] \tag{2.218}$$

$$\boldsymbol{\mathcal{J}}_{21} = [\boldsymbol{e}_{r_1} \quad \boldsymbol{e}_{r_2} \quad \boldsymbol{e}_{r_3}] \tag{2.219}$$

$$\boldsymbol{\mathcal{J}}_{22} = [\boldsymbol{e}_{r_4} \quad \boldsymbol{e}_{r_5} \quad \boldsymbol{e}_{r_6}] \tag{2.220}$$

For this Jacobian, it is possible to separate EE twist for shoulder and wrist joint rates separately.

$$\dot{\boldsymbol{q}} = \begin{bmatrix} \dot{\boldsymbol{q}}_{sh} \\ \dot{\boldsymbol{q}}_{wr} \end{bmatrix} \tag{2.221}$$

$$\dot{\boldsymbol{q}}_{sh} = \begin{bmatrix} \dot{q}_1 \\ \dot{q}_2 \\ \dot{q}_3 \end{bmatrix}, \qquad \dot{\boldsymbol{q}}_{wr} = \begin{bmatrix} \dot{q}_4 \\ \dot{q}_5 \\ \dot{q}_6 \end{bmatrix} \tag{2.222}$$

Then, the formulation can be re-written by using sub-matrices.

$$\boldsymbol{v}_{WCP} = \boldsymbol{\mathcal{J}}_{11}\dot{\boldsymbol{q}}_{sh} \tag{2.223}$$

$$\boldsymbol{\omega}_{EE} = \boldsymbol{\mathcal{J}}_{21}\dot{\boldsymbol{q}}_{sh} + \boldsymbol{\mathcal{J}}_{22}\dot{\boldsymbol{q}}_{wr} \tag{2.224}$$

For solving Eq. (2.222 and 2.223), it is apparently required to solve first $\dot{\boldsymbol{q}}_{sh}$ and then $\dot{\boldsymbol{q}}_{wr}$.

$$\dot{\boldsymbol{q}}_{sh} = [\boldsymbol{\mathcal{J}}_{11}]^{-1}\boldsymbol{v}_{WCP} \tag{2.225}$$

After solving shoulder joint rates, we can define a new instantaneous angular velocity vector as $\boldsymbol{\omega}_{wr}$ which gives a direct relation to the wrist joint rates:

$$\boldsymbol{\omega}_{wr} = \boldsymbol{\omega}_{EE} - \boldsymbol{\mathcal{J}}_{21}\dot{\boldsymbol{q}}_{sh} \tag{2.226}$$

$$\dot{\boldsymbol{q}}_{wr} = [\boldsymbol{\mathcal{J}}_{22}]^{-1}\boldsymbol{\omega}_{wr} \tag{2.227}$$

The equation above is very significant because of the direct relation. We have already stated, in the neighborhood of wrist singular configurations, wrist joint rates get abnormally high. At this point, we figure out, ill-conditioned $\boldsymbol{\mathcal{J}}_{22}$ causes high wrist joint rates. Fairly, we cannot manipulate $\boldsymbol{\mathcal{J}}_{22}$ because it is related to the posture of manipulator but we can manipulate $\boldsymbol{\omega}_{wr}$ in order to overcome high joint rates.

The inverses of Jacobian sub-matrices can be formulated as given below.

$$\Delta_{11} = \det(\boldsymbol{\mathcal{J}}_{11}) = \left[\left(\boldsymbol{e}_{r_1} \times \boldsymbol{r}_1\right) \times \left(\boldsymbol{e}_{r_2} \times \boldsymbol{r}_2\right) \cdot \left(\boldsymbol{e}_{r_3} \times \boldsymbol{r}_3\right)\right] \tag{2.228}$$

$$\Delta_{22} = \det(\boldsymbol{\mathcal{J}}_{22}) = \left[\boldsymbol{e}_{r_4} \times \boldsymbol{e}_{r_5} \cdot \boldsymbol{e}_{r_6}\right] \tag{2.229}$$

$$[\mathcal{J}_{11}]^{-1} = \frac{1}{\Delta_{11}} \begin{bmatrix} [(\boldsymbol{e}_{r_2} \times \boldsymbol{r}_2) \times (\boldsymbol{e}_{r_3} \times \boldsymbol{r}_3)]^T \\ [(\boldsymbol{e}_{r_3} \times \boldsymbol{r}_3) \times (\boldsymbol{e}_{r_1} \times \boldsymbol{r}_1)]^T \\ [(\boldsymbol{e}_{r_1} \times \boldsymbol{r}_1) \times (\boldsymbol{e}_{r_2} \times \boldsymbol{r}_2)]^T \end{bmatrix} \tag{2.230}$$

$$[\mathcal{J}_{22}]^{-1} = \frac{1}{\Delta_{22}} \begin{bmatrix} [\boldsymbol{e}_{r_5} \times \boldsymbol{e}_{r_6}]^T \\ [\boldsymbol{e}_{r_6} \times \boldsymbol{e}_{r_4}]^T \\ [\boldsymbol{e}_{r_4} \times \boldsymbol{e}_{r_5}]^T \end{bmatrix} \tag{2.231}$$

2.3.4 Acceleration analysis

A similar approach to velocity analysis will be introduced here. We firstly define rigid-body acceleration and then create the twist rate vector, $\dot{\boldsymbol{t}}$. After that, we will introduce the Jacobian rate matrix. Then, we will show how to solve joint acceleration parameters.

2.3.4.1 Rigid body acceleration

By differentiating Eq. (2.189), we can obtain the acceleration of a body point of a moving rigid body with respect to an inertial frame.

$$\begin{aligned} \ddot{\boldsymbol{r}}_w(t) = \dot{\boldsymbol{v}}(t) + \dot{\boldsymbol{\omega}}(t) \times \big(\boldsymbol{r}_w(t) - \boldsymbol{p}(t)\big) \\ + \boldsymbol{\omega}(t) \times \big[\boldsymbol{\omega}(t) \times \big(\boldsymbol{r}_w(t) - \boldsymbol{p}(t)\big)\big] \end{aligned} \tag{2.232}$$

Here, angular acceleration defined as $\dot{\boldsymbol{\omega}}(t)$ and can be obtained by differentiating Eq. (2.171):

$$\dot{\boldsymbol{\omega}}(t) = \ddot{\theta}(t)\boldsymbol{e}(t) + \dot{\theta}(t)\dot{\boldsymbol{e}}(t) \tag{2.233}$$

$\dot{\boldsymbol{v}}(t)$ assumed to be known and other terms required to be calculated by velocity analysis.

2.3.4.2 Acceleration analysis of robot manipulator

By differentiating the geometric Jacobian equation that is proposed by Whitney, we get the following formulation.

$$\frac{d\boldsymbol{t}}{dt} = \frac{d\boldsymbol{\mathcal{J}}(\boldsymbol{q})}{dt}\frac{d\boldsymbol{q}}{dt} + \boldsymbol{\mathcal{J}}(\boldsymbol{q})\frac{d^2\boldsymbol{q}}{dt^2} \tag{2.234}$$

$$\dot{\boldsymbol{t}} = \dot{\boldsymbol{\mathcal{J}}}(\boldsymbol{q}, \dot{\boldsymbol{q}})\ \dot{\boldsymbol{q}} + \boldsymbol{\mathcal{J}}(\boldsymbol{q})\ \ddot{\boldsymbol{q}} \tag{2.235}$$

This is the formula for obtaining a twist rate vector by using instantaneous joint accelerations and joint velocities. However, we get an unknown jacobian rate term as $\dot{\boldsymbol{\mathcal{J}}}(\boldsymbol{q}, \dot{\boldsymbol{q}})$ that we are needed to calculate. Before this calculation, it has to be noticed that the formulation would be suitable for only WCP but not TCP. The reason is the same as the velocity approach; after solving the shoulder side, it is much easier to handle the wrist side. In that purpose, we firstly transform twist rate TCP to WCP by using the derivative of Eq. (2.212):

$$\dot{\boldsymbol{v}}_{WCP} = \dot{\boldsymbol{v}}_{TCP} + \dot{\boldsymbol{p}}_{tr} \times \boldsymbol{\omega}_{EE} + \boldsymbol{p}_{tr} \times \dot{\boldsymbol{\omega}}_{EE} \tag{2.236}$$

Here, $\dot{\boldsymbol{\omega}}_{EE}$ is invariant for TCP and WCP as same as $\boldsymbol{\omega}_{EE}$. Hence, we get a formula for all known terms as follows.

$$\dot{\boldsymbol{v}}_{WCP} = \dot{\boldsymbol{v}}_{TCP} + \boldsymbol{\omega}_{EE} \times (\boldsymbol{P}_{tr}\ \boldsymbol{\omega}_{EE}) + \boldsymbol{P}_{tr}\ \dot{\boldsymbol{\omega}}_{EE}$$

Then, we can define the twist rate for WCP as:

$$\dot{\boldsymbol{t}}_{WCP} = \begin{bmatrix} \dot{\boldsymbol{v}}_{WCP} \\ \dot{\boldsymbol{\omega}}_{EE} \end{bmatrix} \tag{2.237}$$

Now, we are going to create the Jacobian rate matrix for the WCP. An algorithm is given by J. Angeles [40]. We will apply the same without giving detailed information.

$$\dot{\boldsymbol{\mathcal{J}}}_{WCP} = \begin{bmatrix} \dot{\boldsymbol{\mathcal{J}}}_{11} & \mathbf{0} \\ \dot{\boldsymbol{\mathcal{J}}}_{21} & \dot{\boldsymbol{\mathcal{J}}}_{22} \end{bmatrix} \tag{2.238}$$

Where **0** is a 3x3 zeros matrix and other sub-matrix are as follows.

$$\dot{\boldsymbol{\mathcal{J}}}_{11} = [\dot{\boldsymbol{u}}_1 \quad \dot{\boldsymbol{u}}_2 \quad \dot{\boldsymbol{u}}_3] \tag{2.239}$$

$$\dot{\boldsymbol{\mathcal{J}}}_{21} = [\dot{\boldsymbol{e}}_{r_1} \quad \dot{\boldsymbol{e}}_{r_2} \quad \dot{\boldsymbol{e}}_{r_3}] \tag{2.240}$$

$$\dot{\boldsymbol{\mathcal{J}}}_{22} = [\dot{\boldsymbol{e}}_{r_4} \quad \dot{\boldsymbol{e}}_{r_5} \quad \dot{\boldsymbol{e}}_{r_6}] \tag{2.241}$$

Before the definition of this sub-vectors, we firstly define the accumulated angular velocity of each joint as:

$$\boldsymbol{\omega}_i = \sum_{i=1}^{6} \boldsymbol{e}_{r_i} \dot{\boldsymbol{q}}_i \tag{2.242}$$

Then, derivatives of joint axes can be defined as

$$\dot{\boldsymbol{e}}_{\boldsymbol{r}_i} = \sum_{i=1}^{6} \boldsymbol{\omega}_i \times \boldsymbol{e}_{r_i} \tag{2.243}$$

$\dot{\boldsymbol{u}}_i$ is defined as follows.

$$\dot{\boldsymbol{u}}_i = \dot{\boldsymbol{e}}_{\boldsymbol{r}_i} \times \boldsymbol{r}_i + \boldsymbol{e}_{r_i} \times \dot{\boldsymbol{r}}_i \tag{2.244}$$

For obtaining $\dot{\boldsymbol{r}}_i$ we create a new vector parameter that defines the instantaneous distance between joint axis:

$$\boldsymbol{a}_i = \boldsymbol{p}_{i+1} - \boldsymbol{p}_i \tag{2.245}$$

And its derivatives equal to:

$$\dot{\boldsymbol{a}}_i = \boldsymbol{\omega}_i \times \boldsymbol{a}_i \tag{2.246}$$

Now, we can define $\dot{\boldsymbol{r}}_i$ as follows.

$$\dot{\boldsymbol{r}}_i = \sum_{i=1}^{3} \dot{\boldsymbol{a}}_i \tag{2.247}$$

All terms in $\dot{\boldsymbol{\mathcal{J}}}_{WCP}$ are defined and matrix can be instantaneously calculated by using instantaneous FGM and joint rates.

2.3.4.3 Solving joint accelerations

We can solve joint accelerations from a given WCP twist rate vector by using the formula below. However, joint velocities should be known.

$$\ddot{\boldsymbol{q}} = [\boldsymbol{\mathcal{J}}_{WCP}]^{-1} \left(\dot{\boldsymbol{t}}_{WCP} - \dot{\boldsymbol{\mathcal{J}}}_{WCP}\, \dot{\boldsymbol{q}} \right) \tag{2.248}$$

Furthermore, we can separate joint accelerations as shoulder and wrist sides.

$$\ddot{\boldsymbol{q}} = \begin{bmatrix} \ddot{\boldsymbol{q}}_{sh} \\ \ddot{\boldsymbol{q}}_{wr} \end{bmatrix} \tag{2.249}$$

Then, they can be solved separately similar to the joint velocities.

$$\ddot{\boldsymbol{q}}_{sh} = [\boldsymbol{\mathcal{J}}_{11}]^{-1} \left(\dot{\boldsymbol{v}}_{WCP} - \dot{\boldsymbol{\mathcal{J}}}_{11} \dot{\boldsymbol{q}}_{sh} \right) \tag{2.250}$$

After solving shoulder joint accelerations, we can define a new instantaneous angular acceleration vector as $\dot{\boldsymbol{\omega}}_{wr}$ which gives a direct relation to the wrist joint accelerations:

$$\dot{\boldsymbol{\omega}}_{wr} = \dot{\boldsymbol{\omega}}_{EE} - \dot{\boldsymbol{\mathcal{J}}}_{21} \dot{\boldsymbol{q}}_{sh} - \boldsymbol{\mathcal{J}}_{21} \ddot{\boldsymbol{q}}_{sh} \tag{2.251}$$

$$\ddot{\boldsymbol{q}}_{wr} = [\boldsymbol{\mathcal{J}}_{22}]^{-1} \dot{\boldsymbol{\omega}}_{wr}$$

The equation above is also significant as same as the wrist velocity equation because of the direct relation. Similar to the velocity equation, ill-conditioned $\boldsymbol{\mathcal{J}}_{22}$ causes high wrist joint accelerations. We can manipulate $\dot{\boldsymbol{\omega}}_{wr}$ in order to overcome this issue.

2.4 Path Planning

In this section, the time-varying motion of the TCP frame will be introduced. The path will contain position information of TCP and orientation information of the tool frame with respect to the base frame. Besides, velocity and acceleration levels will be investigated. In robot models, instantaneous twist and twist rate vectors were deemed their significance for solving joint parameters. Here, these vectors will be created according to the desired path.

For moving robot TCP through one point to another, there are infinite ways of accomplishing this task because the motion is time-dependent and there is no constraint. However, if the time-varying position information between these points is specified mathematically, then there is a unique solution because velocities and accelerations of the path are also specified. In this manner, the desired motion should be known in the first step.

In robotic applications, the velocity of the TCP is generally expected to stay constant on the desired value. This helps improving process quality and also beneficial for robot motion. In our models, we will assume robot would start from desired motion speed

and keep that speed in the whole path. Hence, there will be no initial condition disturbances in our models.

It is possible to mention to two fundamental paths for position: linear and circular path. Their mathematical formulations are going to be represented below.

2.4.1 Linear path

At least two points are required for drawing a line. It requires the same for the linear path. Let us define two points, $\boldsymbol{p}_a$and $\boldsymbol{p}_b$, to establish start and end positions of the linear path.

The length of the linear path would be the distance between these points as defined below.

$$L = \|\boldsymbol{p}_b - \boldsymbol{p}_a\| \tag{2.252}$$

Then, we are going to define the direction of the path, $\boldsymbol{d}$, as a normalized vector as follows.

$$\boldsymbol{d} = \frac{\boldsymbol{p}_b - \boldsymbol{p}_a}{L} \tag{2.253}$$

After they are found, robot sample time and desired velocity of the path are required for discretization. We assume they are constant values, respectively Δt and v_0. For tracking with the resolved-rate method, it is required to obtain a discretized path. Firstly, we are going to determine the travel time of the path, T, as follows.

$$T = \frac{L}{v_0} \tag{2.254}$$

Then, we will describe position incrementation of the path by using direction, velocity, and sample time as below.

$$\Delta\boldsymbol{p} = \frac{v_0}{\Delta t}\boldsymbol{d} \tag{2.255}$$

After they defined, another calculation is needed for how many steps there will be in the discretized position path. We define a count parameter for steps, H, as below.

$$H = T/\Delta t \tag{2.256}$$

H has to be a natural number because it corresponds to the count of steps and should be rounded if needed. Further, It defines the upper limit of the discrete variable, k.

After all the calculations, it is possible to create a whole discretized path as below. The first point would be equal to the start point and the next step would be calculated successively by using position incrementation.

$$\boldsymbol{p}_{TCP}(1) = \boldsymbol{p}_a \tag{2.257}$$

The equation below is going to be repeated for each step while k value is incremented for 1 at every step. The process starts when k equal to 1 and should be stopped when k reaches to H.

$$\boldsymbol{p}_{TCP}(k+1) = \boldsymbol{p}_{TCP}(k) + \Delta\boldsymbol{p} \tag{2.258}$$

The velocity of the linear path would be constant for every discrete variable, k, as follows.

$$\boldsymbol{v}_{TCP}(k) = v_0\boldsymbol{d} \tag{2.259}$$

Acceleration of the linear path would equal to zero for every discrete variable, k, as follows.

$$\dot{\boldsymbol{v}}_{TCP}(k) = 0 \tag{2.260}$$

An illustration is given in **Figure 2.18**.

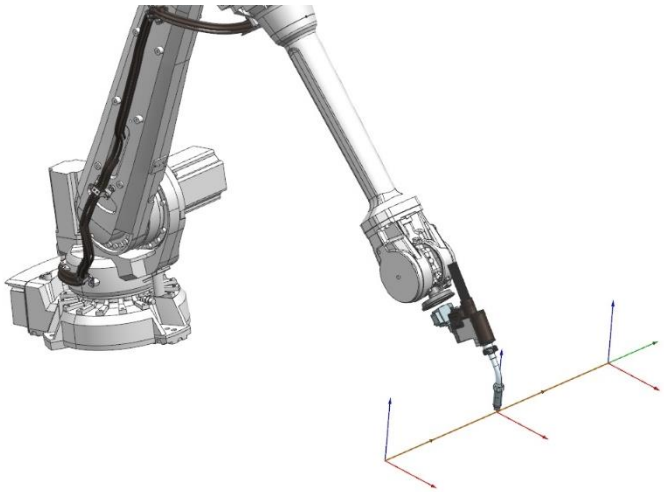

Figure 2.18 : Linear Path.

2.4.2 Circular path

To draw a circle in a plane, the most common method is using the information of the circle center point and its radius, inherently. On the other hand, one can use the information of three points that circle is going to pass through. That also provides the required information to establish a circle or an arc model. However, to draw a circle in Euclidean space, plane information is also required. In that case, three points on the circle can be more useful because they also form a plane in Euclidean space.

In our model, we are going to use three points that circular path starts, passes through, and ends. To establish a model, it is required to find the circle center correspond to those three points. In that regard, we firstly define the plane and then solve the linear system to find out the circle center point.

Let us define points as $\boldsymbol{p}_a$, $\boldsymbol{p}_b$ and $\boldsymbol{p}_c$. Two vectors, $\boldsymbol{v}_1$ and $\boldsymbol{v}_2$, can be created from these points as below.

$$\boldsymbol{v}_1 = \boldsymbol{p}_b - \boldsymbol{p}_a \tag{2.261}$$

$$\boldsymbol{v}_2 = \boldsymbol{p}_c - \boldsymbol{p}_b \tag{2.262}$$

The normalized cross-production of those vectors would correspond to a normal vector of the plane which we call as $\boldsymbol{e}_{circle}$.

$$\boldsymbol{e}_{circle} = \frac{\boldsymbol{v}_1 \times \boldsymbol{v}_2}{\|\boldsymbol{v}_1 \times \boldsymbol{v}_2\|} \tag{2.263}$$

Circle center point, $\boldsymbol{p}_{cp}$, can be found from the linear system below.

$$\boldsymbol{p}_{cp} = \boldsymbol{K}.\boldsymbol{f} \tag{2.264}$$

Where $\boldsymbol{K}$ matrix and $\boldsymbol{f}$ vector are formed as follows.

$$\boldsymbol{K} = [\boldsymbol{v}_1 \quad \boldsymbol{v}_2 \quad \boldsymbol{e}_{circle}] \tag{2.265}$$

$$\boldsymbol{f} = \begin{bmatrix} (\boldsymbol{p}_b^T \boldsymbol{p}_b - \boldsymbol{p}_a^T \boldsymbol{p}_a)/2 \\ (\boldsymbol{p}_c^T \boldsymbol{p}_c - \boldsymbol{p}_b^T \boldsymbol{p}_b)/2 \\ \boldsymbol{e}^T \boldsymbol{p}_b \end{bmatrix} \tag{2.266}$$

After the center point is found, the radius of the circle can be found from any given points:

$$r = \|\boldsymbol{p}_c - \boldsymbol{p}_{cp}\| \tag{2.267}$$

Moreover, radius vectors can be created for the given points:

$$\boldsymbol{r}_a = \boldsymbol{p}_a - \boldsymbol{p}_{cp} \tag{2.268}$$

$$\boldsymbol{r}_b = \boldsymbol{p}_b - \boldsymbol{p}_{cp} \tag{2.269}$$

$$\boldsymbol{r}_c = \boldsymbol{p}_c - \boldsymbol{p}_{cp} \tag{2.270}$$

If $\boldsymbol{p}_a$ and $\boldsymbol{p}_c$ is given as the starting point and endpoint of the circular path, we can define the arc angle where the motion carries out. In that purpose, starting and end coordinate frames can be created just like in **Figure 2.19** as follows.

$$\boldsymbol{R}_a = \left[\frac{\boldsymbol{r}_a}{r} \quad \frac{\boldsymbol{r}_a}{r} \times \boldsymbol{e}_{circle} \quad \boldsymbol{e}_{circle}\right] \tag{2.271}$$

$$\boldsymbol{R}_c = \left[\frac{\boldsymbol{r}_c}{r} \quad \frac{\boldsymbol{r}_c}{r} \times \boldsymbol{e}_{circle} \quad \boldsymbol{e}_{circle}\right] \tag{2.272}$$

These frames can be transformed into quaternion form by Cayley's method which we have proposed in sub-section 2.2.2. After they created, one can solve the arc angle from those quaternions.

$$\overline{\boldsymbol{q}}_{arc} = \overline{\boldsymbol{q}}_c^{\,-1} \star \overline{\boldsymbol{q}}_a \tag{2.273}$$

$$\overline{\boldsymbol{q}}_{arc} \rightarrow \theta_{arc} \tag{2.274}$$

Arc angle also defines the length of the circular path as follows.

$$L = \theta_{arc} * r \tag{2.275}$$

By using the length and process speed, we can find travel time as below.

$$T = \frac{L}{v_0} \tag{2.276}$$

And then, we can calculate the discrete step count as follows.

$$H = T/\Delta t \tag{2.277}$$

Also, we can define angle increment for each step by using process speed and radius:

$$\Delta\theta = \frac{v_0}{r} \tag{2.278}$$

Now, it is possible to form a discretized path for circular motion. We will start with the TCP positions. The first point would be equal to the start point and the next step would be calculated successively by using position incrementation. They would be created with the rotation matrices which formed with a plane normal and an angle increment.

$$\boldsymbol{p}_r(1) = \boldsymbol{r}_a \tag{2.279}$$

Equations below are going to be repeated for each step while k value is incremented for 1 at every step. The process starts when k equal to 1 and should be stopped when k reaches to H.

$$\boldsymbol{p}_r(k+1) = \boldsymbol{R}(\Delta\theta, \boldsymbol{e}_{circle})\, \boldsymbol{p}_r(k) \tag{2.280}$$

$$\boldsymbol{p}_{TCP}(k+1) = \boldsymbol{p}_{cp} + \boldsymbol{p}_r(k+1) \tag{2.281}$$

For velocity and acceleration, it is required to determine a new parameter as the angular velocity of the process, $\dot{\theta}_0$. It can be calculated by the equation given below.

$$\dot{\theta}_0 = \frac{\Delta\theta}{\Delta t} \tag{2.282}$$

Then, the angular velocity vector of the process can be formed as follows.

$$\boldsymbol{\omega}_0 = \dot{\theta}_0 \boldsymbol{e}_{circle} \tag{2.283}$$

After those, we can define velocity and acceleration analytically for each step as follows.

$$\boldsymbol{v}_{TCP}(k) = \boldsymbol{\omega}_0 \times \boldsymbol{p}_r(k) \tag{2.284}$$

$$\dot{\boldsymbol{v}}_{TCP}(k) = -\boldsymbol{\omega}_0^T \boldsymbol{\omega}_0 \boldsymbol{p}_r(k) \tag{2.285}$$

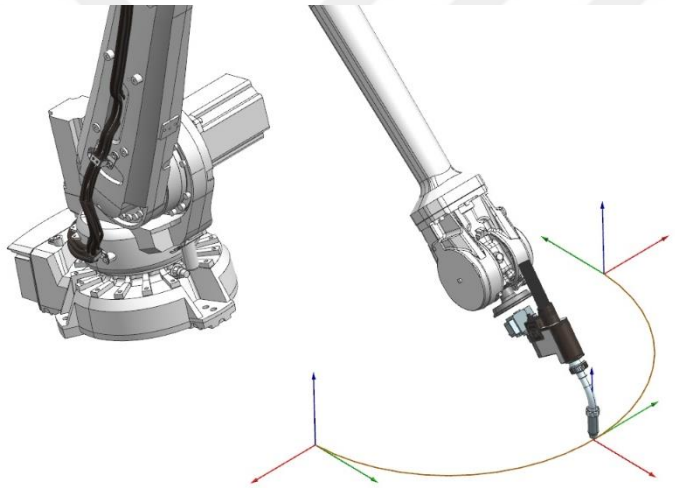

Figure 2.19 : Circular path.

2.4.3 Orientation path

Discussion for the orientation path will be done superficially. Arbitrary two orientations can be interpolated along the minimum path between them. By assumption, initial velocity can be selected equal to path angular velocity. Hence, there will not be any angular acceleration through motion. Travel time and step count which are found in position path would be also valid for orientation path as they simultaneously change.

First of all, by using rotation quaternion between two quaternion orientations, we can get the angle and axis which is given below.

$$\overline{\boldsymbol{q}}_{rot} = \overline{\boldsymbol{q}}_b^{\,-1} \star \overline{\boldsymbol{q}}_a \tag{2.286}$$

From rotation quaternion, angle and axis can be solved as follows.

$$\frac{\theta_{rot}}{2} = \cos^{-1}(q_{rot_0}) \tag{2.287}$$

$$\sin\frac{\theta_{rot}}{2} = \sqrt{1 - \left(q_{rot_0}\right)^2} \tag{2.288}$$

$$\boldsymbol{e}_{rot} = \boldsymbol{R}(q_0, \boldsymbol{q}) * \frac{1}{\sin\frac{\theta_{rot}}{2}} * \begin{bmatrix} q_{rot_1} \\ q_{rot_2} \\ q_{rot_3} \end{bmatrix} \tag{2.289}$$

When the total rotation angle is divided by the step count, we can get the incremental angle as follows.

$$\Delta\theta_{rot} = \frac{\theta_{rot}}{H} \tag{2.290}$$

After that, the incremental rotation quaternion can be created as below.

$$\Delta\overline{\boldsymbol{q}}_{rot} = \begin{bmatrix} \cos(\Delta\theta_{rot}/2) \\ \sin(\Delta\theta_{rot}/2)\,\boldsymbol{e}_{rot} \end{bmatrix} \tag{2.291}$$

The first point would be equal to start point and then, incrementally rotating initial orientation through steps will form orientation path.

$$\overline{\boldsymbol{q}}_{TCP}(1) = \overline{\boldsymbol{q}}_a \tag{2.292}$$

The equation below is going to be repeated for each step while k value is incremented for 1 at every step. The process starts when k equal to 1 and should be stopped when k reaches to H.

$$\overline{\boldsymbol{q}}_{TCP}(k+1) = \Delta\overline{\boldsymbol{q}}_{rot} \star \overline{\boldsymbol{q}}_{TCP}(k) \tag{2.293}$$

As the rotation axis is not changing through motion, angular velocity would be a constant value which is given below.

$$\boldsymbol{\omega}_{TCP}(k) = \frac{\theta_{rot}}{T}\boldsymbol{e}_{rot} \tag{2.294}$$

Angular acceleration of the orientation path would equal to zero for every discrete variable, k, as follows.

$$\dot{\boldsymbol{\omega}}_{TCP}(k) = 0 \tag{2.295}$$

2.4.4 Re-construction of the path

In this subsection, we are going to describe how to implement path changes in our path models. As we mentioned in the resolved-rate method, in order to overcome infeasible solutions of joint velocities and accelerations which are caused by ill-conditioned $\boldsymbol{J}_{22}$, we desire to manipulate $\boldsymbol{\omega}_{wr}$ and $\dot{\boldsymbol{\omega}}_{wr}$. In that purpose, path re-construction directly helps our cause. It affects resolved-rate solutions because $\boldsymbol{\omega}_{wr}$ and $\dot{\boldsymbol{\omega}}_{wr}$ are nothing but variables that are calculated with respect to the given path. However, we are planning to change the path whenever our robot is around the neighborhood of wrist singularity. Thus, this manipulation should be done in a specific interval of the discretized path.

Continuity is the key issue for path planning. The highest derivative of our models is at the acceleration level. At an arbitrary step of k, it is not possible to change the position of velocity directly because they are generated with respect to acceleration. If we decide to make a change in an arbitrary step of k, we can only change acceleration. Whenever the acceleration is changed, it would affect the velocity of the next step and also would affect the two-step forward position. That is why integration is required whenever a change is specified in the acceleration level.

After position and orientation path models are determined, one can form twist and twist rate vectors for the discrete path by concatenating translation and rotation vertically as below.

$$\boldsymbol{t}_{TCP}(k) = \begin{bmatrix} \boldsymbol{v}_{TCP}(k) \\ \boldsymbol{\omega}_{TCP}(k) \end{bmatrix}, \qquad \dot{\boldsymbol{t}}_{TCP}(k) = \begin{bmatrix} \dot{\boldsymbol{v}}_{TCP}(k) \\ \dot{\boldsymbol{\omega}}_{TCP}(k) \end{bmatrix} \tag{2.296}$$

Now the integration of the twist vector can be done with respect to its rate as shown below.

$$\boldsymbol{t}_{TCP}(k+1) = \boldsymbol{t}_{TCP}(k) + \Delta t * \dot{\boldsymbol{t}}_{TCP}(k) \tag{2.297}$$

As we showed how to solve twist and twist rate vectors with the resolved-rate method, one can obtain joint velocity and accelerations for the given path. Then, one may require integration of robot joint parameters according to its velocities and accelerations which is given below.

$$\boldsymbol{q}(k+1) = \boldsymbol{q}(k) + \Delta t * \dot{\boldsymbol{q}}(k) + \frac{\Delta t^2}{2} * \ddot{\boldsymbol{q}}(k) \tag{2.298}$$

3. STUDIES FOR WRIST SINGULARITY

In this section, we are going to discuss proposed methods for handling the wrist singularity of industrial robots. Using kinematic redundancies in the neighbor of singularities is the key solution because we are losing the feasible motion of the EE at least in an arbitrary direction. This motion should be satisfied by the redundant axis, if possible, to pass through singularity. By assumption, we are not going to interested in shoulder based singularities. As we stated in the introduction, they can be easily avoidable by replacing the path in a suitable workspace area. Hence, the main purpose is carrying out the desired motion in presence of wrist singularity without violating path accuracy.

3.1 Definition of the Problem

In the most general case, we can create the formulation instantaneous twist of the EE for the redundant or non-redundant robot manipulator by using the equation below.

$$\boldsymbol{t} = \boldsymbol{\mathcal{J}}(\boldsymbol{q})\dot{\boldsymbol{q}} \tag{3.1}$$

The twist vector, $\boldsymbol{t}$, owns a size as operational space dimension, o, and joint rate vector, $\dot{\boldsymbol{q}}$, owns the size of n which is the joint space dimension. Thus, Jacobian is a $o \times n$ matrix.

This equation forms a linear system whose coefficient matrix parameters are related to robot posture. If the Jacobian is square, it would be possible to approach through directly taking the inverse of the matrix Jacobian to solve this linear equation. However, two obstructions are encountered:

- $\boldsymbol{\mathcal{J}}^{-1}(\boldsymbol{q})$ is not defined for the exact singular values because it is not full rank. Thus, in special configurations, it is incapable of determining joint space velocities for the desired instantaneous EE twist.

- Solutions that are given by the equation $\dot{\boldsymbol{q}} = \boldsymbol{\mathcal{J}}^{-1}(\boldsymbol{q})\,\boldsymbol{t}$, is correct near singularities. Whereas, they are generally impractical because of the corresponding very high joint rates.

If the Jacobian is not square, then, the linear system must be overdetermined or underdetermined which are relevant to dimensions of operational space and joint space. For the underdetermined case, the exact solution for joint rates cannot be found for the desired twist because there is not enough joint rate parameter to determine. Only an approximate solution exists and the good results are not guaranteed.

For the overdetermined case, as long as Jacobian is not losing rank by singularities, infinite joint rate solutions exist for the desired twist which is caused by kinematic redundancy. As we discussed in the introduction, kinematic redundancy may be related to intrinsically redundant manipulators or reduced operational space. In any case, $n > o$ and there are two subspace of Jacobian called *range space* and *null space*. The range space of the Jacobian corresponds to a change in EE twist for given joint rates, however, the null space of the Jacobian does not differ in twist motion but only in robot posture. Thus, arranging these subspaces according to intention is significant for optimization purposes.

Rectangular Jacobians cannot be inverted directly. Yet, linear systems are mostly used in other engineering areas and there is a method for inverting rectangular coefficient matrices called *pseudo-inverse*. We will give information about pseudo-inverse matrices.

3.2 Pseudo-Inverse of a Rectangular Matrix

Pseudo-Inverse or Moore-Penrose Inverse is a special subset of generalized inverses. Let us have an overdetermined linear system as follows.

$$\boldsymbol{y} = \boldsymbol{A}\boldsymbol{x} \tag{3.2}$$

Assuming **A** has a matrix size of $o \times n$, the general solution for $\boldsymbol{x}$ is given as follows.

$$\boldsymbol{x} = \underbrace{\boldsymbol{A}^{\dagger}\boldsymbol{y}}_{minimum-norm\ solution} + \underbrace{(\boldsymbol{I} - \boldsymbol{A}^{\dagger}\boldsymbol{A})\boldsymbol{z}}_{homogeneous-solution} \tag{3.3}$$

Where $\boldsymbol{z}$ is arbitrary vector and we denote $\boldsymbol{A}^{\dagger}$ for psuedo-inverse of the coefficient matrix. It satisfies following four conditions:

$$\boldsymbol{A}\boldsymbol{A}^{\dagger}\boldsymbol{A} = \boldsymbol{A} \tag{3.4}$$

$$\boldsymbol{A}^{\dagger}\boldsymbol{A}\boldsymbol{A}^{\dagger} = \boldsymbol{A}^{\dagger} \tag{3.5}$$

$$(\boldsymbol{A}\boldsymbol{A}^{\dagger})^{T} = \boldsymbol{A}\boldsymbol{A}^{\dagger} \tag{3.6}$$

$$(\boldsymbol{A}^{\dagger}\boldsymbol{A})^{T} = \boldsymbol{A}^{\dagger}\boldsymbol{A} \tag{3.7}$$

The first term in RHS of the Eq. (3.3) is called the minimum-norm solution that in the case of there is not any exact solution, pseudo-inverse provides minimum Euclidean distance for the least square solution of equation $\boldsymbol{y} = \boldsymbol{A}\boldsymbol{x}$ which minimizes $\boldsymbol{x}$.

$$\underbrace{min}_{\boldsymbol{x}} \ \|\boldsymbol{y} - \boldsymbol{A}\boldsymbol{x}\| \tag{3.8}$$

The second term in the Eq. (3.3) is called the homogenous solution and it corresponds to the null space of $\boldsymbol{A}$ which does not affect solution **x** but changes **y** by using arbitrary vector **z** to obtain the same **x**.

For obtaining pseudo-inverse of a rectangular matrix, there are some methods. Nakamura (1991) [30] mentions important ones. In the case of $\boldsymbol{A}$ is full rank following methods are useful:

If $n > o$ then rank A is equal to o. This inverse is called the *right-generalized inverse*.

$$\boldsymbol{A}^{\dagger} = \boldsymbol{A}^{T}(\boldsymbol{A}\boldsymbol{A}^{T})^{-\mathbf{1}} \tag{3.9}$$

If $n < o$ then rank A is equal to n. This inverse is also called the *left-generalized inverse*.

$$\boldsymbol{A}^{\dagger} = (\boldsymbol{A}\boldsymbol{A}^{T})^{-\mathbf{1}}\boldsymbol{A}^{T} \tag{3.10}$$

The left-generalized inverse can be used for underdetermined systems, but it is out of our scope. In the cases of the rank of A is decreased, we cannot use these methods. *Singular Value Decomposition*, SVD, is the most robust method for obtaining pseudo-inverse but it is computationally heavier than others.

3.3 Proposed Methods in Literature

3.3.1 Pseudo-inverse of Jacobian

In the year 1977, Liégeois [33] introduced a pseudo-inverse solution for the Jacobian matrix for kinematically redundant cases. He suggested solving joint rates with the following equation.

$$\dot{\boldsymbol{q}} = \mathcal{J}^{\dagger}\boldsymbol{t} + (\boldsymbol{I} - \mathcal{J}^{\dagger}\mathcal{J})\boldsymbol{z} \tag{3.11}$$

And defined $\boldsymbol{z}$ vector as follows.

$$\boldsymbol{z} = \alpha * \nabla_{\boldsymbol{q}}\, \boldsymbol{H} \tag{3.12}$$

Where α is a real scalar and $\nabla_{\boldsymbol{q}}\, \boldsymbol{H}$ is the gradient of a smooth function $\boldsymbol{H}(\boldsymbol{q})$. He stated that if $\boldsymbol{H}(\boldsymbol{q})$ function is properly chosen, solutions for $\dot{\boldsymbol{q}}$ can be optimized for avoiding obstacles, for obtaining minimum displacement at the joints and for avoiding mechanical limits or singularities.

In his proposed application, he used a six-axis robot and specified task as a circular path whose orientation is irrelevant. Thus, the operational space dimension is decreased to $o = 3$. Jacobian matrix dimensions are also decreased to 3x6 and linear system equations became an overdetermined system. He defined $\boldsymbol{H}(\boldsymbol{q})$ as a quadratic cost function for minimization of the deviations from the mean joint displacement. The motion of the robot stayed out of singular positions because $(\boldsymbol{I} - \mathcal{J}^{\dagger}\mathcal{J})\boldsymbol{z}$ solution had always tried to hold joint positions in the mean of joint limits.

This approach had been also used by other scholars with different $\boldsymbol{H}(\boldsymbol{q})$ functions. Moreover, a combination of different optimization criteria had been applied. This assists in finding solutions for optimization priority. Studies can be viewed for further information [34][14].

Unfortunately, this approach does not give a practical solution for passing through singular configurations. Because $\mathcal{J}$ could be singular still when the redundancy solutions are ineffective. Instead, this method locally optimizes robot posture to avoid singularity or satisfy other criteria.

3.3.2 Singularity-robust inverse of Jacobian

In the year 1986, Nakamura and Hanafusa [15], introduced Singularity-Robust Inverse of the Jacobian matrix instead of pseudo-inverse. They defined it as follows:

$$\boldsymbol{\mathcal{J}}^{*} = \boldsymbol{\mathcal{J}}^{T}(\boldsymbol{\mathcal{J}}\boldsymbol{\mathcal{J}}^{T} + \lambda^{2}\boldsymbol{I})^{-1} \tag{3.13}$$

Where λ is a damping factor. Even if $\boldsymbol{\mathcal{J}}$ is singular, the inverse above gives a feasible solution to overcome singularity. The critical point here is specifying λ value because high damping factors gives inaccurate solutions, whereas low damping factor gives impractically high joint rates. Hence, solutions for joint rates with this SR-Inverse matrix stay between exactness and feasibility. For defining the damping factor, SVD of Jacobian or manipulability measure of the robot, that is given by Yoshikawa [13], can be used.

A contribution to this approach had been done by Chiaverini et al. [35] by adding weighting matrices to increase accuracy in specified directions.

$$\boldsymbol{\mathcal{J}}^{\#} = \boldsymbol{W}^{T}\boldsymbol{\mathcal{J}}^{T}(\boldsymbol{\mathcal{J}}\boldsymbol{W}\boldsymbol{W}^{T}\boldsymbol{\mathcal{J}}^{T} + \lambda^{2}\boldsymbol{I}) \tag{3.14}$$

Where $\boldsymbol{W}$ is a 6x6 matrix and satisfies the following condition:

$$\tilde{\boldsymbol{t}} = \boldsymbol{W}\boldsymbol{t} \tag{3.15}$$

$\tilde{\boldsymbol{t}}$ is the specified twist vector whose infeasible directions are lowered by weighting matrix $\boldsymbol{W}$.

This approach requires mostly user-defined parameters without foreseeing system behaviors. It is difficult to standardize this approach for different applications with different robots. Ultimately, a specified accuracy is not guaranteed.

3.3.3 The degenerate axis

In the year 1987, Aboaf and Paul [18] stated that any rotation through the degenerate axis cannot be satisfied by wrist joints in wrist singular configurations. They suggested a solution with eliminating rotation through the degenerate axis and compensating position error that is caused by this rotation.

In that purpose, they specially transformed the Jacobian matrix into the fifth link frame. The link frames were organized by the DH convention. Thus, the degenerate

direction always stayed at x-direction of that frame. After that, they dropped the fourth row of the linear equation. Moreover, they dropped the fourth column of the transformed Jacobian to obtain a square matrix which is always non-singular in wrist singular configuration. In the end, they obtained a linear system similar to the below.

$$\begin{bmatrix} \dot{q}_1 \\ \dot{q}_2 \\ \dot{q}_3 \\ \dot{q}_5 \\ \dot{q}_6 \end{bmatrix} = \boldsymbol{\mathcal{J}}_T^{-1} \begin{bmatrix} \dot{x}_T \\ \dot{y}_T \\ \dot{z}_T \\ w_{y_T} \\ w_{z_T} \end{bmatrix} \tag{3.16}$$

Where $\boldsymbol{\mathcal{J}}_T$ defines 5x5 transformed Jacobian. By this system, they cannot determine $\dot{q}_4$. Hence, they give it a constant value as $\dot{q}_{4_{max}}$ and then removed its effect on twist vector by using the fourth column of typical Jacobian. In control policy, they used this algorithm whenever resolved-rate solution for wrist joint rates exceeds their limit values. This resulted in obtaining a solution for passing through the wrist singularity case without any position error but just a reasonable orientation error.

Nevertheless, this approach contains some deficiencies such as immobilizing joint-4 rate for the whole path near wrist singularity and stipulating near wrist singular area as exceeding joint velocity limits. Firstly, immobilizing the joint-4 rate most probably causes increasing in orientation error because path requirements can vary. Secondly, before exceeding joint velocity limits, joint acceleration limits would be violated near singularities. Hence, developments are required to obtain less orientation error and to determine near wrist singular area.

3.3.4 Augmented Jacobian with virtual joint

If we turn back our cause, we have selected six axes industrial robots that are not intrinsically redundant manipulators. Moreover, we are representing our task with a twist vector of 6x1. Thus, we require to create redundancy firstly in the task space.

5-DOF continuous tasks can be represented in the instantaneous TCP frame with 5 coordinates, namely, a combination of Cartesian and spherical coordinates. However, we have created our models with respect to the inertial frame called base frame and all of the orientation representation with quaternions. While the EE pose vector has a size of 7x1, we have managed to transform it into a twist vector of 6x1. We have created our geometric Jacobian of the manipulator by using this transformation. Henceforth,

relating a 5 coordinate represented task with our mathematical robot model seems unlikely.

A remedy to this problem is given by Baron [20], by adding a virtual joint on the robot side. This would be resulting in an additional DOF on the robot side while representing a 5-DOF task with 6 parameters. Hence, redundancy can be provided. He called this manipulation as augmentation and obtain an Augmented Jacobian matrix as follows.

$$\boldsymbol{\mathcal{J}_a} = [\,\boldsymbol{\mathcal{J}} \quad \boldsymbol{j}_a] \tag{3.17}$$

Where $\boldsymbol{j}_a$ is the column vector of the virtual joint. Also, the joint rates vector would be formed as below.

$$\dot{\boldsymbol{q}}_a = [\dot{\boldsymbol{q}} \quad \dot{q}_a]^T \tag{3.18}$$

Where $\dot{q}_a$ is the virtual joint rate. Then, he solved joint rates as follows.

$$\dot{\boldsymbol{q}}_a = \boldsymbol{\mathcal{J}}_a^{\dagger}\,\boldsymbol{t} \tag{3.19}$$

The problem here is any motion created by a virtual joint affects the twist vector. Thus, the instantaneous solution of the Jacobian valid for the previous twist vector that is not affected by the virtual joint. However, instantaneously updating the twist vector was not mentioned by Baron which is ambiguous.

The development of this approach has been done by Huo and Baron (2005) [31], by decomposing twist vector separately into task-related and redundancy-related twists which are given below respectively.

$$\boldsymbol{t} = [\boldsymbol{t}]_{\mathcal{T}} + [\boldsymbol{t}]_{\mathcal{T}^{\perp}} \tag{3.20}$$

They used the projection matrices that we have shown in **Section 2.1**. They defined the projection line as the symmetry axis of the welding electrode. Open expressions of the terms above are given as follows.

$$[\boldsymbol{t}]_{\mathcal{T}} = \begin{bmatrix} \boldsymbol{v}_{TCP} \\ (\boldsymbol{I} - \boldsymbol{e}\boldsymbol{e}^T)\boldsymbol{\omega}_{TCP} \end{bmatrix} \tag{3.21}$$

$$[\boldsymbol{t}]_{\mathcal{T}^{\perp}} = \begin{bmatrix} \boldsymbol{0} \\ \boldsymbol{e}\boldsymbol{e}^T\boldsymbol{\omega}_{TCP} \end{bmatrix} \tag{3.22}$$

Then, they suggested the following solution.

$$\dot{\boldsymbol{q}}_a = \boldsymbol{\mathcal{J}}_a^{\dagger}[\boldsymbol{t}]_{\mathcal{T}} + \left(\boldsymbol{I} - \boldsymbol{\mathcal{J}}_a^{\dagger}\boldsymbol{\mathcal{J}}_a\right)[\boldsymbol{t}]_{\mathcal{T}^{\perp}}\boldsymbol{k} \tag{3.23}$$

Its homogenous solution is not affected by the virtual joint.

In addition to these studies, in recent years, Geu Flores et al. published papers for passing through singularity with the virtual redundant axis method [41][42]. They had created augmented jacobians where virtual joints placed in the direction of degenerate axes. They had obtained solutions for virtual joints that are not applied to the real robot when the robot is near or at singularities. They also had added weighting matrices to the models to penalize virtual joint values when the robot is out of singular configurations. Thereby, they had managed to obtain passing through algorithm by using augmented Jacobian which has both position and orientation tracking errors.

3.4 Our Method: Infeasible Rotation Compensation

3.4.1 Motivation

By regarding proposed methods, we have gained consciousness for Jacobian related methods is not guaranteeing a specified path accuracy when passing through wrist singularity. Using the null space of the Jacobian matrix for redundancy with gradient functions seems somewhat unpredictable case to case. Henceforth, we prefer using a task related method to minimize the effects of wrist singularity similar to the study of Aboaf and Paul [18]. However, we will interpret their method with our mathematical models. In addition to that, we will use a redundant axis to overcome orientation errors. If redundant axis rotation is feasible enough, we will completely eliminate wrist singularity effects on intrinsically non-redundant 6 axis robots in a sense of path accuracy.

3.4.2 The degenerate or infeasible rotation axis

First of all, let us define the infeasible rotation axis in the kinematics of the robot wrist. It is the same axis that is defined by Aboaf and Paul.

As can be seen in **Figure 3.1**, we have defined a vector through the infeasible axis of rotation.

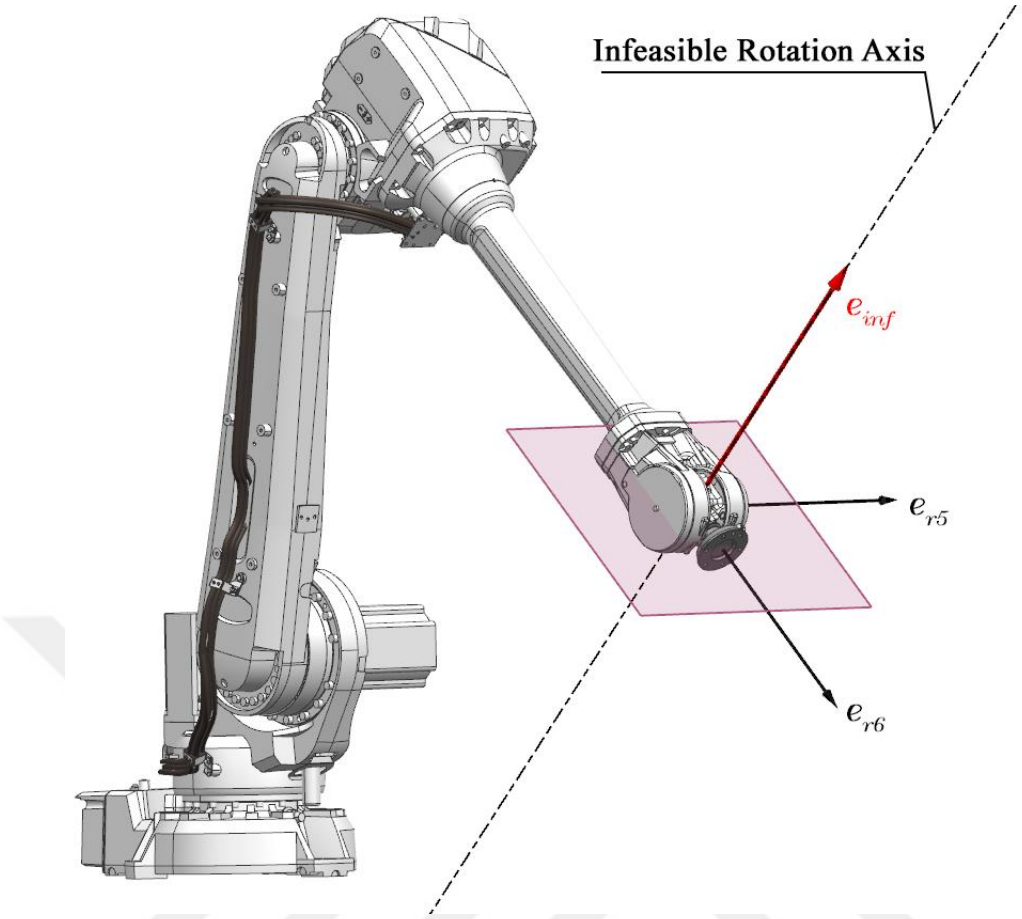

Figure 3.1 : Infeasible rotation axis (Robot: IRB-4600 of ABB).

In order to obtain that vector, we use rotated joint axis vectors which the robot Jacobian matrix contains as below.

$$\boldsymbol{e}_{inf} = \boldsymbol{e}_{r6} \times \boldsymbol{e}_{r5} \tag{3.24}$$

Then, we are going to create infeasible and feasible projection matrices of these unit vectors as follows.

$$\boldsymbol{P}_{inf} = \left[\boldsymbol{e}_{inf}\boldsymbol{e}_{inf}^{T}\right] \tag{3.25}$$

$$\boldsymbol{P}_{fsb} = \left[\boldsymbol{I} - \boldsymbol{e}_{inf}\boldsymbol{e}_{inf}^{T}\right] \tag{3.26}$$

Then we are going to define infeasible and feasible instantaneous angular velocities of wrist joints that are mentioned in **Section 2.3.3**.

$$\boldsymbol{\omega}_{wr} = [\boldsymbol{\omega}_{wr}]_{inf} + [\boldsymbol{\omega}_{wr}]_{fsb} \tag{3.27}$$

Where:

$$[\boldsymbol{\omega}_{wr}]_{inf} = \boldsymbol{P}_{inf}\boldsymbol{\omega}_{wr} \tag{3.28}$$

$$[\boldsymbol{\omega}_{wr}]_{fsb} = \boldsymbol{P}_{fsb}\boldsymbol{\omega}_{wr} \tag{3.29}$$

In order to get feasible joint rate solutions from ill-conditioned $\boldsymbol{\mathcal{J}}_{22}$ by regarding Eq. (2.227), it is required to be eliminated $[\boldsymbol{\omega}_{wr}]_{inf}$ from $\boldsymbol{\omega}_{wr}$. To do this we have to re-construct task with respect to $[\boldsymbol{\omega}_{wr}]_{inf}$. If it is satisfied, newly created $\boldsymbol{\omega}_{wr}$ would stay on the feasible plane.

After it is applied for velocity level kinematics, it is required to be move on acceleration kinematics because wrist singularity also affects joint acceleration behaviors. By doing so, let us define infeasible and feasible instantaneous angular accelerations of wrist joints:

$$[\dot{\boldsymbol{\omega}}_{wr}]_{inf} = \boldsymbol{P}_{inf}\dot{\boldsymbol{\omega}}_{wr} \tag{3.30}$$

$$[\dot{\boldsymbol{\omega}}_{wr}]_{fsb} = \boldsymbol{P}_{fsb}\dot{\boldsymbol{\omega}}_{wr} \tag{3.31}$$

If we directly remove these infeasible rotation vectors in WCP, we get a huge position error in TCP. So, it would not be suitable for industrial applications. We suggest compensating the vectors in TCP which means re-constructing the task according to the infeasible rotation axis. Two methods of compensation are going to be shown in this study.

3.4.3 Compensation through the parallel infeasible rotation axis

The first method we are suggesting is compensation through a parallel infeasible rotation axis by applying the instantaneous angular velocity vector to the task. Here, we first solve the inverse kinematic model with respect to the given initial twist vector. Then, we calculate infeasible instantaneous angular velocity, $[\boldsymbol{\omega}_{wr}]_{inf}$, for the initial twist vector which we desire to make feasible when the robot is near or at wrist singularity. For the visualization of the compensation vector, one can see **Figure 3.2**.

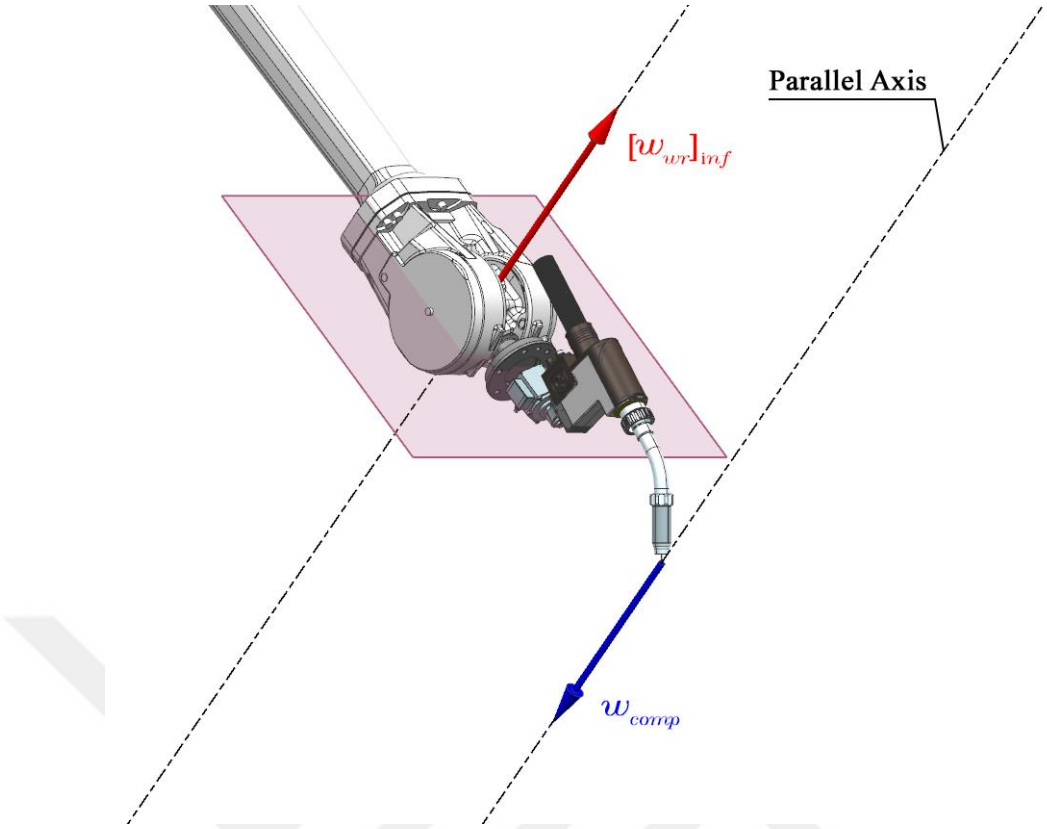

Figure 3.2 : Compensation through parallel infeasible rotation axis.

Compensation vector that is being applied is not at WCP, which means its magnitude would be different than $[\boldsymbol{\omega}_{wr}]_{inf}$ even if it is at the exact opposite direction of $[\boldsymbol{\omega}_{wr}]_{inf}$. Hence, the magnitude of the compensation vector should be calculated by transforming it to the WCP from TCP just like in **Figure 2.17**. Transformation of the compensation vector would correspond not just rotation but translation at WCP. This translation would have resulted in additional shoulder joints motion that is needed to be specified. As we discussed in **Section 2.3** when we are obtaining $\boldsymbol{\omega}_{wr}$, we determine wrist joints velocities relatively to shoulder joints. Hence, additional shoulder joints velocities would also create additional angular velocity vector. Taking into account all of these, we define a 3x3 compensation matrix in the equation below to find the angular velocity vector in the infeasible axis that is generated by the additional shoulder joints motion.

$$\boldsymbol{M}_{comp} = -\boldsymbol{P}_{inf}\boldsymbol{\mathcal{J}}_{21}\boldsymbol{\mathcal{J}}_{11}^{-1}\boldsymbol{P}_{tr} \tag{3.32}$$

This matrix contains a projection matrix and 3 matrices that we have discussed in **sub-Section 2.3.3**. When it is multiplied by any TCP rotation vector at the right-hand side, it gives shoulder joints angular velocity vector in the infeasible axis of rotation.

We have not determined the magnitude of the compensation vector yet. To solve it, we firstly define the direction of the compensation vector, as a unit vector, as given below.

$$\boldsymbol{e}_{compVel} = -\frac{[\boldsymbol{\omega}_{wr}]_{inf}}{\left\|[\boldsymbol{\omega}_{wr}]_{inf}\right\|} \tag{3.33}$$

Then, we want to calculate the response of the shoulder joints for this unit vector by using a compensation matrix in order to obtain a ratio between them.

$$\left[\boldsymbol{e}_{compVel}\right]_{sh} = \boldsymbol{M}_{comp}\, \boldsymbol{e}_{compVel} \tag{3.34}$$

The magnitude of the obtained vector may not be one, whereas, we called it as $\boldsymbol{e}$ because it is related to unit vector response.

Then, we define a ratio parameter called *compensation ratio for velocity* which combines the direction of compensation vector with shoulder joints response to it.

$$k_{compVel} = \frac{1}{\left\|\boldsymbol{e}_{compVel} + \left[\boldsymbol{e}_{compVel}\right]_{sh}\right\|} \tag{3.35}$$

This ratio parameter has been defined because it is going to be used for finding the required compensation vector according to infeasible instantaneous angular velocity, $[\boldsymbol{\omega}_{wr}]_{inf}$. After finding the compensation ratio, we can calculate the compensation vector for the instantaneous angular velocity vector as follows.

$$\boldsymbol{\omega}_{comp} = -k_{compVel}[\boldsymbol{\omega}_{wr}]_{inf} \tag{3.36}$$

In the velocity level, if it were possible to add this compensation vector to the task, it would eliminate the infeasible instantaneous angular velocity. However, as we discussed in the task re-construction section, we define our task models at the acceleration level. Hence, it will not be possible to add this compensation directly to the task velocity. To get into the acceleration level, we define the following parameter by using the sample time and velocity compensation vector.

$$\dot{\boldsymbol{\omega}}_{compVel} = \boldsymbol{\omega}_{comp} \,/\, \Delta t \tag{3.37}$$

After this calculation, we can add a portion of this vector to our new twist rate vector with respect to acceleration limits. Before that, let us make similar calculations for acceleration level compensation.

It is also required to compensate accelerations because of the ill-conditioned $\boldsymbol{\mathcal{J}}_{22}$. Thus, we need to calculate the compensation vector for acceleration according to infeasible instantaneous angular acceleration, $[\dot{\boldsymbol{\omega}}_{wr}]_{inf}$, for the initial twist rate vector. Then, we can solve the acceleration compensation vector with a similar approach by creating similar parameters for acceleration as follows.

$$\boldsymbol{e}_{compAcc} = -\frac{[\dot{\boldsymbol{\omega}}_{wr}]_{inf}}{\left\|[\dot{\boldsymbol{\omega}}_{wr}]_{inf}\right\|} \tag{3.38}$$

$$\left[\boldsymbol{e}_{compAcc}\right]_{sh} = \boldsymbol{M}_{comp}\, \boldsymbol{e}_{compAcc} \tag{3.39}$$

$$k_{compAcc} = \frac{1}{\left\|\boldsymbol{e}_{compAcc} + \left[\boldsymbol{e}_{compAcc}\right]_{sh}\right\|} \tag{3.40}$$

$$\dot{\boldsymbol{\omega}}_{compAcc} = -k_{compAcc}[\dot{\boldsymbol{\omega}}_{wr}]_{inf} \tag{3.41}$$

After solving velocity and acceleration compensation vectors, we define the actual compensation vector that will be used in task re-construction, $\dot{\boldsymbol{\omega}}_{comp}$. We are combining both components in one as given below.

$$\dot{\boldsymbol{\omega}}_{comp} = \dot{\boldsymbol{\omega}}_{compAcc} + k_{accUsage} * \dot{\boldsymbol{\omega}}_{compVel} \tag{3.42}$$

Where $k_{accUsage}$ determines how much acceleration can be used to for the infeasible velocity compensation with respect to robot joint acceleration limits. It is a function of a user-defined parameter, $\eta_{accUsage}$, which is a ratio concerning real wrist joint acceleration limits of the robot. The calculation is given below.

$$k_{accUsage} = \eta_{accUsage} * \frac{\ddot{q}_{4max} * |\Delta_{22}| * \Delta t}{\left\|[\boldsymbol{\omega}_{wr}]_{inf}\right\|} \tag{3.43}$$

Let acceleration limits of wrist joints are given respectively as $\ddot{q}_{4max}, \ddot{q}_{5max}, \ddot{q}_{6max}$. Generally, $\ddot{q}_{4max}$ is the lowest acceleration limit of industrial robots, hence, we have

given formulation by concerning it. However, on the contrary case, it can be replaced with the lowest joint acceleration limit.

After the actual compensation vector has been found, the initial task twist rate can be compensated as follows.

$$[\dot{\boldsymbol{t}}_{TCP}]_{new} = [\dot{\boldsymbol{t}}_{TCP}]_{initial} + \begin{bmatrix} \mathbf{0}_{3x1} \\ \dot{\boldsymbol{\omega}}_{comp}(k) \end{bmatrix} \tag{3.44}$$

In programming representation, the instantaneous angular acceleration would be re-assigned by adding the compensation vector to it as follows.

$$\dot{\boldsymbol{\omega}}_{TCP}(k) \leftarrow \dot{\boldsymbol{\omega}}_{TCP}(k) + \dot{\boldsymbol{\omega}}_{comp}(k) \tag{3.45}$$

Then, it is required to solve the inverse kinematics models again with the new assignment. In solutions, all the joint velocities and accelerations would be feasible and the desired position displacement would be satisfied. Whereas, there would be orientation errors in the amount of compensation. Thus, this method provides almost zero position error but a reasonable orientation error in the wrist singular areas. We believe that compensating through the exact opposite direction of infeasible wrist rotation would give us a minimum amount of orientation error rather than other direction compensations if we are not considering redundancy. The integration of the twist rate can be done by using Eq. (2.297).

It is also possible to collect information for the amount of compensation that has been done by accumulating the compensation vectors for each step as follows.

$$\dot{\boldsymbol{\omega}}_{compTotal} = \sum_{k=1}^{m} \dot{\boldsymbol{\omega}}_{comp}(k) \tag{3.46}$$

Where m can be selected any step after passing the neighborhood of wrist singular configuration. The accumulation of the compensation vectors corresponds to the orientation error if they would be integrated and it can be removed by the manipulator by reversing the whole process. Nevertheless, in a certain amount of time the path would be tracked with orientation errors.

3.4.4 Compensation through the redundant rotation axis

To overcome the orientation error, the redundant axis of rotation can be used. By the definition, compensation through this axis cannot be qualified as an orientation error. Instead, it can assist us in tracking path with lesser or zero orientation error. The problem is the redundant axis and infeasible axis may be perpendicular to each other. In that case, no rotation through the redundant axis fulfills infeasible wrist rotation. Moreover, in near perpendicular case, rotation through the redundant axis may be a very high amount. So, even though $\boldsymbol{\omega}_{wr}$ stay in the feasible plane, joint rates may not be feasible because of extreme redundant axis rotation. For those cases, we prefer using the first suggested method to obtain minimum orientation error.

The visualization of the redundant axis and compensation vector through this axis is given in **Figure 3.3**.

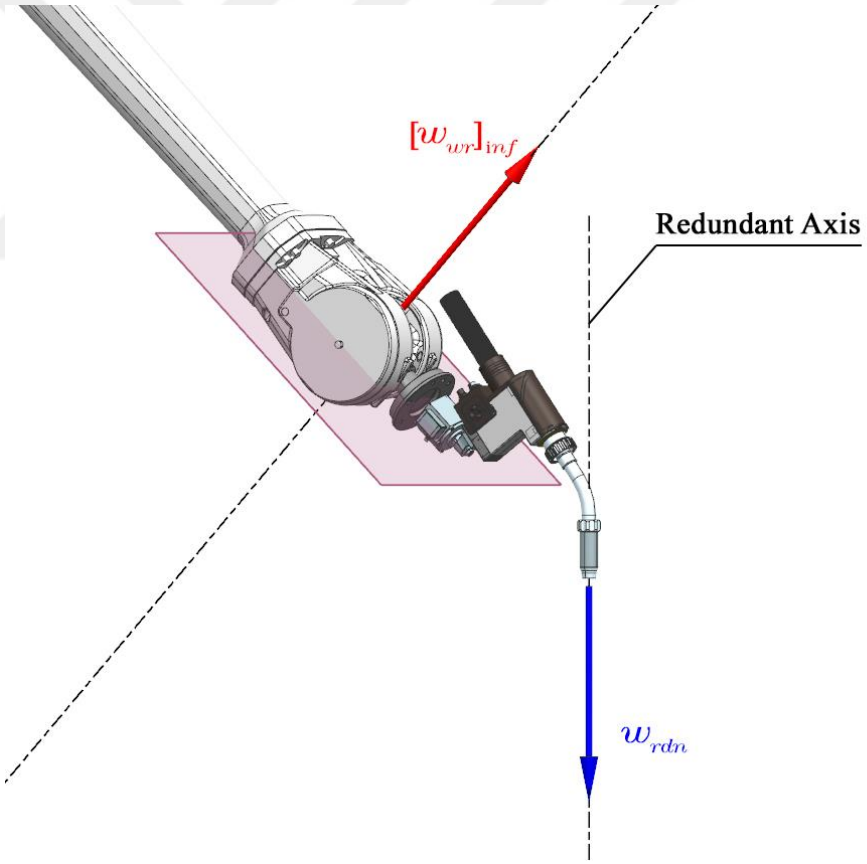

Figure 3.3 : Compensation through the redundant axis of rotation.

Firstly, we require a mathematical definition of the redundant axis of rotation. In our thesis, we are going to use an arc welding torch whose dimension parameters are given in **Figure 3.4**.

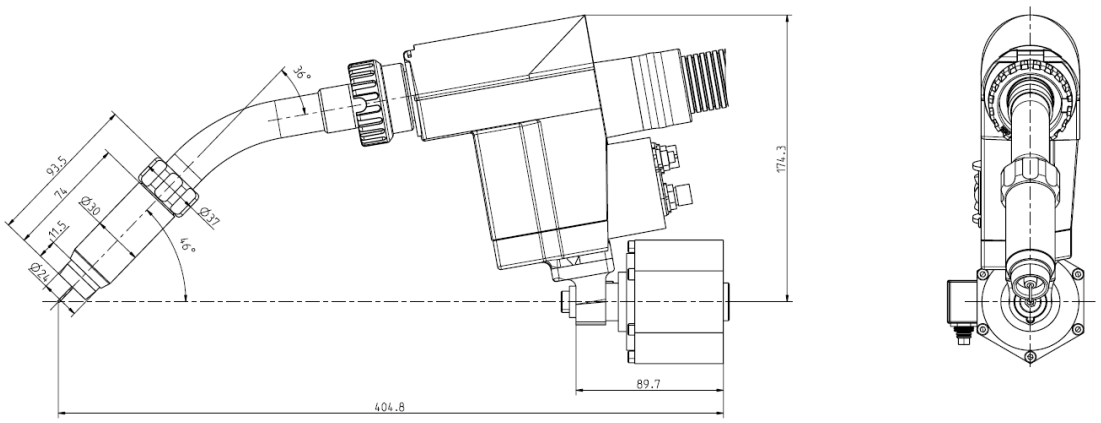

Figure 3.4 : Fronius RA 7000 G36 CMT torch (courtesy of Fronius Gmbh).

The redundant axis of arc welding torches generally stays on the same plane with the sixth joint axis of industrial robots with an angle. The angle can vary between 22°-60° concerning the torch type. In our study, we are using 44° angled torch with respect to the sixth joint axis. Hence, it always has a perpendicular component in that plane with respect to the sixth joint axis which can be used for infeasible axis compensation.

For other applications, position or orientation of the redundant axis can vary according to the intention but making the change in redundant axis is expected to make a change in robot WCP or tool orientation, somehow.

For our case, we specify the redundant axis vector as the z-axis of the TCP frame. So, we define it as follows:

$$\boldsymbol{e}_{rdn} = \boldsymbol{R}_{TCP} \begin{bmatrix} 0 \\ 0 \\ 1 \end{bmatrix} \tag{3.47}$$

Then, we define its projection into the infeasible rotation axis.

$$[\boldsymbol{e}_{rdn}]_{inf} = \boldsymbol{P}_{inf}\, \boldsymbol{e}_{rdn} \tag{3.48}$$

This projection vector provides information about how efficient the redundant rotation axis for compensating the infeasible axis of rotation. Henceforth, to use this method, we put a basic criterion by checking the magnitude of this vector. If it satisfies the criterion, the method may be desired to be used.

$$\eta_{rdn} > \left\| [\boldsymbol{e}_{rdn}]_{inf} \right\| \tag{3.49}$$

Where η_{rdn} is a user-defined parameter called redundancy ratio.

For defining the direction of rotation through the redundant axis, we can use direction cosines between $[\boldsymbol{\omega}_{wr}]_{inf}$ and $[\boldsymbol{e}_{rdn}]_{inf}$. The result would be 1 or -1 depending on which side projection of redundant axis vector drops. We desire it to be in the opposite direction of $[\boldsymbol{\omega}_{wr}]_{inf}$. The same conditions are also valid for acceleration. Thus, we define the sign parameter for velocity and acceleration as follows.

$$\boldsymbol{e}_{rdnSignVel} = -\frac{[\boldsymbol{\omega}_{wr}]_{inf}}{\left\| [\boldsymbol{\omega}_{wr}]_{inf} \right\|} \cdot \frac{[\boldsymbol{e}_{rdn}]_{inf}}{\left\| [\boldsymbol{e}_{rdn}]_{inf} \right\|} \tag{3.50}$$

$$\boldsymbol{e}_{rdnSignAcc} = -\frac{[\dot{\boldsymbol{\omega}}_{wr}]_{inf}}{\left\| [\dot{\boldsymbol{\omega}}_{wr}]_{inf} \right\|} \cdot \frac{[\boldsymbol{e}_{rdn}]_{inf}}{\left\| [\boldsymbol{e}_{rdn}]_{inf} \right\|} \tag{3.51}$$

Then, we are going to follow the steps similar to the first method, we would look into shoulder response for velocity and acceleration as follows.

$$[\boldsymbol{e}_{rdnVel}]_{sh} = \boldsymbol{M}_{comp}\, \boldsymbol{e}_{rdnSignVel}\, \boldsymbol{e}_{rdn} \tag{3.52}$$

$$[\boldsymbol{e}_{rdnAcc}]_{sh} = \boldsymbol{M}_{comp}\, \boldsymbol{e}_{rdnSignAcc}\, \boldsymbol{e}_{rdn} \tag{3.53}$$

Now, we are going to define two ratios of the redundant rotation axis compensation.

$$k_{rdnVel} = \boldsymbol{e}_{rdnSignVel} \frac{\left\| [\boldsymbol{\omega}_{wr}]_{inf} \right\|}{\left\| \boldsymbol{e}_{rdnSignVel} [\boldsymbol{e}_{rdn}]_{inf} + [\boldsymbol{e}_{rdnVel}]_{sh} \right\|} \tag{3.54}$$

$$k_{rdnAcc} = \boldsymbol{e}_{rdnSignAcc} \frac{\left\| [\boldsymbol{\omega}_{wr}]_{inf} \right\|}{\left\| \boldsymbol{e}_{rdnSignAcc} [\boldsymbol{e}_{rdn}]_{inf} + [\boldsymbol{e}_{rdnAcc}]_{sh} \right\|} \tag{3.55}$$

Consequently, we can obtain the instantaneous angular velocity and acceleration vectors of redundant rotation axis compensation.

$$\boldsymbol{\omega}_{rdnVel} = k_{rdnVel}\, \boldsymbol{e}_{rdn} \tag{3.56}$$

$$\dot{\boldsymbol{\omega}}_{rdnAcc} = k_{rdnAcc}\, \boldsymbol{e}_{rdn} \tag{3.57}$$

However, similar to the first method, we require combining velocity and acceleration components of compensation vectors into a single vector for task re-construction. The final formulation for the redundant rotation axis compensation vector is given below.

$$\dot{\boldsymbol{\omega}}_{rdn} = \dot{\boldsymbol{\omega}}_{rdnAcc} + k_{compUsage} * \dot{\boldsymbol{\omega}}_{rdnVel} \tag{3.58}$$

Where the velocity component is calculated as follows.

$$\dot{\boldsymbol{\omega}}_{rdnVel} = \boldsymbol{\omega}_{rdnVel} / \Delta t \tag{3.59}$$

After that, the initial task twist rate can be compensated by adding the redundant rotation axis compensation vector as follows.

$$[\dot{\boldsymbol{t}}_{TCP}]_{new} = [\dot{\boldsymbol{t}}_{TCP}]_{initial} + \begin{bmatrix} \mathbf{0}_{3x1} \\ \dot{\boldsymbol{\omega}}_{rdn}(k) \end{bmatrix} \tag{3.60}$$

3.4.5 Determining boundaries of wrist singular areas

To use methods above for a robot that is tracking path should move into a wrist singular configuration or come into a specified closeness to that configuration. Still, this near wrist singular area is not well defined. Some scholars basically defined it as a distance of q_5 value to zero. Also, we know that in the neighborhood of wrist singularity, typical resolved-rate solutions becomes abnormally high. However, we had seen in different task simulations, when the wrist joints became abnormally high, the distances of q_5 value to zero was varying. Thus, these boundaries are highly dependent on the task.

Hence, it is required to locate the boundaries of wrist singular configurations before abnormally high joint solutions created. We can control increment in joint rate solutions by looking into joint acceleration solutions. However, there would be cases that joint velocity limits are violated while acceleration limits are not. Thus, we should check both velocity and acceleration limits at the same time to understand if we are near wrist singular configurations. Furthermore, it is possible to look into jerk values but for the sake of the thesis, we were not able to study for jerk models.

One can check the following conditions, in a simulation step, to determine how much joint acceleration and velocity required for the wrist rotations that are in the direction of the infeasible rotation axis.

$$\eta_{acc} > \frac{\left\| [\dot{\boldsymbol{\omega}}_{wr}]_{inf} \right\|}{|\Delta_{22}| * \ddot{q}_{4max}} \tag{3.61}$$

$$\eta_{vel} > \frac{\left\| [\boldsymbol{\omega}_{wr}]_{inf} \right\|}{|\Delta_{22}| * \dot{q}_{4max}} \tag{3.62}$$

Where, η_{acc} and η_{vel} are user-defined parameters, we call acceleration and velocity ratios, and can be thought of as thresholds for being in wrist singular area when any of the conditions are satisfied. For determining values, one can run an initial path-tracking simulation without using compensation by ignoring infeasible rotations to see the values of these parameters. After seeing results, one can specify the parameters for the thresholds and determine the wrist singular area approximately. An example can be seen in **Figure 3.5**.

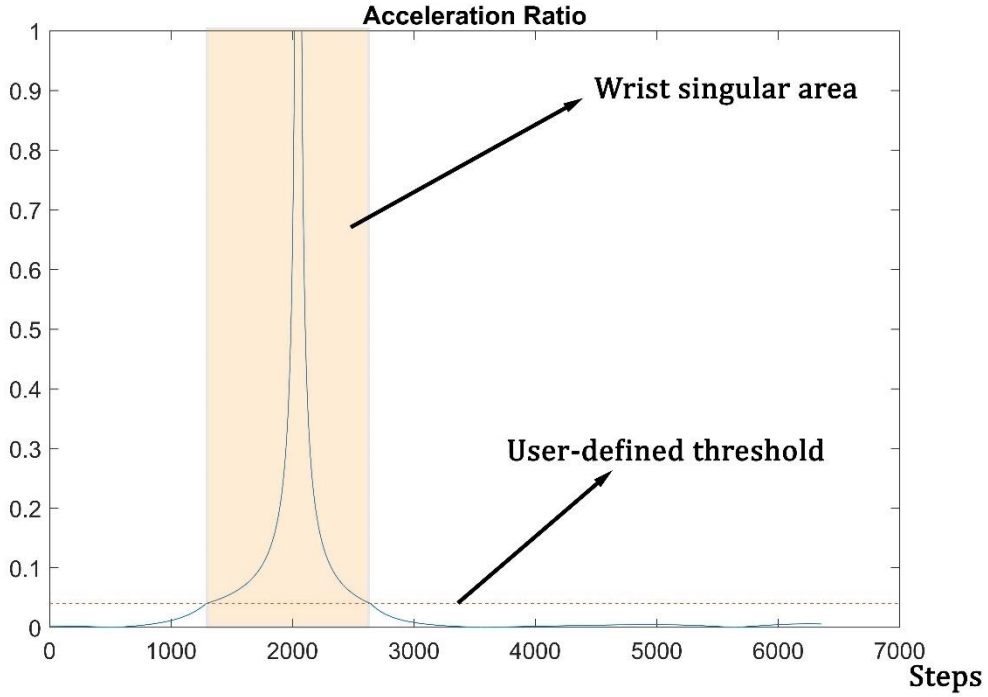

Figure 3.5 : Determining wrist singular area by using acceleration limits ratio.

After user-defined parameters specified, one can use them in the main simulations and check the results. The critical point here is the width of the determined wrist singular area. It should be large enough to eliminate the infeasible instantaneous angular velocities before the determinant of the $\boldsymbol{\mathcal{J}}_{22}$ becomes too small. Δ_{22} approaches to zero the more we are close to the exact singular point. In that point, $k_{compUsage}$ is also approaches to zero by taking account of Eq. (3.43) and hence, we cannot compensate

remaining infeasible angular velocities if any exists. Thus, the magnitude of the infeasible instantaneous angular velocity vector should be considered while determining wrist singular area boundaries.

4. PATH TRACKING AND SIMULATIONS

4.1 Path Tracking Algorithm

In this section, simulations results of our novel method will be given. Before that, let us introduce our path tracking algorithm to inform how the created mathematical models are going to be used.

First of all, we define robot parameters: link dimensions, joint limits, and sample time. Then, end-effector parameters are defined. After the robot side definitions, path planning is entirely done. By using the sample time and process speed; position increment, angle increment, and total step count should be calculated. Then, the discretized path models should be created from step 1 to H.

After those, we simply check the inverse geometrical model for the robot reachability of the path. We define suitable configurations by considering joint limits. In industrial applications, direct-shoulder elbow-up shoulder configuration which can be seen in **Figure 2.13** is generally preferred, and hence, we try to use the same. We also check if the wrist singularity occurs. If the wrist singularity exists, we try to determine its step to specify the center of the wrist singular area. Besides, we check if the transition between the two configurations is possible and check if those two configurations are suitable for the whole path.

The next stage of the algorithm would be determining the wrist singular area boundaries. In that purpose, we will solve the inverse kinematic models for just the feasible directions of the whole path. Even if it will not meet with our desired tracking needs at the end, results can give us the information on which step the infeasible rotation vectors exceed our threshold values. Moreover, we check if the selected threshold values are suitable for eliminating the infeasible angular velocities. We control the distance to the center of the wrist singular area and calculate how many steps do we need for eliminating the infeasible angular velocity vector. If everything is suitable, we are getting the main stage of the algorithm.

In the main stage of the algorithm, a loop is established for each step of the discretized path. The loop starts with solving joint velocities and accelerations with respect to the given twist and twist rate vector by using Jacobian and Jacobian rate matrices. Then, Infeasible projections are made for wrist angular velocity and acceleration vectors. Also, the redundant axis is defined. After that, we check whether the current step is in the wrist singular area boundaries. If not, we simply integrate the twist by using the twist rate and integrate joint parameters by using found velocity and acceleration solutions. These integrations are to be used for the next step. Then, we evaluate the forward geometrical model and Jacobian matrix by using joint parameters of the next step and get back to the start of the loop. If the robot is in the step of wrist singular area boundaries, we start with calculating required compensation vectors and matrices. Then, we check for redundancy if it can be used. After the decision, we change the current twist rate vector by adding the compensation vector. For the new twist rate, we solve the accelerations again. Then, we integrate twist and joint parameters of the next step by using found joint accelerations. Moreover, total compensation applied to the twist rate is accumulated in the algorithm. Whenever the step is out wrist singular area, it tries to remove this accumulated compensation from the task with respect to the acceleration usage ratio to minimize orientation error.

In general terms, the algorithm flowchart is given in **Figure 4.1** to summarize how we are doing the simulations. It can be run in any programing language or software to generate robot joint trajectories for desired path tracking.

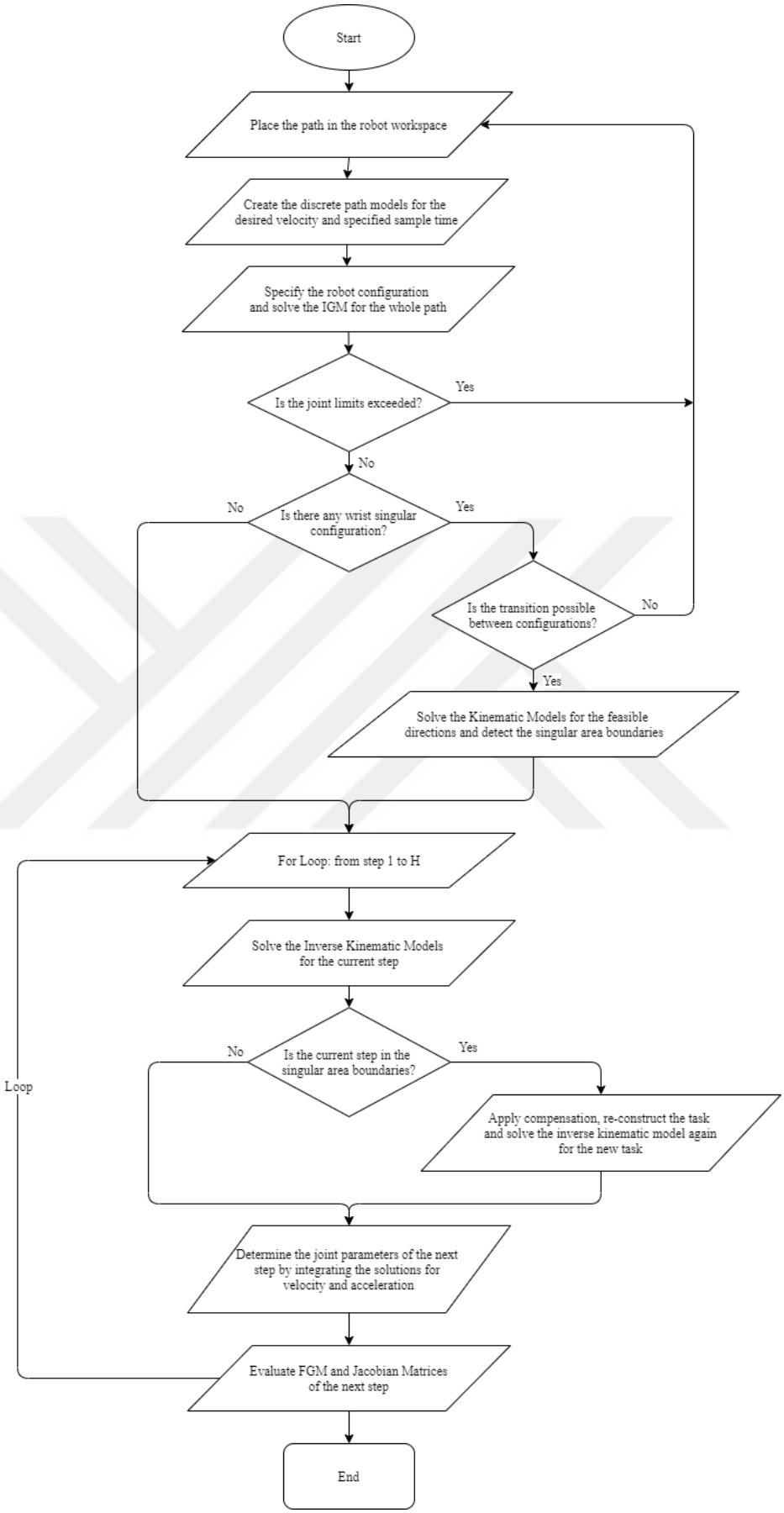

Figure 4.1 : Flowchart of path tracking algorithm for simulations.

4.2 Generated Trajectories for Example Paths

In this sub-section, we are going to establish paths for our robot to track and then, show simulations results. Let us give robot parameters which we are using.

Robot model: ABB IRB-4600 40 kg payload with 2.55 m reach.

Link parameters: d1:0.495, d2:1.095, d3:1.27, d4:0.135, a1:0, a2:0.175, a3:0.175 (all values in meter units).

Joint Limits: Joint parameter and velocity limits are specified by the manufacturer which can be seen in **Figure 4.2**.

Movement

Axis movement	Working range	Axis max. speed
Axis 1 rotation	+180° to -180°	175°/s
Axis 2 arm	+150° to -90°	175°/s
Axis 3 arm	+75° to -180°	175°/s
Axis 4 wrist	+400° to -400°	250° (360° for IRB 4600-20/2.50)
Axis 5 bend	+120° to -125°	250° (360° for IRB 4600-20/2.50)
Axis 6 turn	+400° to -400°	360° (500° for IRB 4600-20/2.50)

Figure 4.2 : IRB-4600 40/2.55 joint limits (courtesy of ABB).

Joint acceleration limits are not mentioned. Hence, we have assumed they are 10 times of velocity limits.

Sample Time: It is not specified also. By assumption, we have taken it 1 ms.

Now, we are going to define end-effector parameters as follows.

End-effector parameters:

$$p_{tx} = 0.4248, \qquad p_{ty} = p_{tz} = 0,$$

$$\boldsymbol{R}_t = \boldsymbol{R}(-0.7679, [0 \quad 1 \quad 0]^{\mathrm{T}})$$

After these, we are going to define paths that we are going to track. All of them will be on the reference plane which is given in **Figure 4.3**.

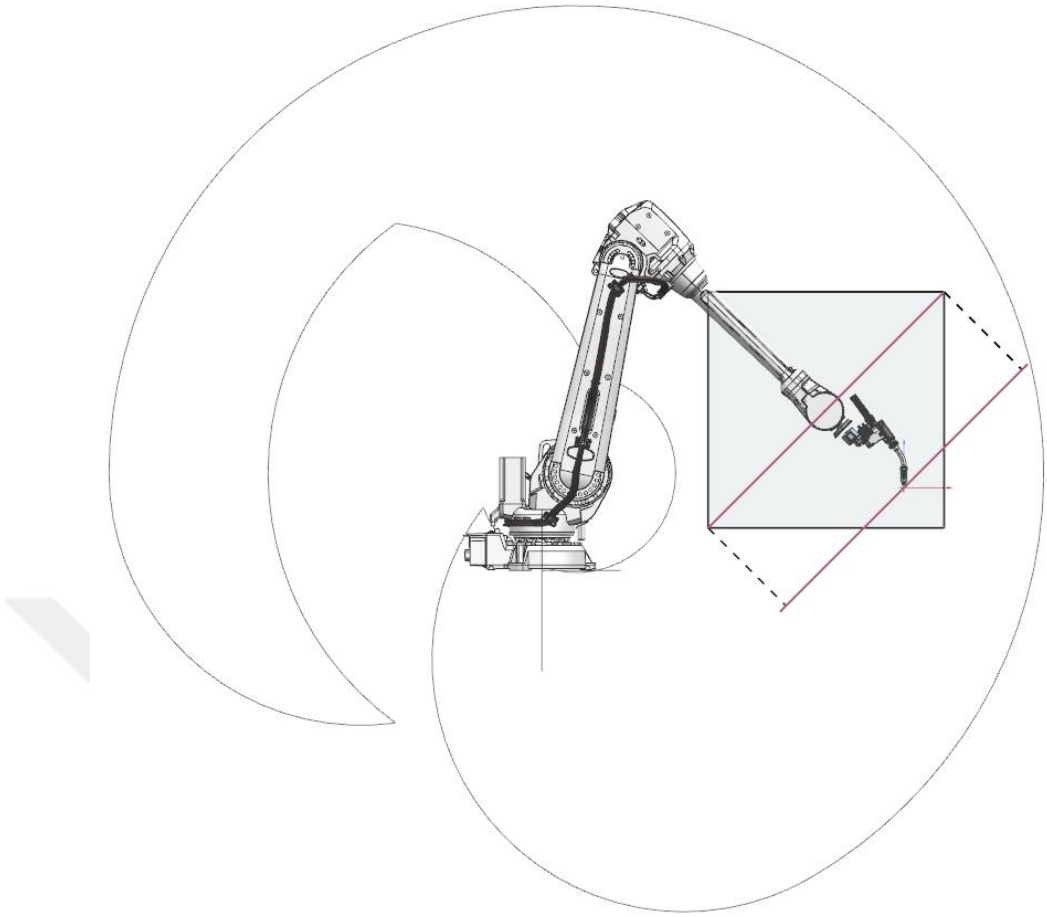

Figure 4.3 : Reference plane for establishing paths.

We are going to give just parameters for the models. To use these parameters, please see **Section 2.4**.

Linear Paths:

$$\boldsymbol{p}_a = [1.5576 \quad -0.5 \quad 0.138]^{\mathrm{T}}$$

$$\boldsymbol{p}_b = [2.1233 \quad 0.5 \quad 0.7037]^{\mathrm{T}}$$

Without Rotation: $\bar{\boldsymbol{q}}_a = [1 \quad 0 \quad 0 \quad 0]^{\mathrm{T}}, \qquad \theta_{rot} = 0.$

Rotation about x-axis: $\bar{\boldsymbol{q}}_a = [0.7071 \quad 0.7071 \quad 0 \quad 0]^{\mathrm{T}}, \qquad \theta_{rot} = -\pi$

Rotation about y-axis: $\bar{\boldsymbol{q}}_a = [0.8660 \quad 0 \quad -0.5 \quad 0]^{\mathrm{T}}, \qquad \theta_{rot} = 2\pi/3$

Rotation about z-axis: $\bar{\boldsymbol{q}}_a = [0.7071 \quad 0 \quad 0 \quad -0.7071]^{\mathrm{T}}, \qquad \theta_{rot} = \pi$

Circular Paths:

$$\boldsymbol{p}_a = [1.4434 \quad -0.3659 \quad 0.0238]^{\mathrm{T}}$$

$$\boldsymbol{p}_b = [1.8404 \quad 0 \quad 0.4208]^{\mathrm{T}}$$

$$\boldsymbol{p}_c = [1.4434 \quad 0.3659 \quad 0.0238]^{\mathrm{T}}$$

$$r = 0.4$$

$$\boldsymbol{e}_{circle} = [-0.7071 \quad 0 \quad 0.7071]^{\mathrm{T}}$$

Arc Rotation: $\bar{\boldsymbol{q}}_a = [0.8792 \quad 0.3369 \quad 0 \quad -0.3369]^{\mathrm{T}}, \quad \theta_{arc} = 3.9727$

Rotation about x-axis: $\bar{\boldsymbol{q}}_a = [0.9397 \quad 0.3420 \quad 0 \quad 0]^{\mathrm{T}}, \quad \theta_{rot} = -2\pi/3$

Rotation about y-axis: $\bar{\boldsymbol{q}}_a = [0.9221 \quad 0 \quad -0.3869 \quad 0]^{\mathrm{T}}, \quad \theta_{rot} = \theta_{arc}/3$

Rotation about z-axis: $\bar{\boldsymbol{q}}_a = [0.8660 \quad 0 \quad 0 \quad -0.5]^{\mathrm{T}}, \quad \theta_{rot} = \pi$

All of the process speeds are going to be taken as 0.25 m/s.

All of the user-defined parameters are given below unless otherwise indicated.

$$\eta_{accUsage} = \%5, \quad \eta_{acc} = \%5, \quad \eta_{vel} = \%50, \quad \eta_{rdn} = \%100$$

We are not going to use the redundant axis of rotation for compensation in simulations in order to see the exact orientation errors of our proposed method.

4.2.1 Linear path without rotation

Simulation scenes, generated trajectories, position and orientation error graphs are given respectively in **Figure 4.4**, **Figure 4.5**, **Figure 4.6,** and **Figure 4.7**.

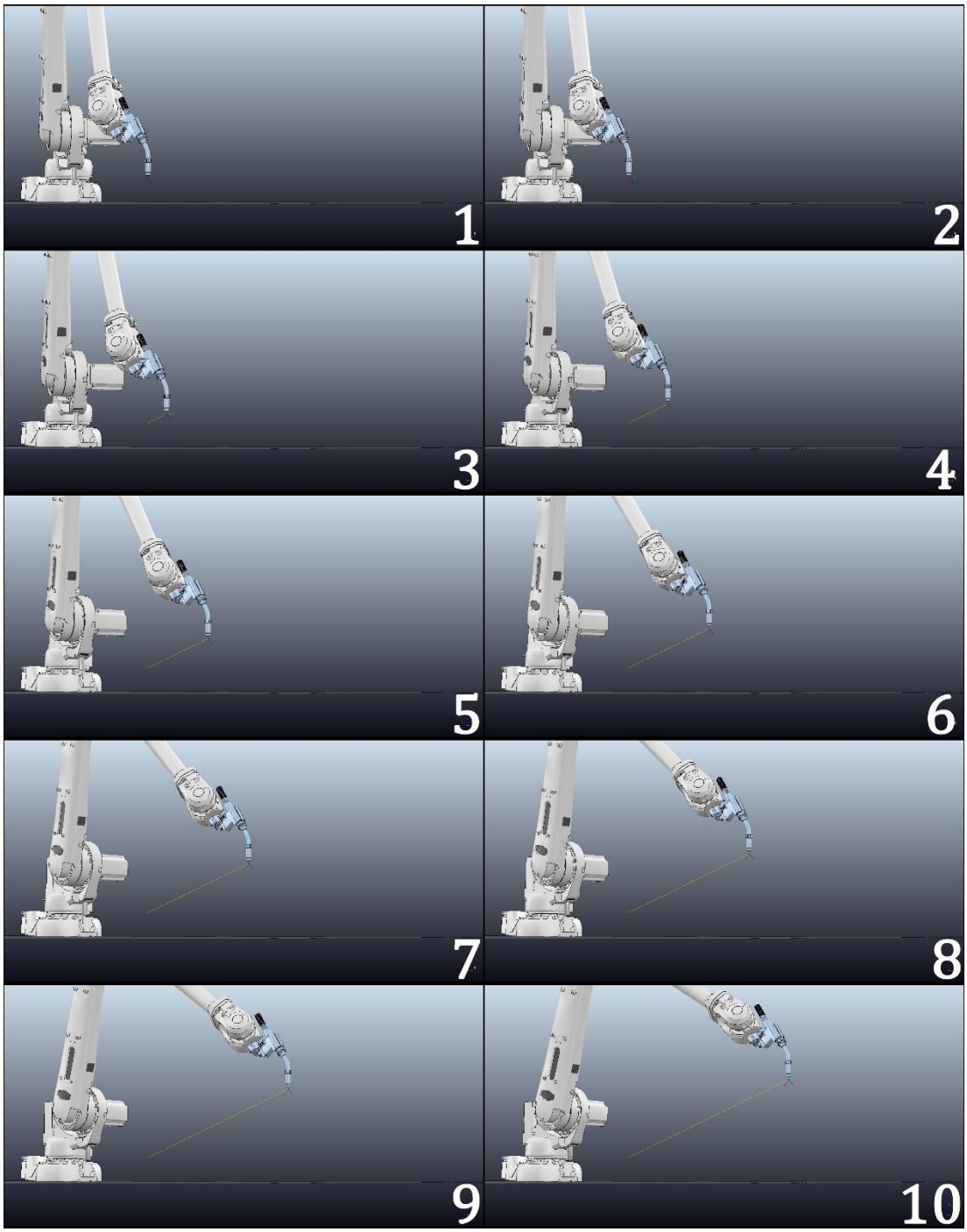

Figure 4.4 : Simulation scenes: linear path without rotation.

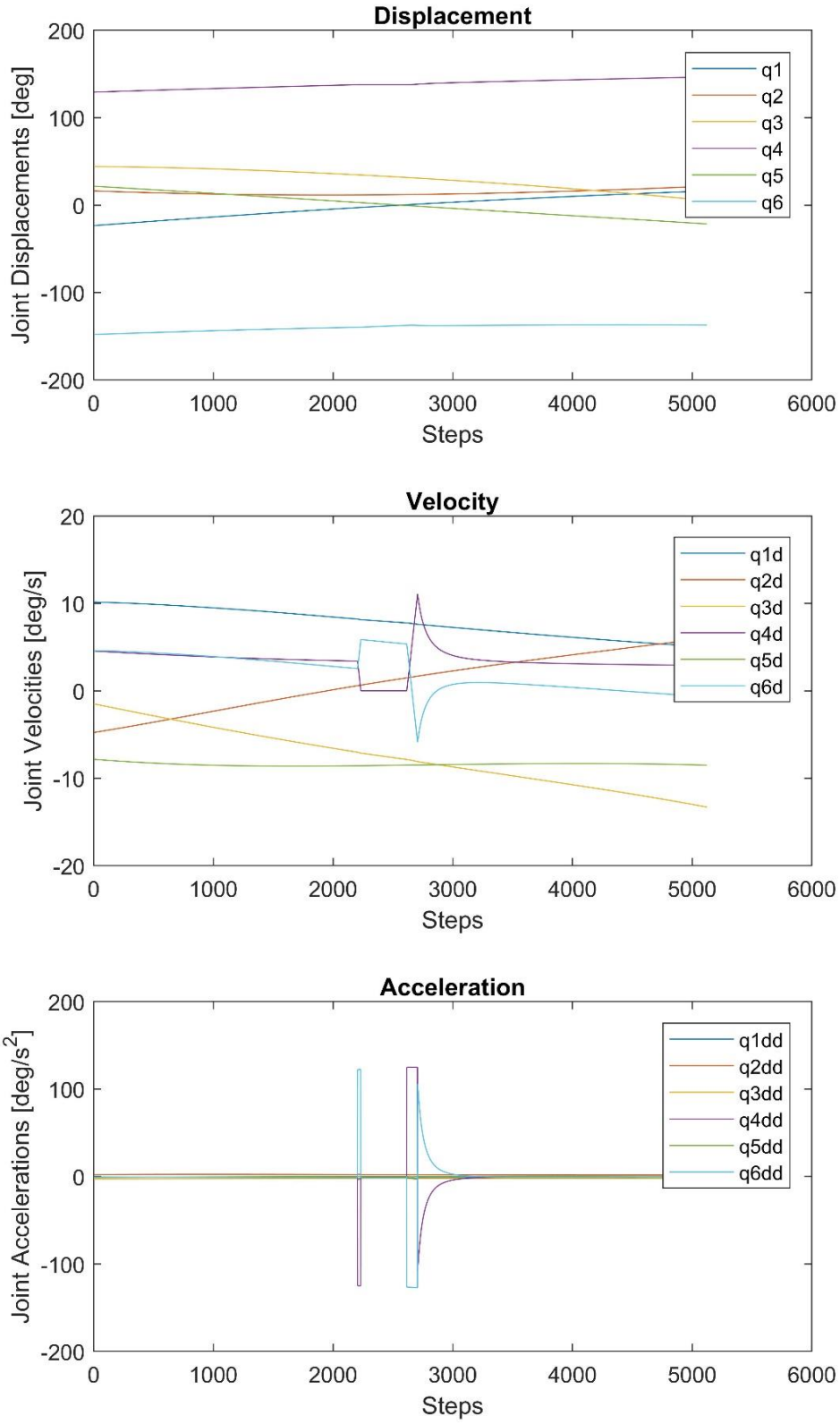

Figure 4.5 : Joint trajectories: linear path without rotation.

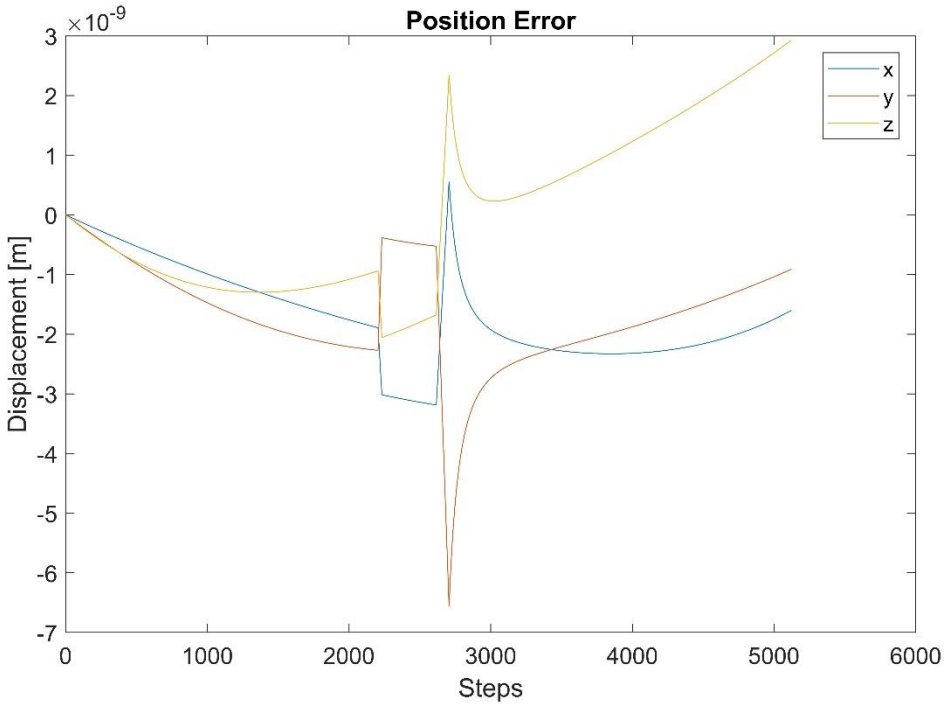

Figure 4.6 : Position error: linear path without rotation.

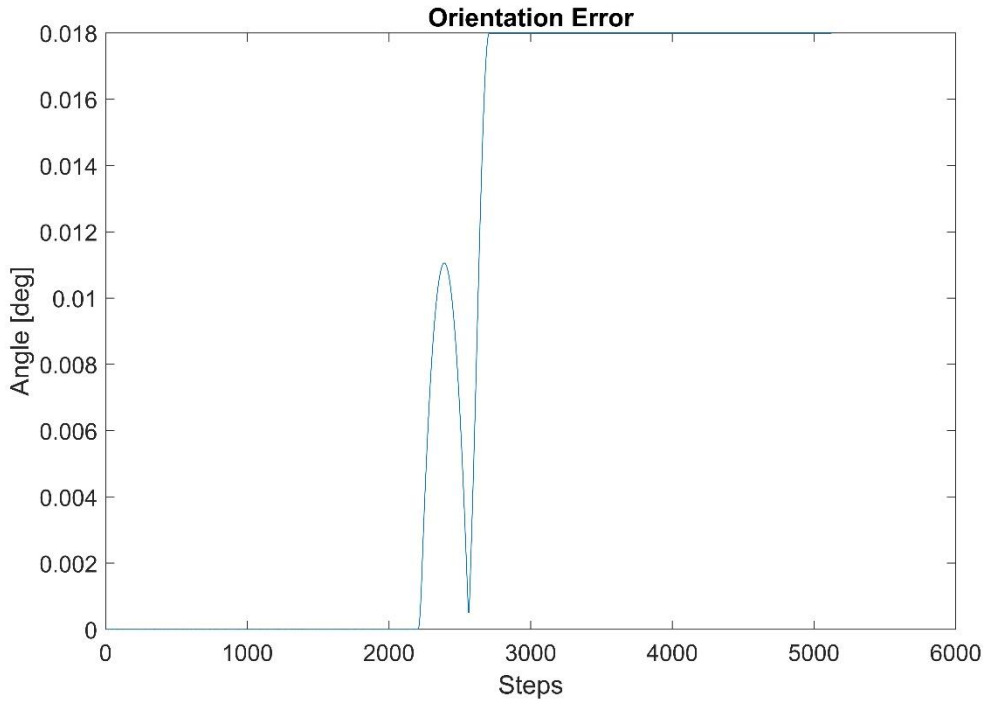

Figure 4.7 : Orientation error: linear path without rotation.

4.2.2 Linear path with rotation about x-axis

Simulation scenes, generated trajectories, position and orientation error graphs are given respectively in **Figure 4.8**, **Figure 4.9**, **Figure 4.10,** and **Figure 4.11**.

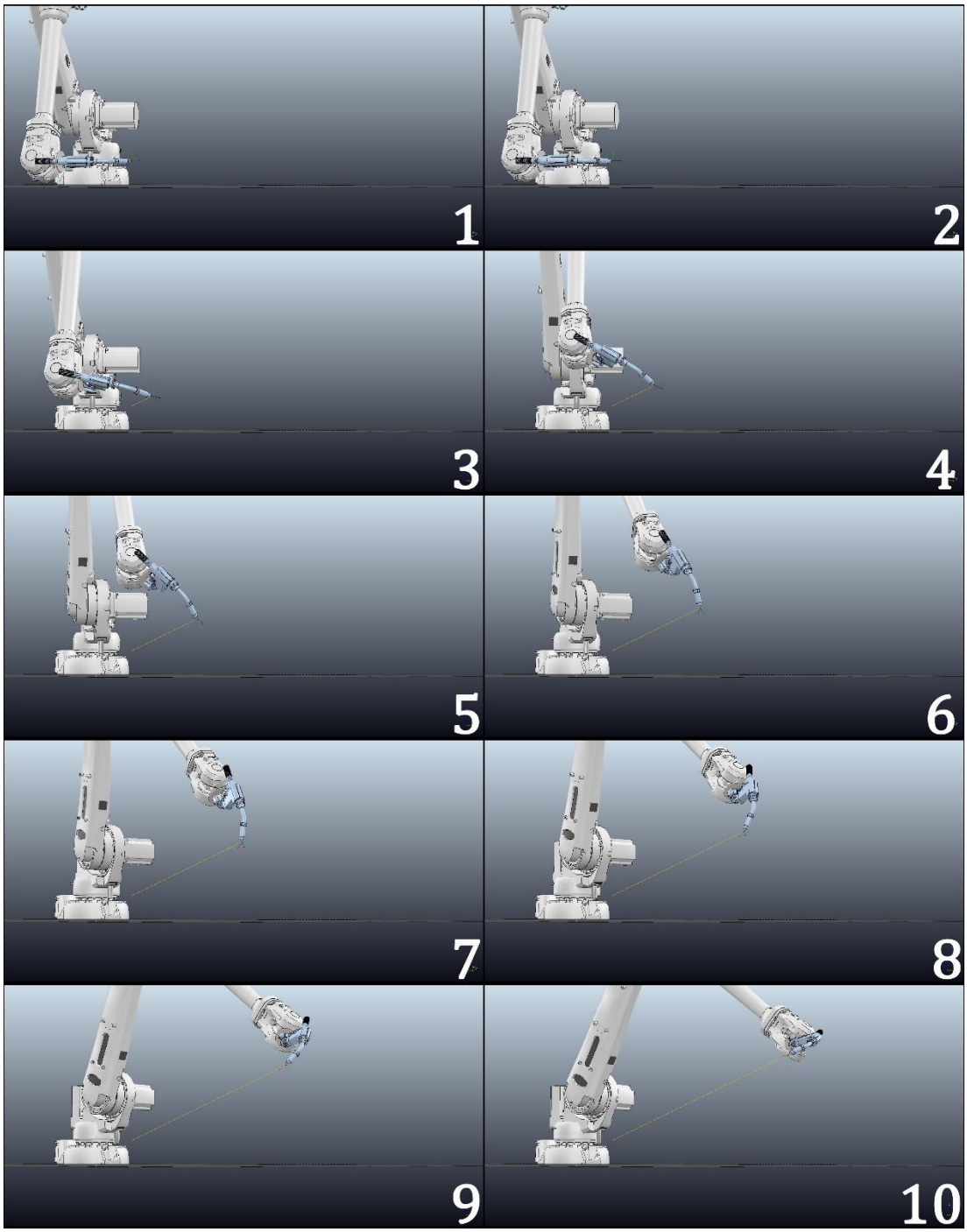

Figure 4.8 : Simulation scenes: linear path with rotation about x-axis.

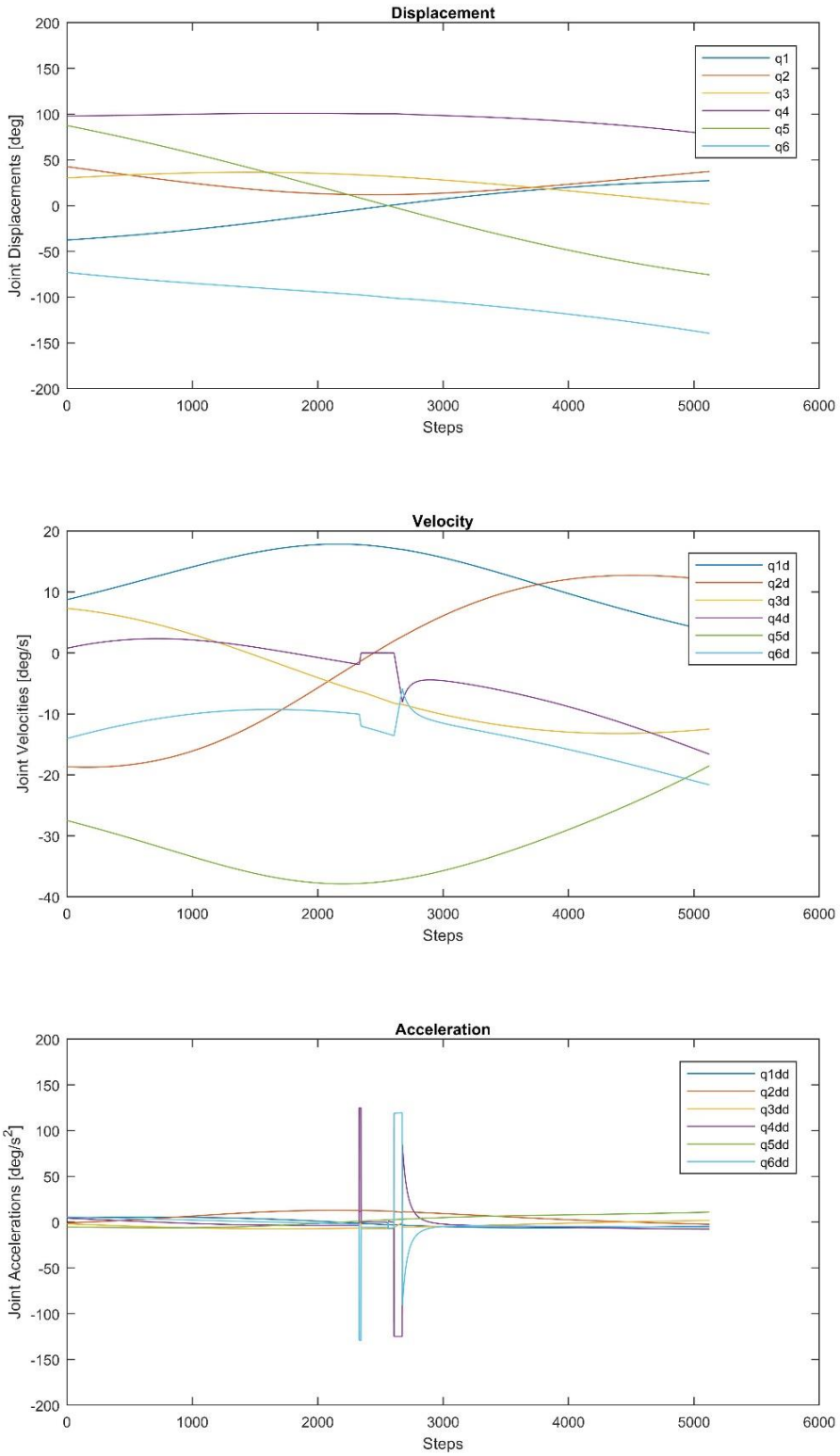

Figure 4.9 : Joint trajectories: linear path with rotation about x-axis.

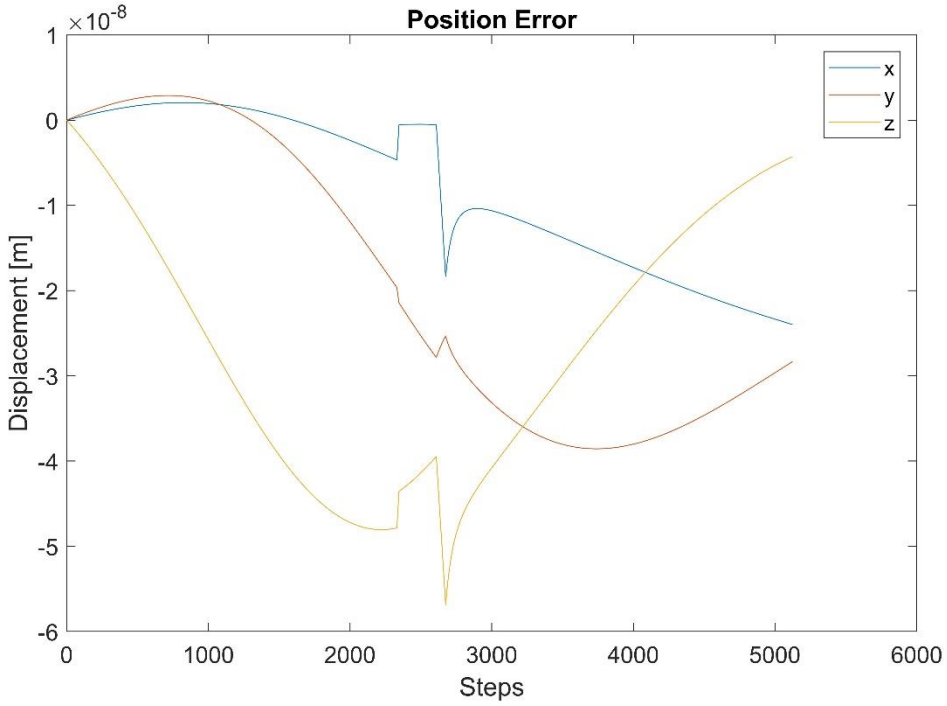

Figure 4.10 : Position error: linear path with rotation about x-axis.

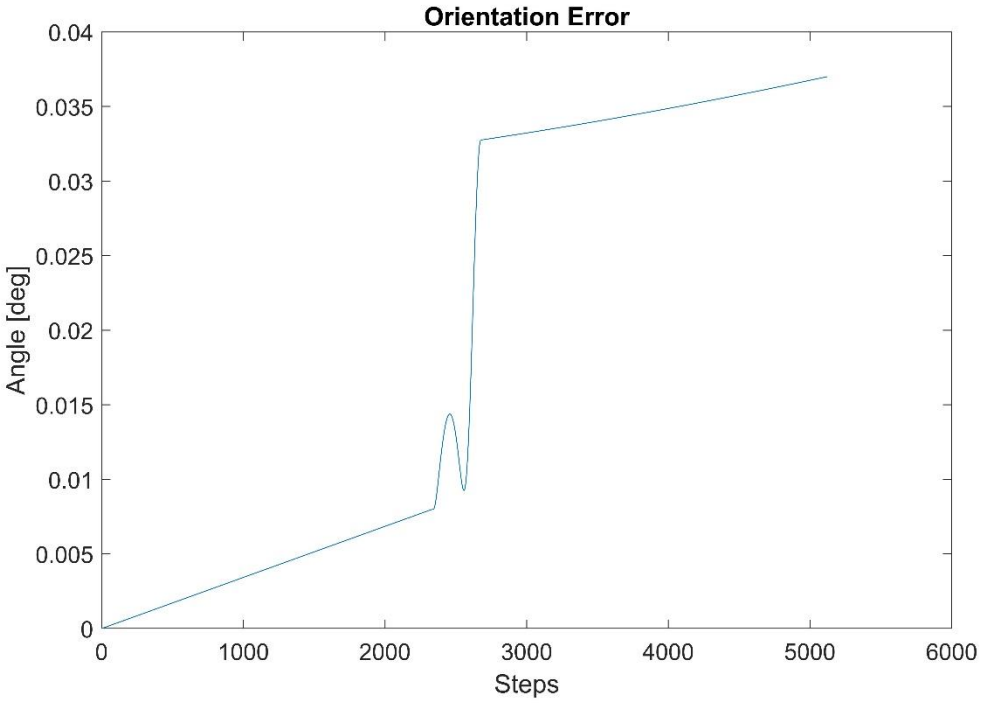

Figure 4.11 : Orientation error: linear path with rotation about x-axis.

4.2.3 Linear path with rotation about y-axis

Simulation scenes, generated trajectories, position and orientation error graphs are given respectively in **Figure 4.12**, **Figure 4.13**, **Figure 4.14,** and **Figure 4.15**.

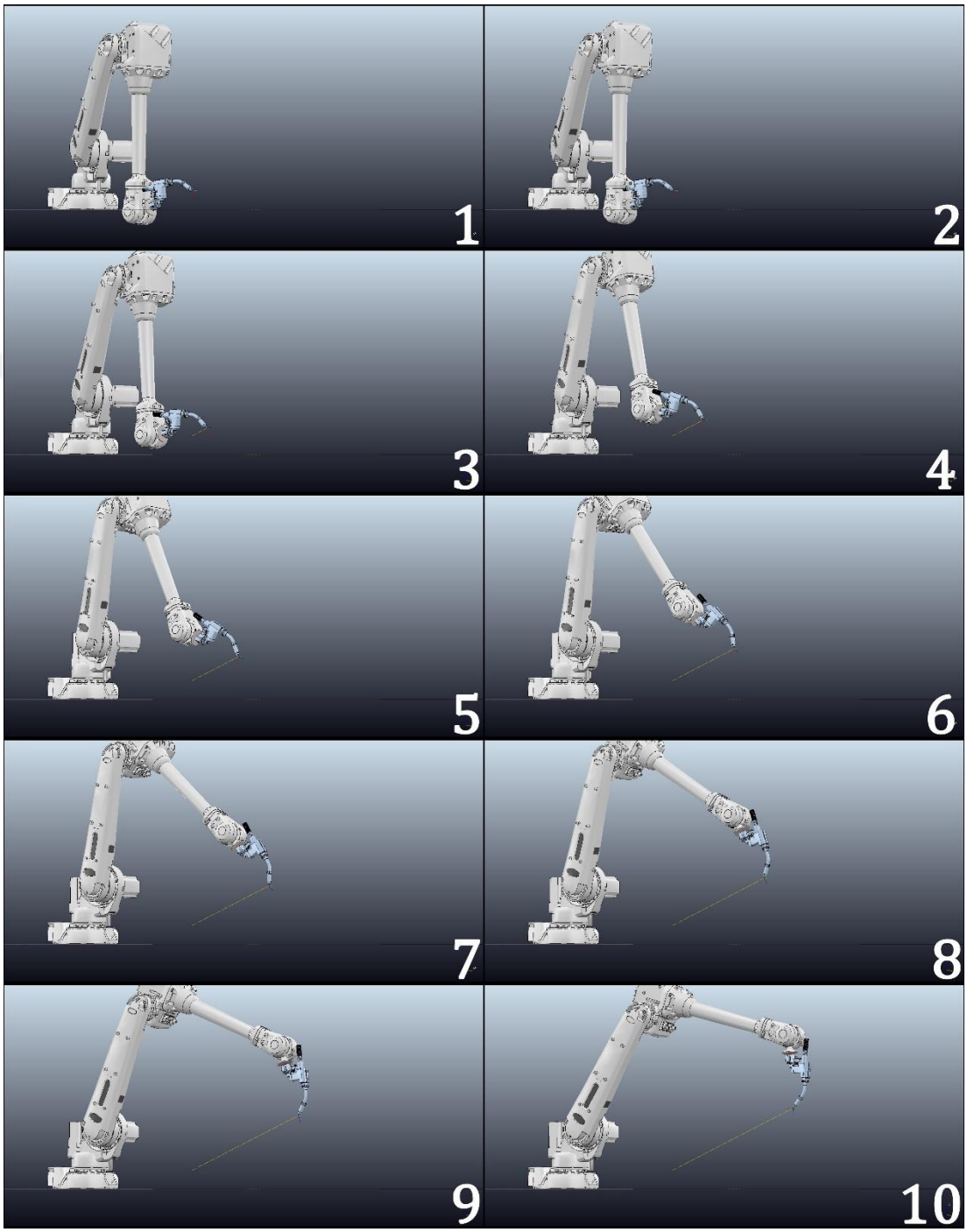

Figure 4.12 : Simulation scenes: linear path with rotation about y-axis.

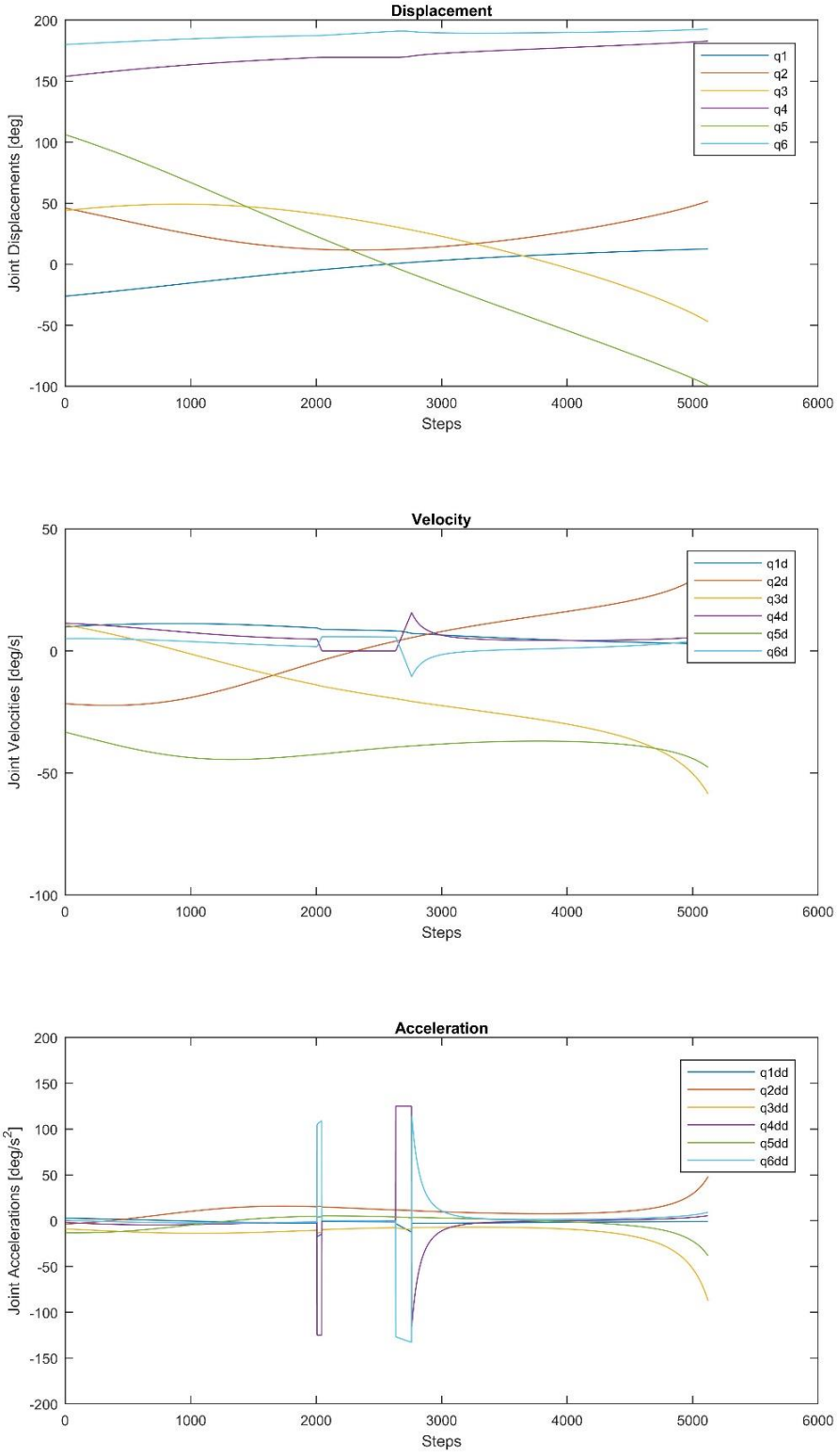

Figure 4.13 : Joint trajectories: linear path with rotation about y-axis.

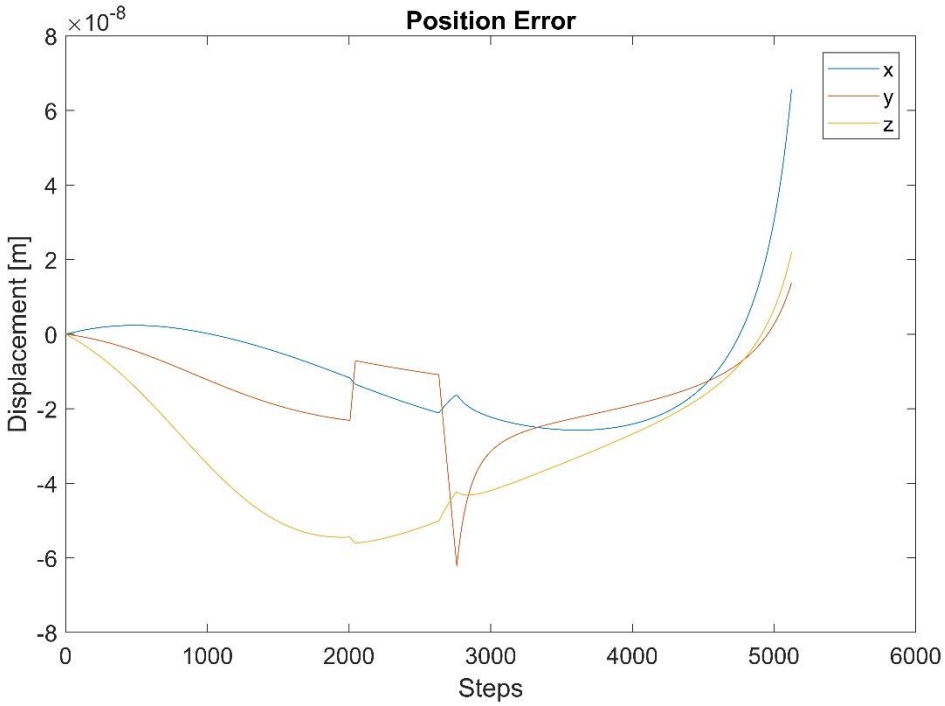

Figure 4.14 : Position error: linear path with rotation about y-axis.

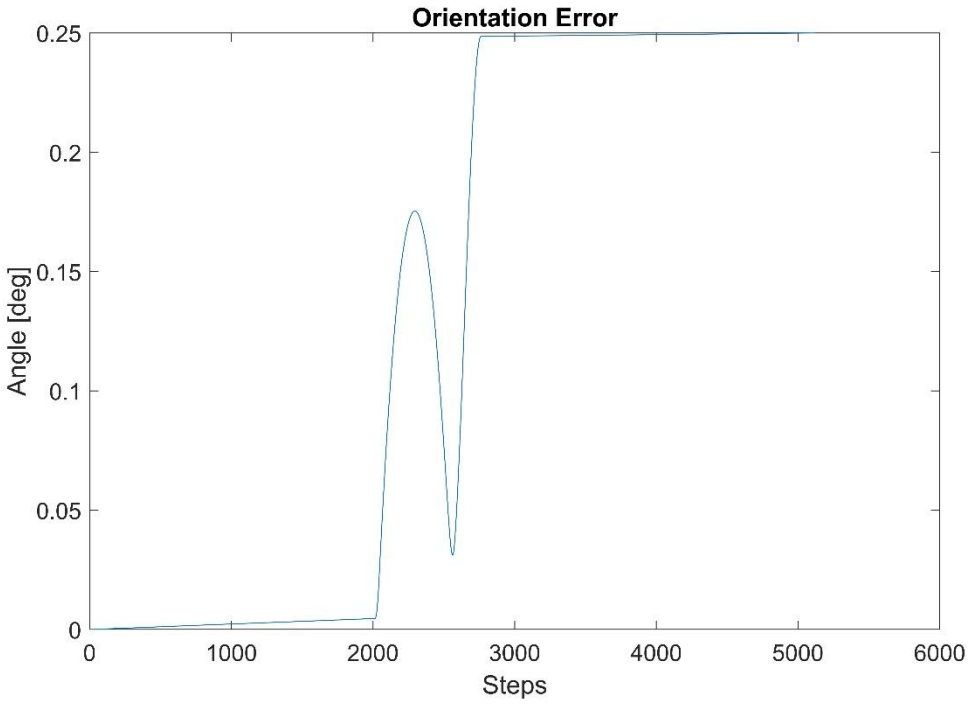

Figure 4.15 : Orientation error: linear path with rotation about y-axis.

4.2.4 Linear path with rotation about z-axis

Simulation scenes, generated trajectories, position and orientation error graphs are given respectively in **Figure 4.16**, **Figure 4.17**, **Figure 4.18,** and **Figure 4.19**.

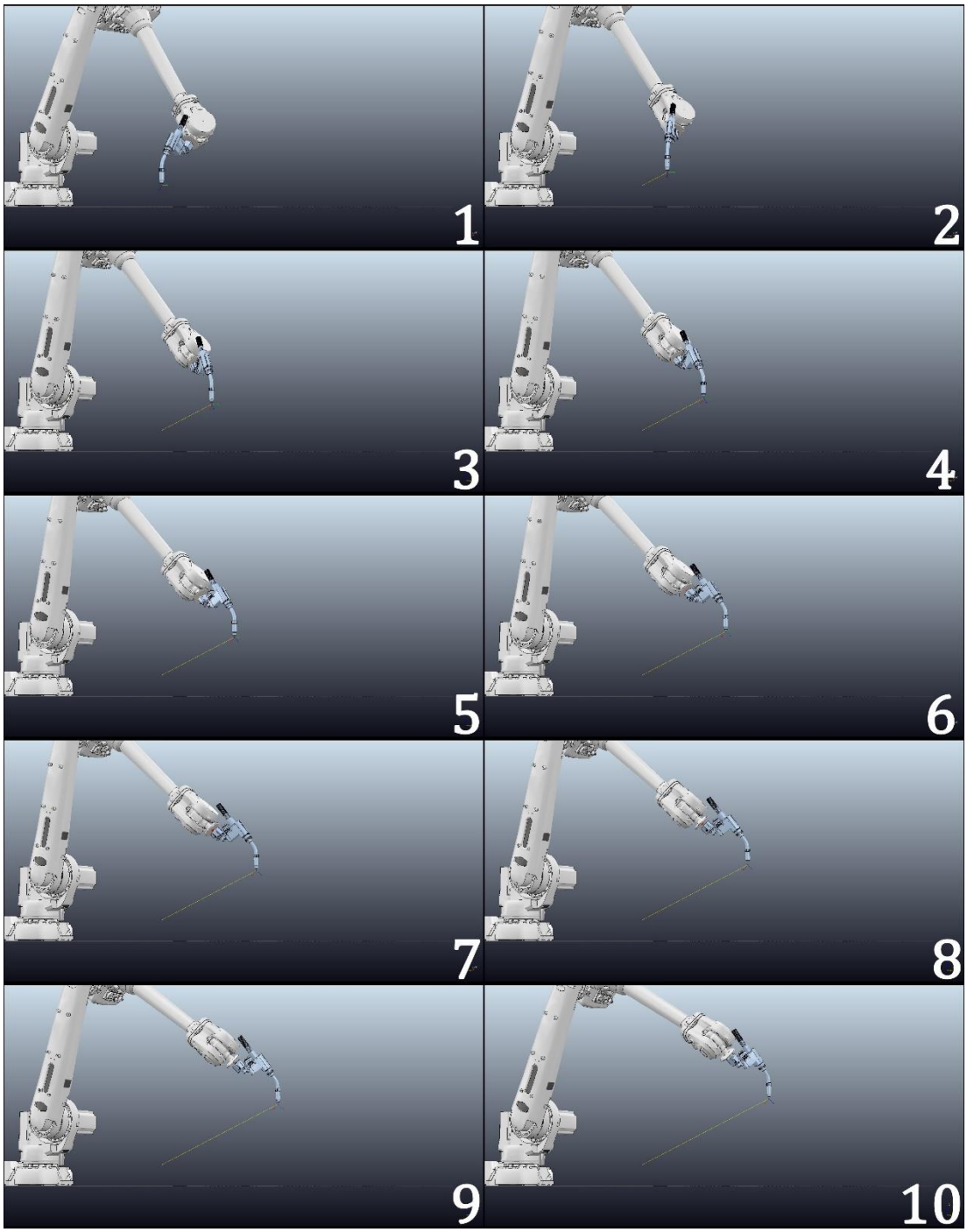

Figure 4.16 : Simulation scenes: linear path with rotation about z-axis.

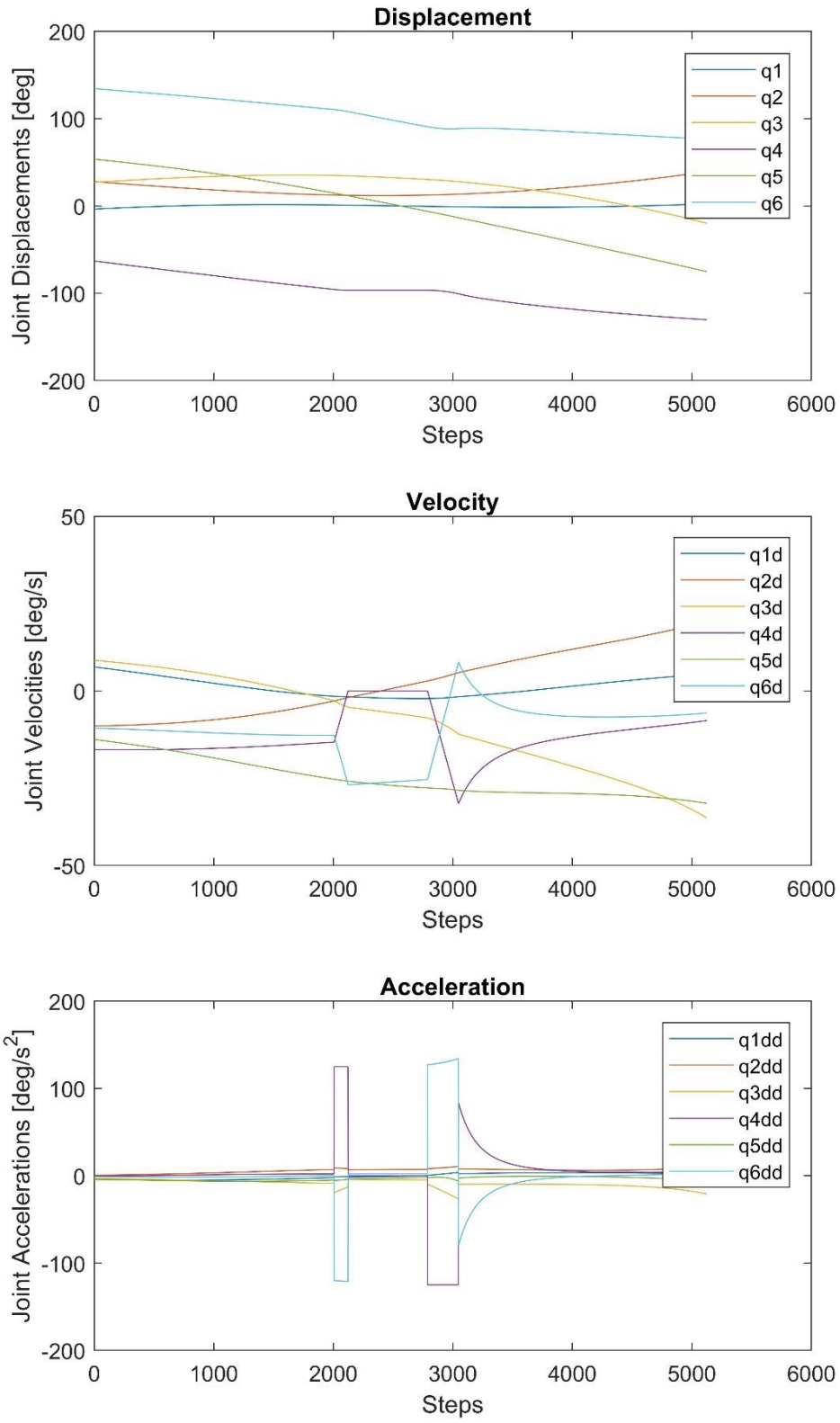

Figure 4.17 : Joint trajectories: linear path with rotation about z-axis.

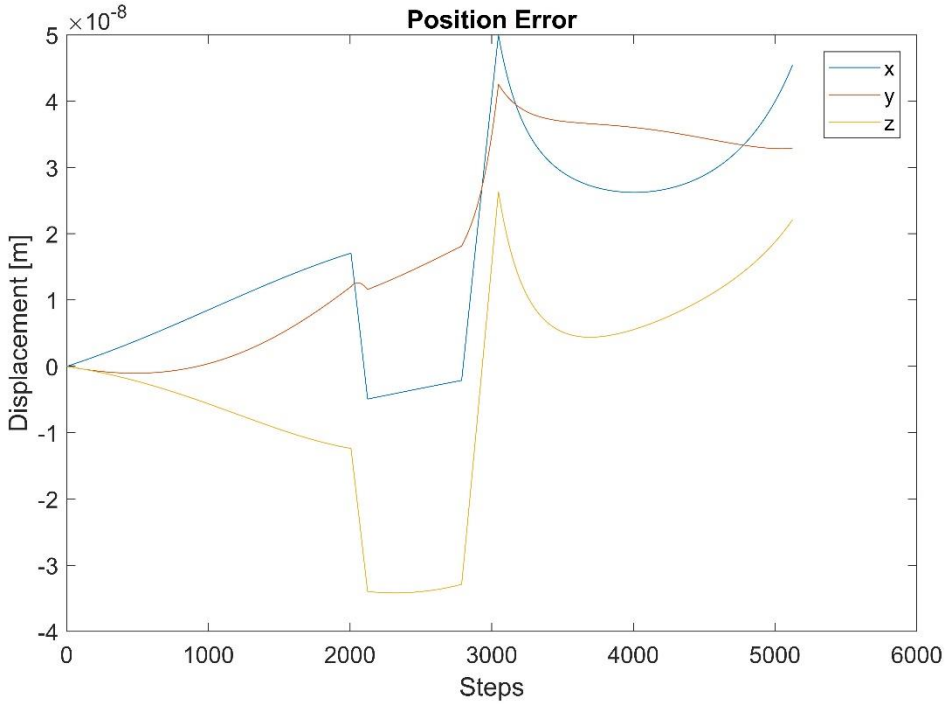

Figure 4.18 : Position error: linear path with rotation about z-axis.

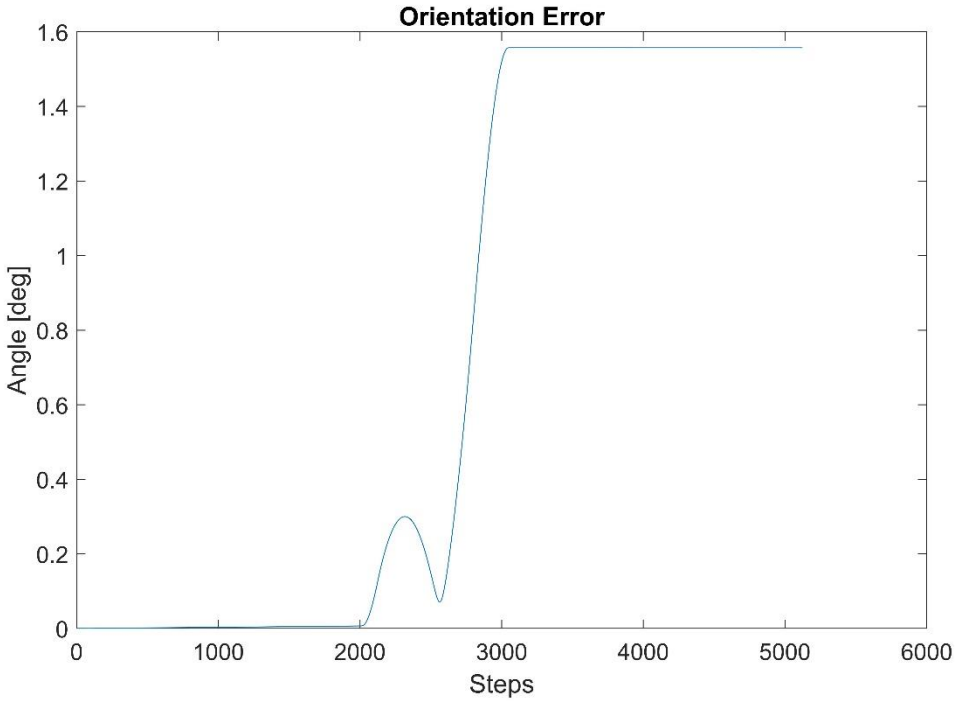

Figure 4.19 : Orientation error: linear path with rotation about z-axis.

4.2.5 Circular path with arc rotation

Simulation scenes, generated trajectories, position and orientation error graphs are given respectively in **Figure 4.20**, **Figure 4.21**, **Figure 4.22,** and **Figure 4.23**. For this path, user-defined parameters are selected as below.

$$\eta_{accUsage} = \%5, \qquad \eta_{acc} = \%4, \qquad \eta_{vel} = \%50$$

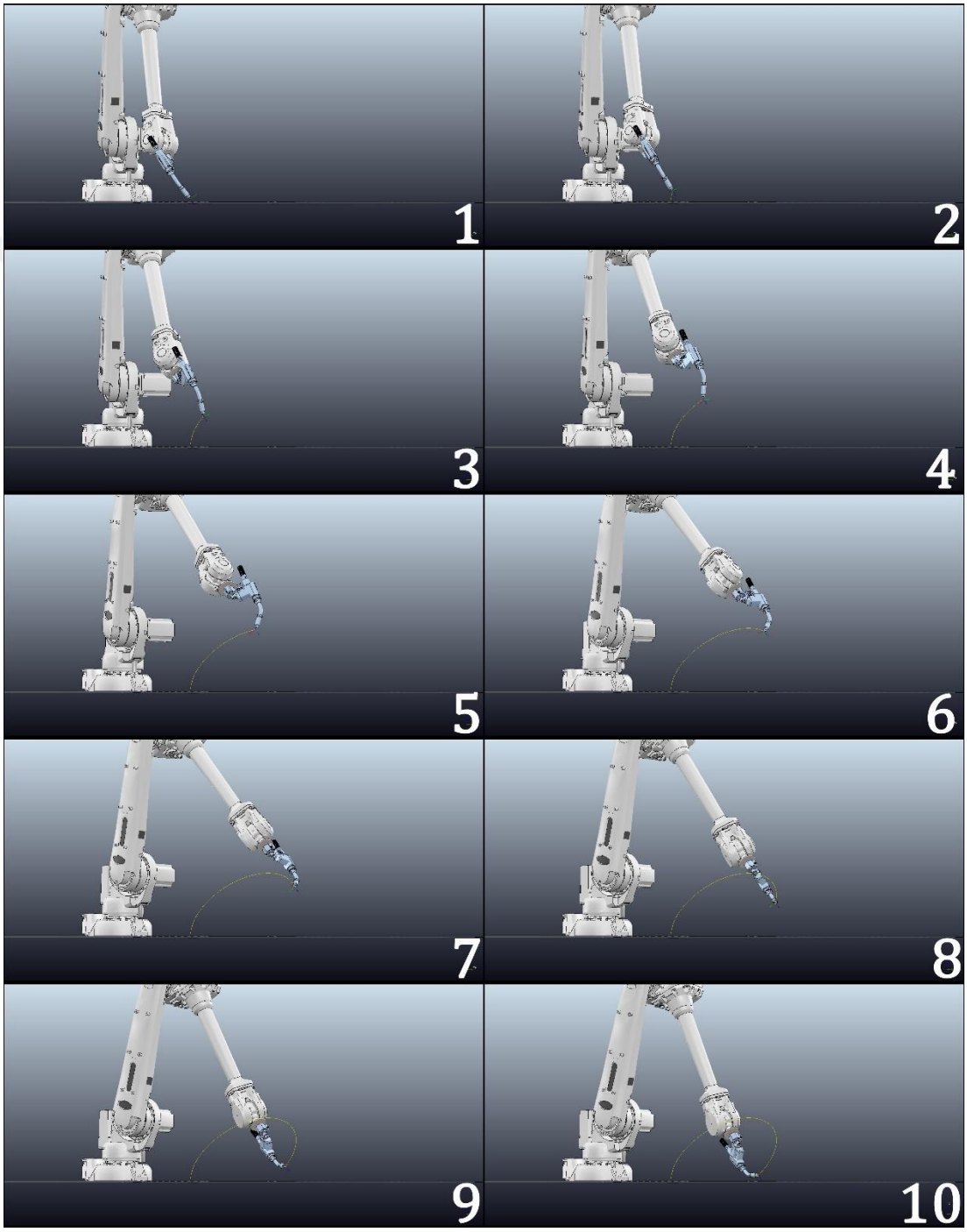

Figure 4.20 : Simulation scenes: circular path with arc rotation.

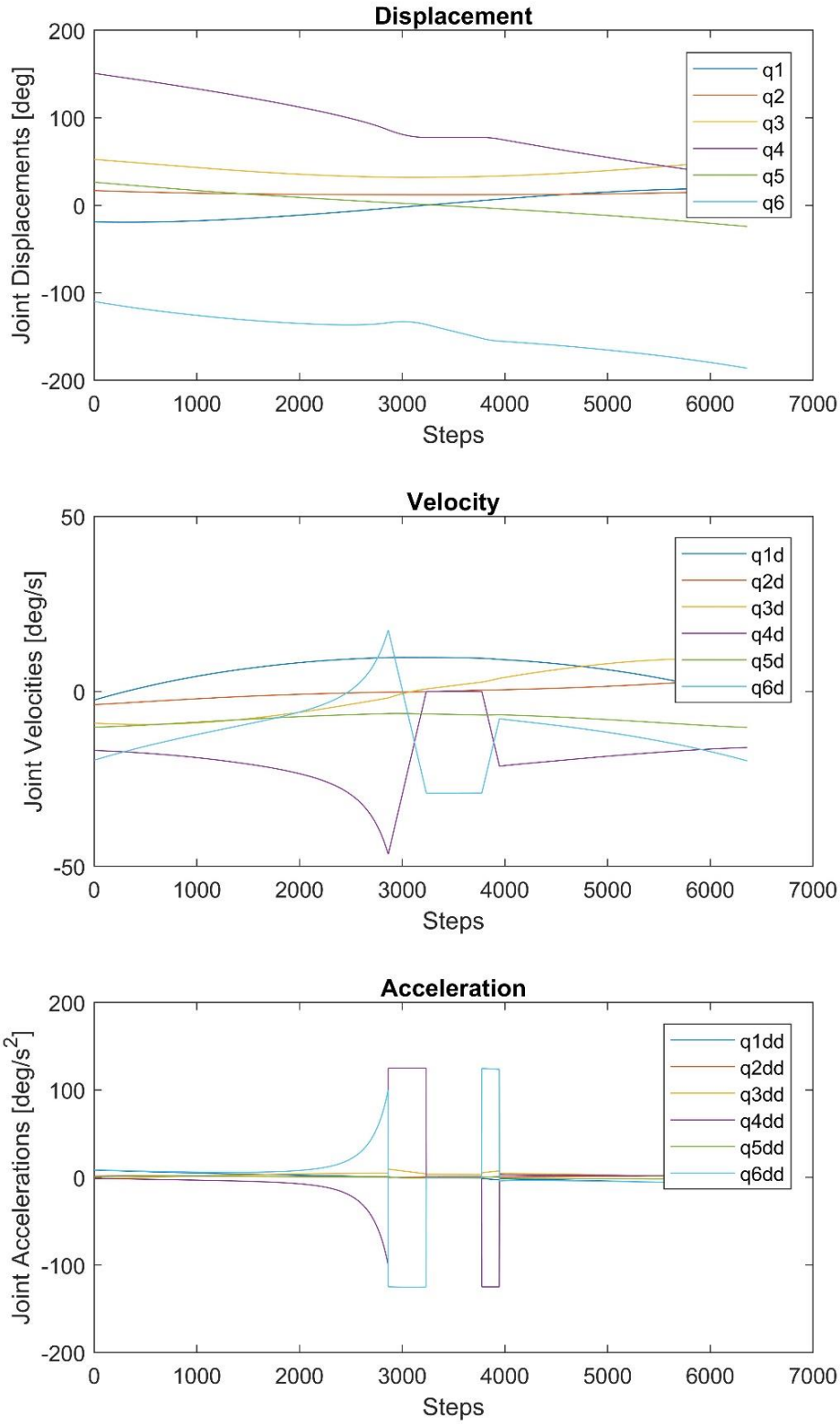

Figure 4.21 : Joint trajectories: circular path with arc rotation.

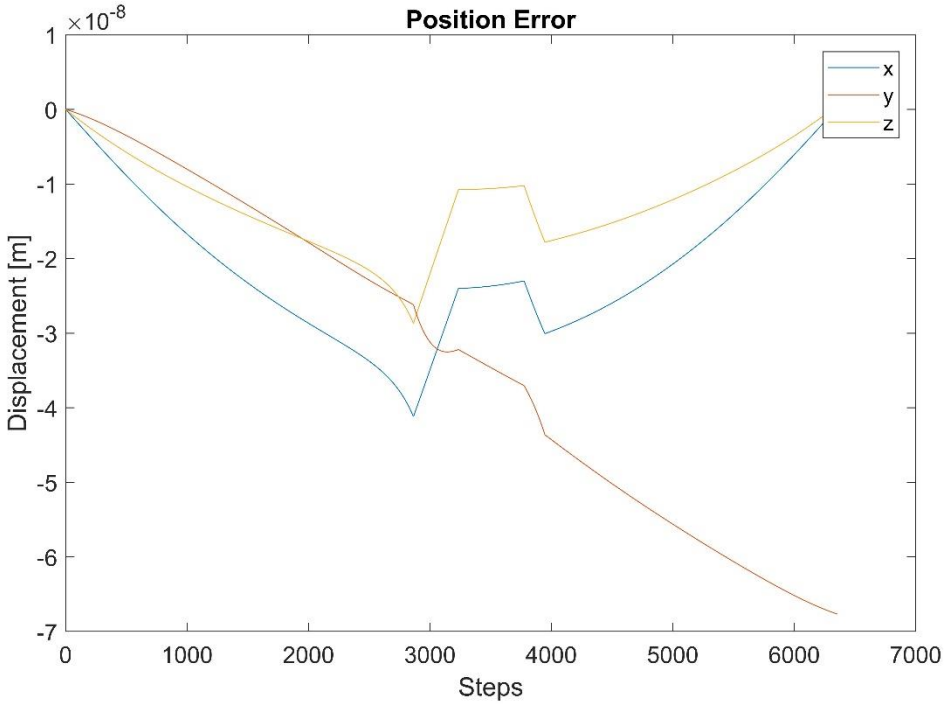

Figure 4.22 : Position error: circular path with arc rotation.

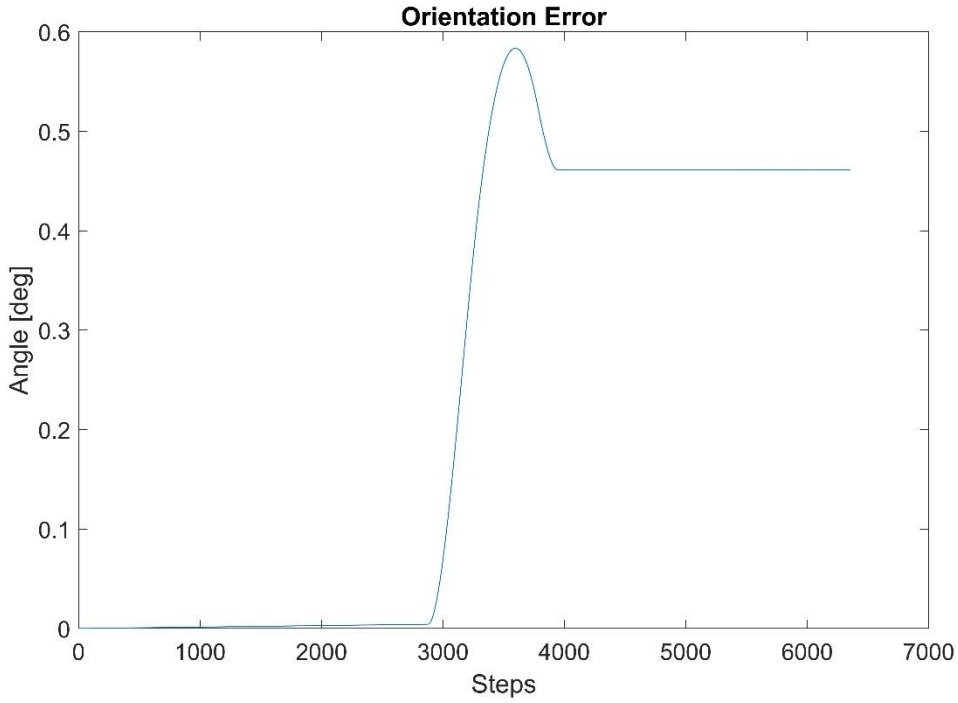

Figure 4.23 : Orientation error: circular path with arc rotation.

4.2.6 Circular path with rotation about x-axis

Simulation scenes, generated trajectories, position and orientation error graphs are given respectively in **Figure 4.24**, **Figure 4.25**, **Figure 4.26** and **Figure 4.27**.

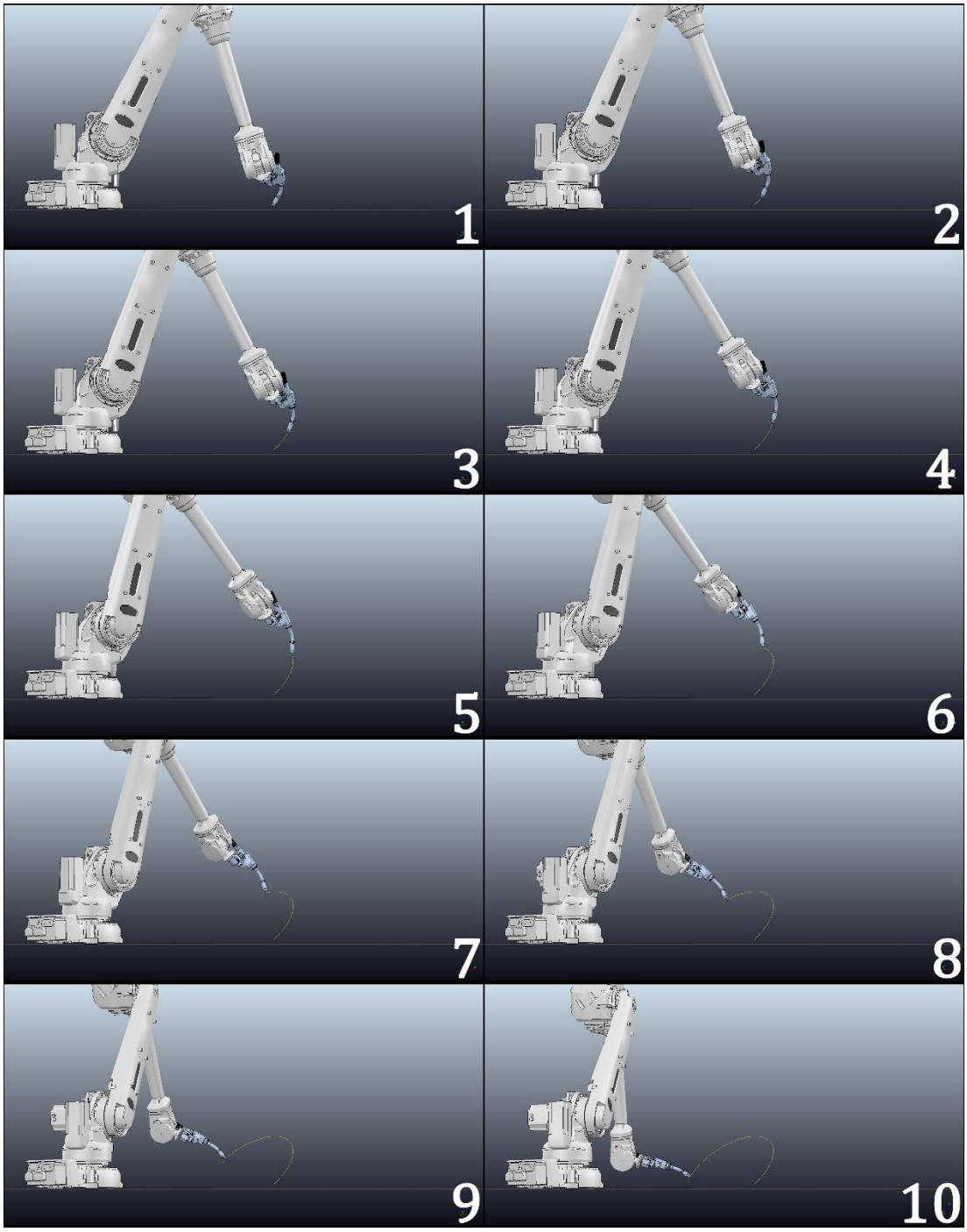

Figure 4.24 : Simulation scenes: circular path with rotation about x-axis.

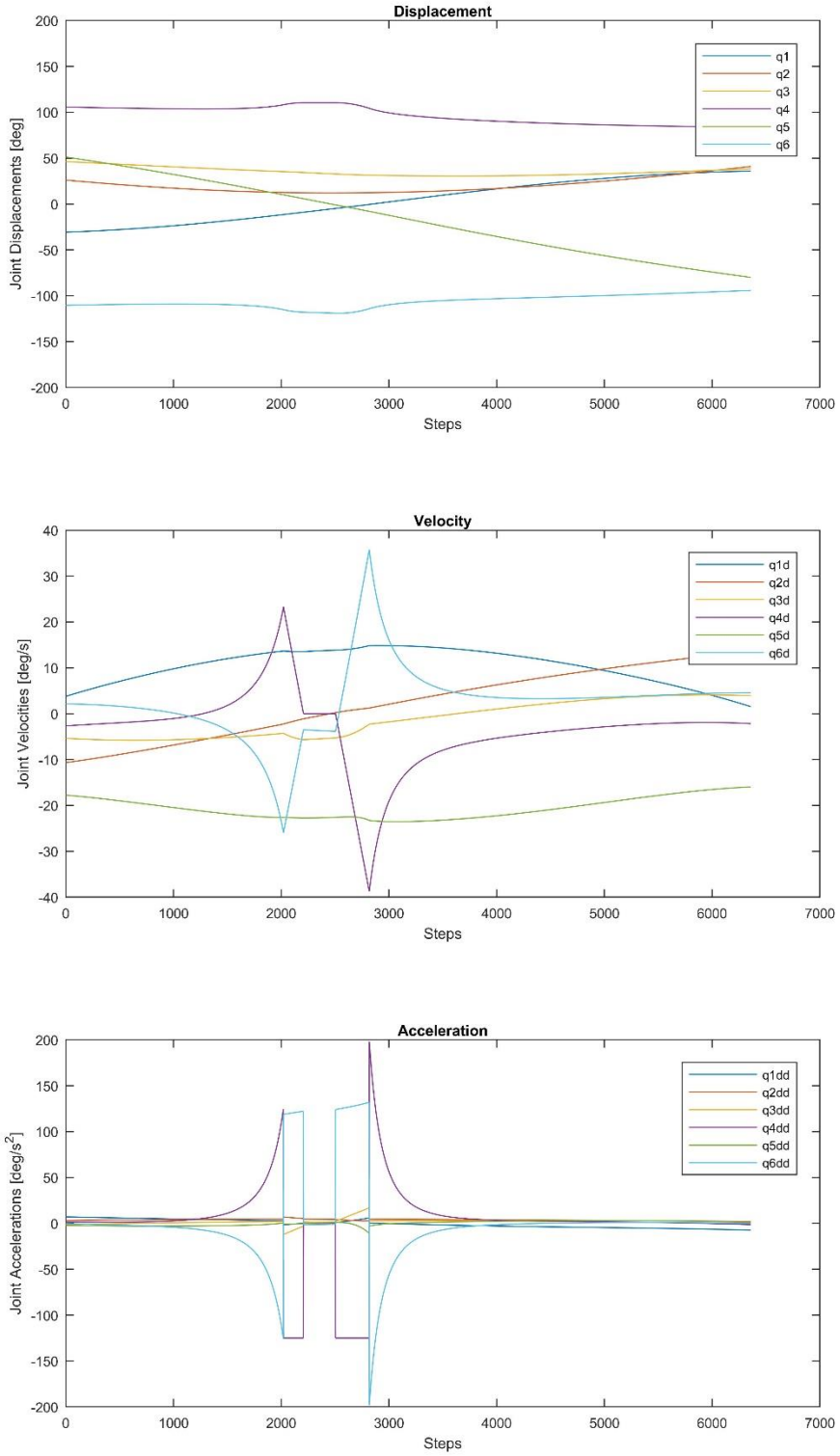

Figure 4.25 : Joint trajectories: circular path with rotation about x-axis.

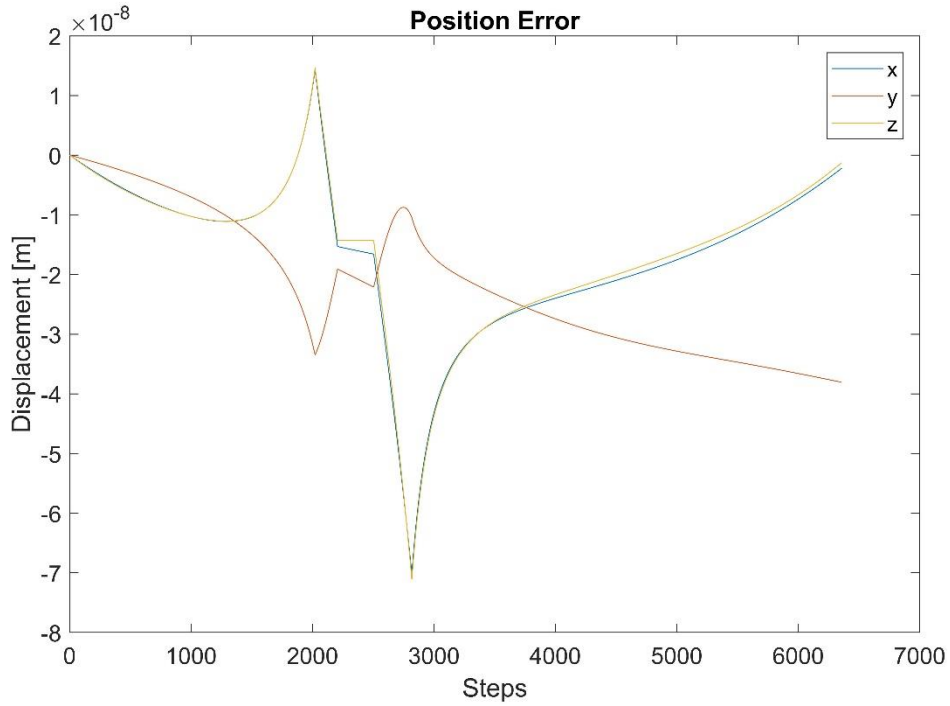

Figure 4.26 : Position error: circular path with rotation about x-axis.

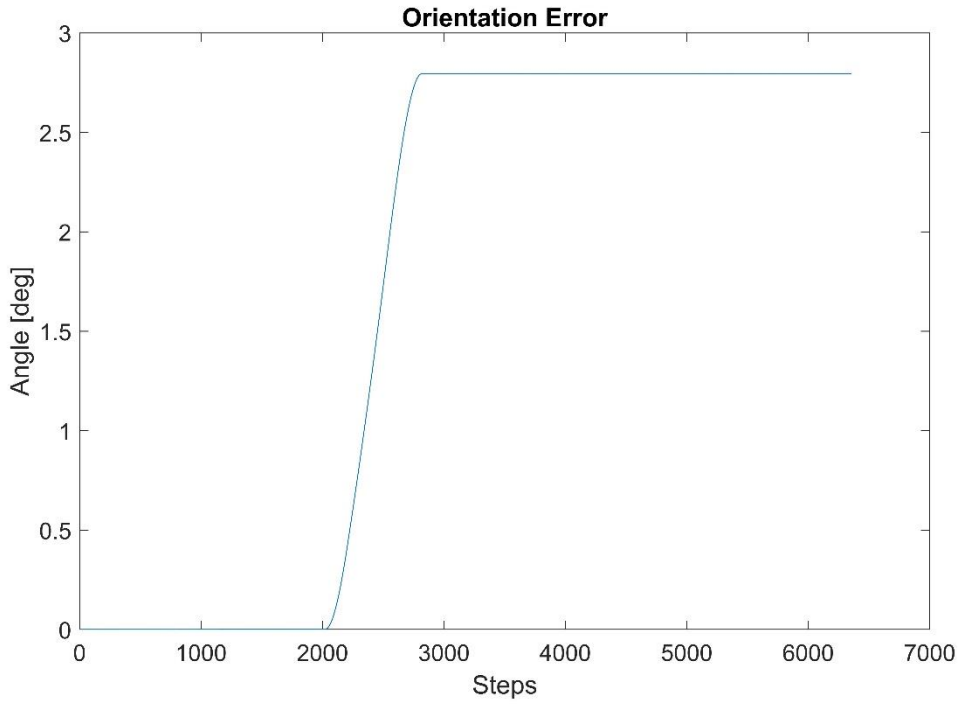

Figure 4.27 : Orientation error: circular path with rotation about x-axis.

4.2.7 Circular path with rotation about y-axis

Simulation scenes, generated trajectories, position and orientation error graphs are given respectively in **Figure 4.28**, **Figure 4.29**, **Figure 4.30,** and **Figure 4.31**.

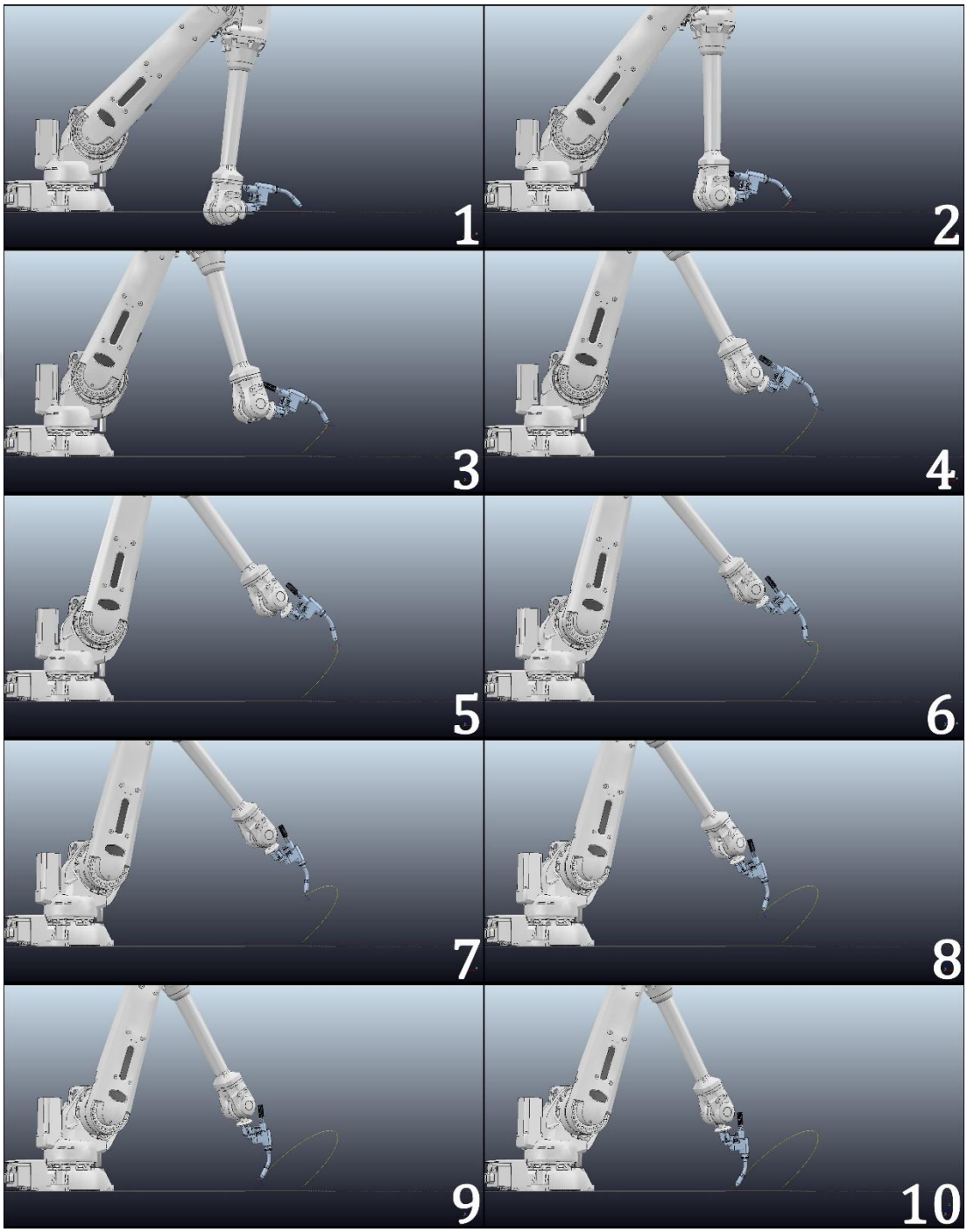

Figure 4.28 : Simulation scenes: circular path with rotation about y-axis.

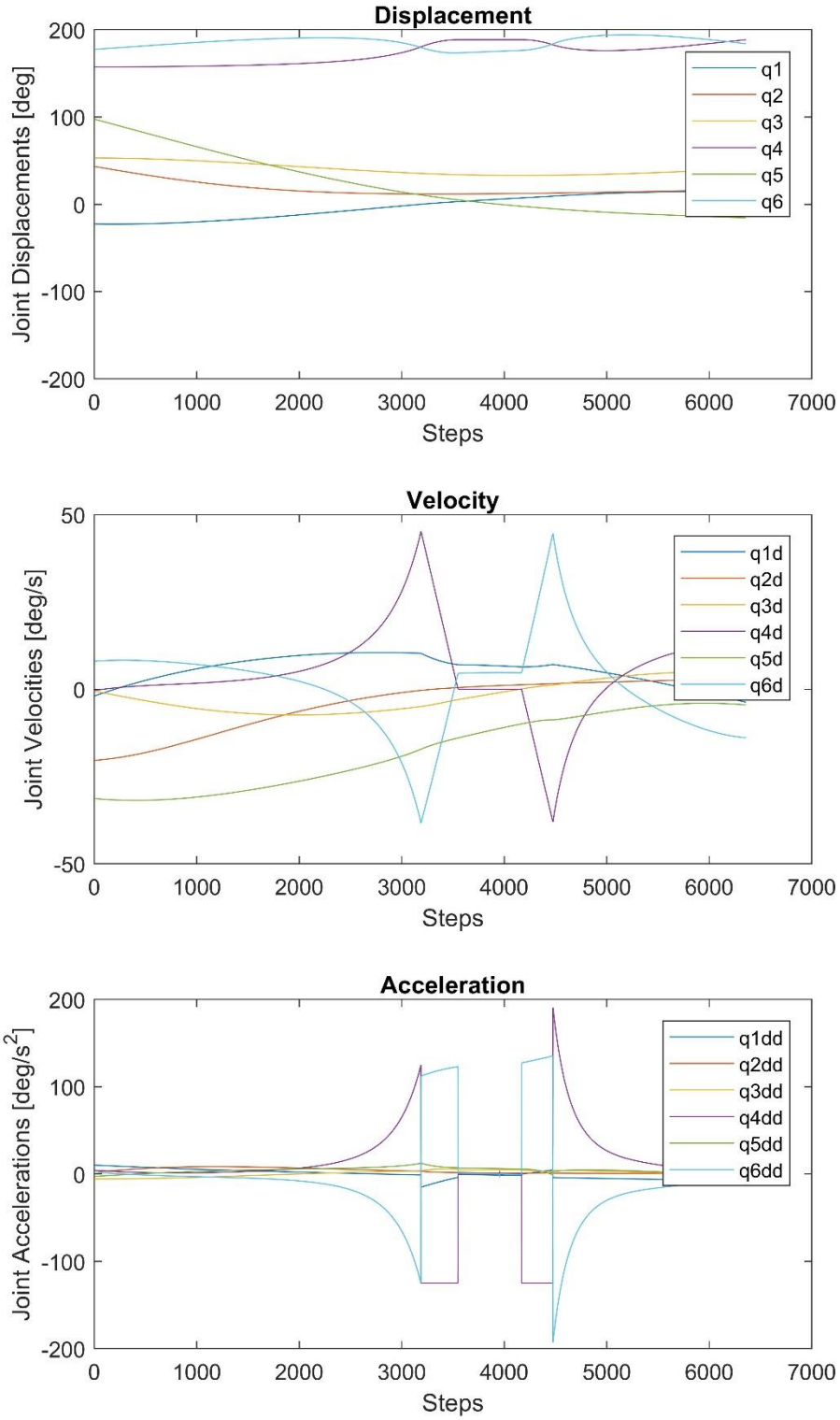

Figure 4.29 : Joint trajectories: circular path with rotation about y-axis.

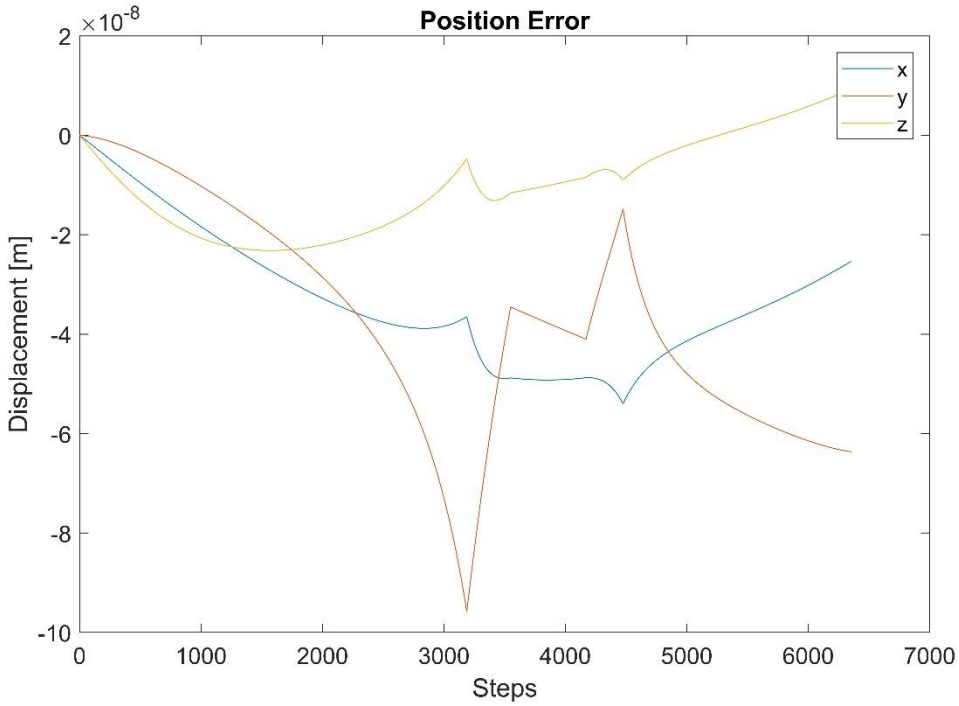

Figure 4.30 : Position error: circular path with rotation about y-axis.

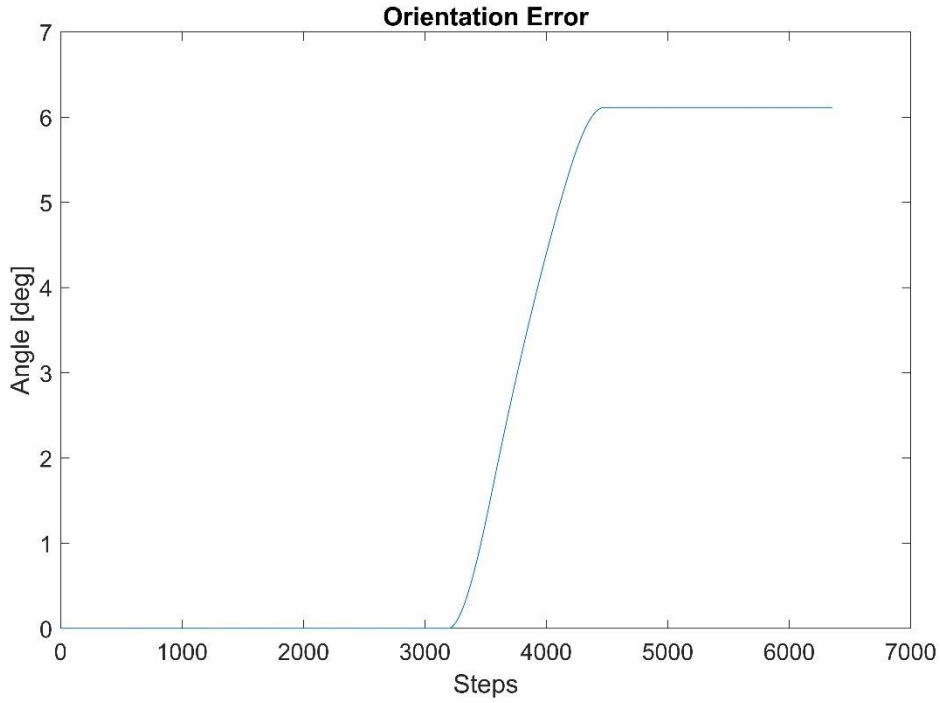

Figure 4.31 : Orientation error: circular path with rotation about y-axis.

Because of the relatively high orientation error, it is possible to represent the orientation difference between initial and compensated paths in **Figure 4.32**.

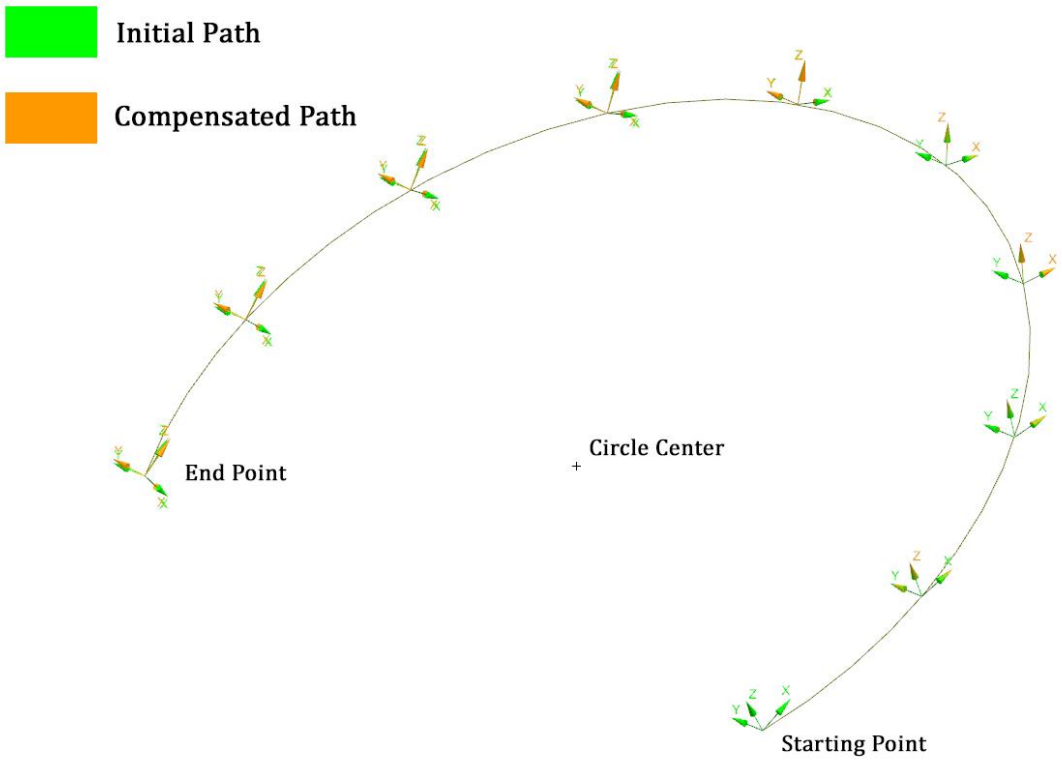

Figure 4.32 : Initial and compensated path representations: circular path with rotation about y-axis.

4.2.8 Circular path with rotation about z-axis

Simulation scenes, generated trajectories, position and orientation error graphs are given respectively in **Figure 4.33**, **Figure 4.34**, **Figure 4.35,** and **Figure 4.36**. For this path, user-defined parameters are selected as below.

$$\eta_{accUsage} = \%5, \qquad \eta_{acc} = \%4, \qquad \eta_{vel} = \%50$$

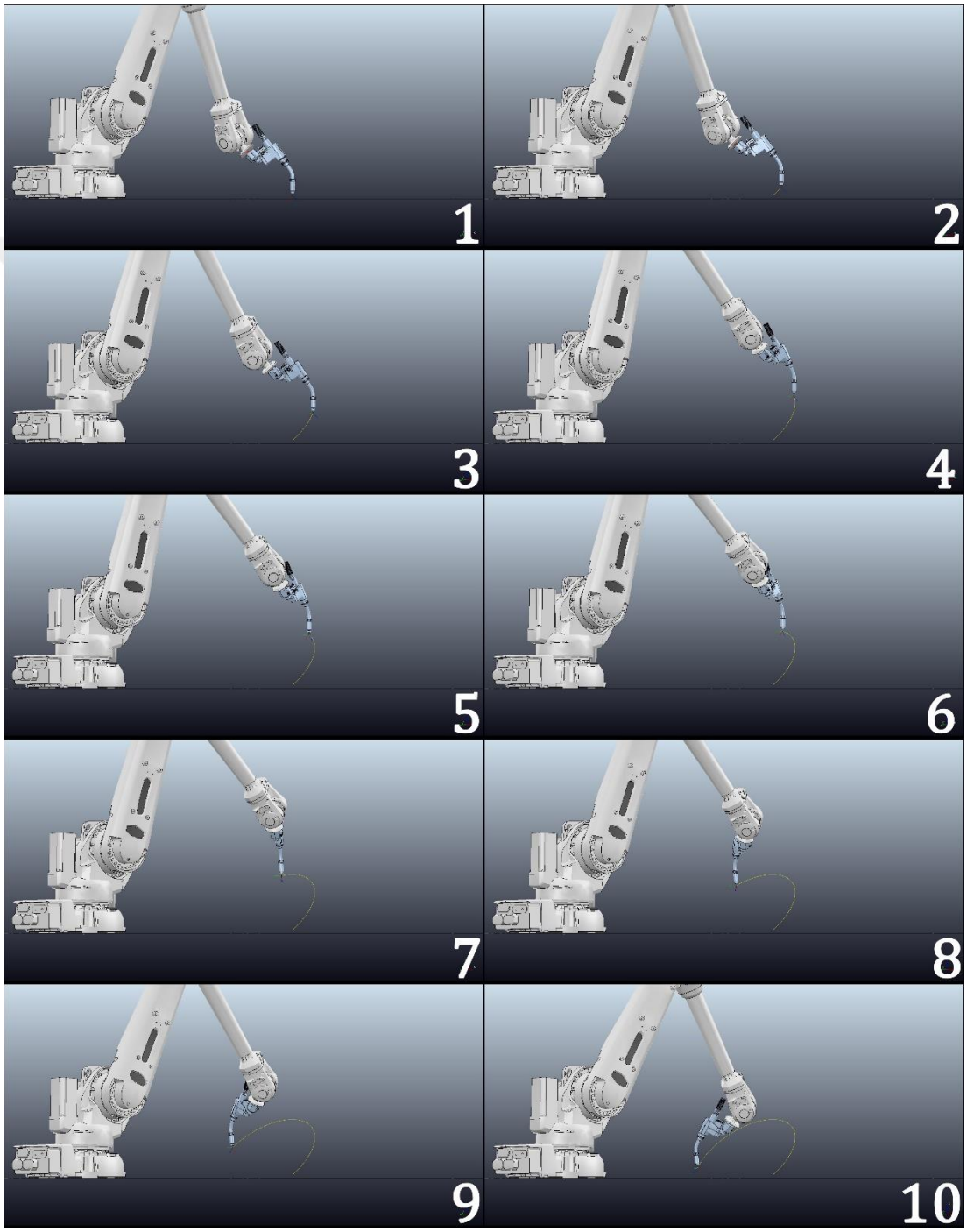

Figure 4.33 : Simulation scenes: circular path with rotation about z-axis.

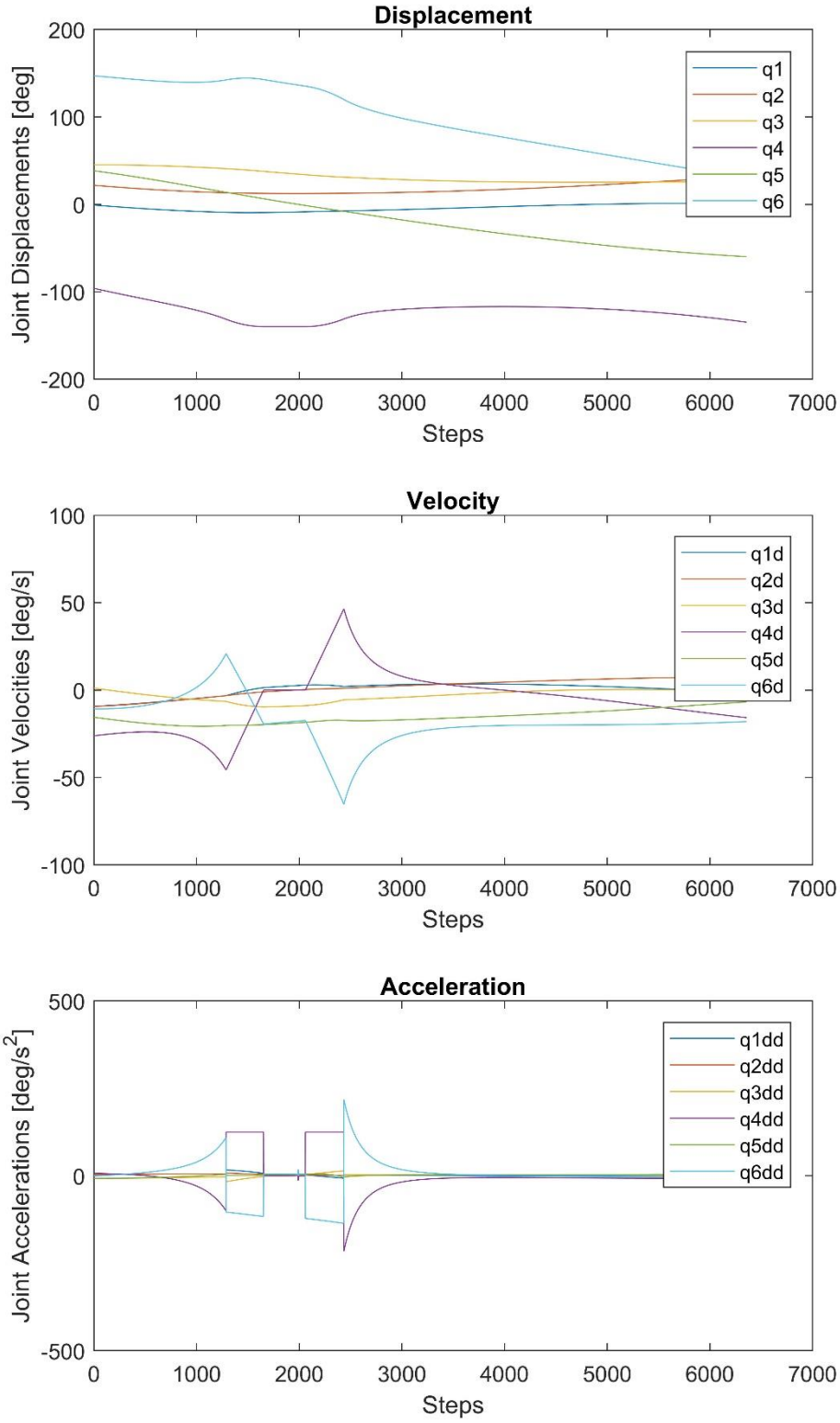

Figure 4.34 : Joint trajectories - circular path with rotation about z-axis.

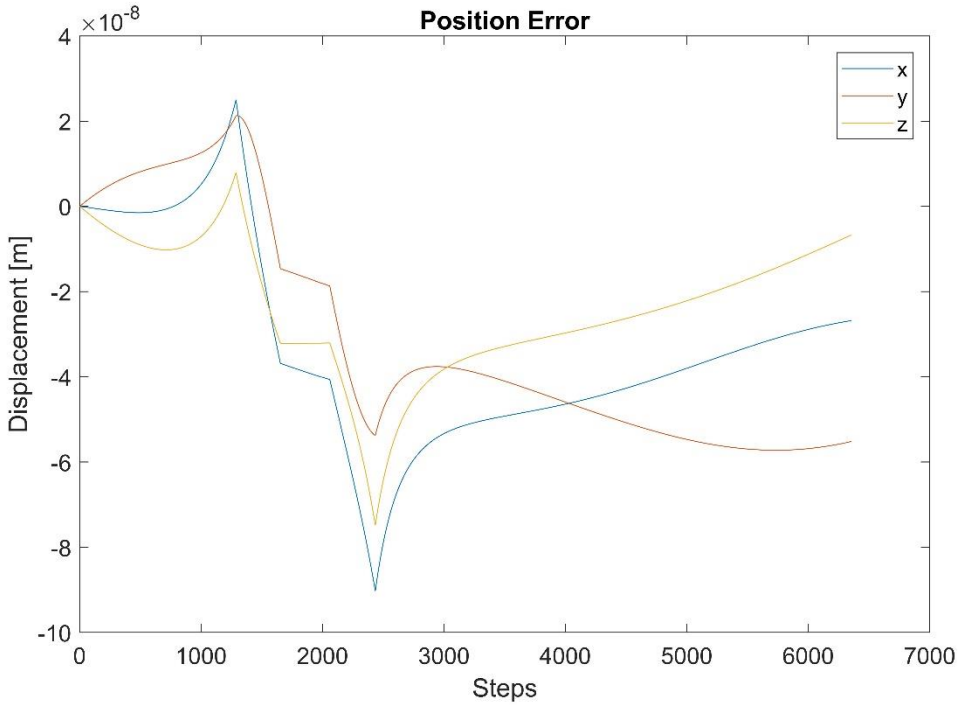

Figure 4.35 : Position error: circular path with rotation about z-axis.

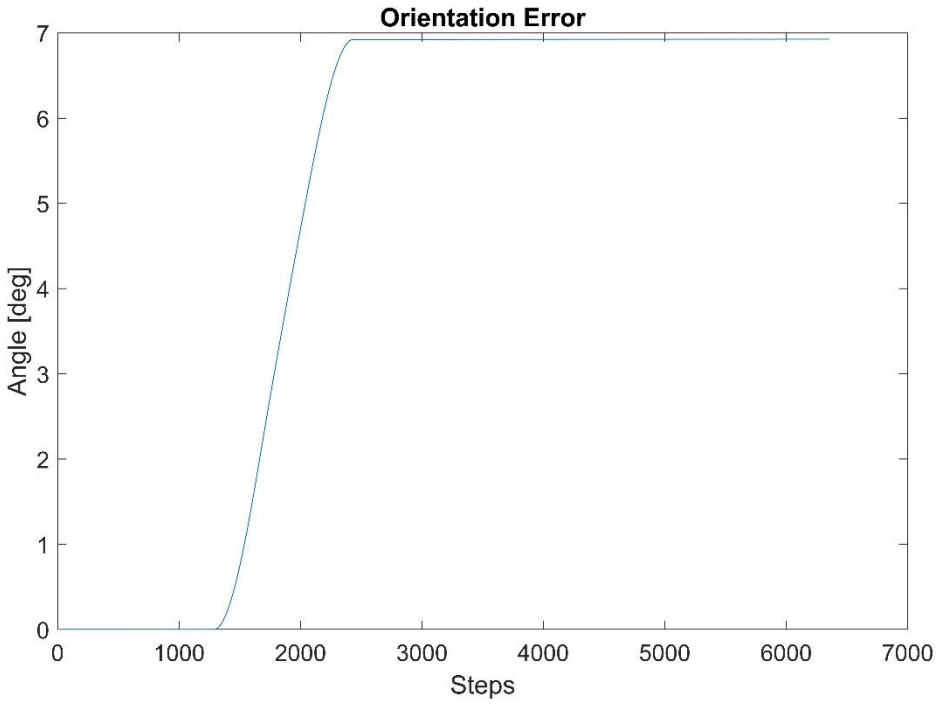

Figure 4.36 : Orientation error: circular path with rotation about z-axis.

Because of the relatively high orientation error, it is possible to represent the orientation difference between initial and compensated paths in **Figure 4.37**.

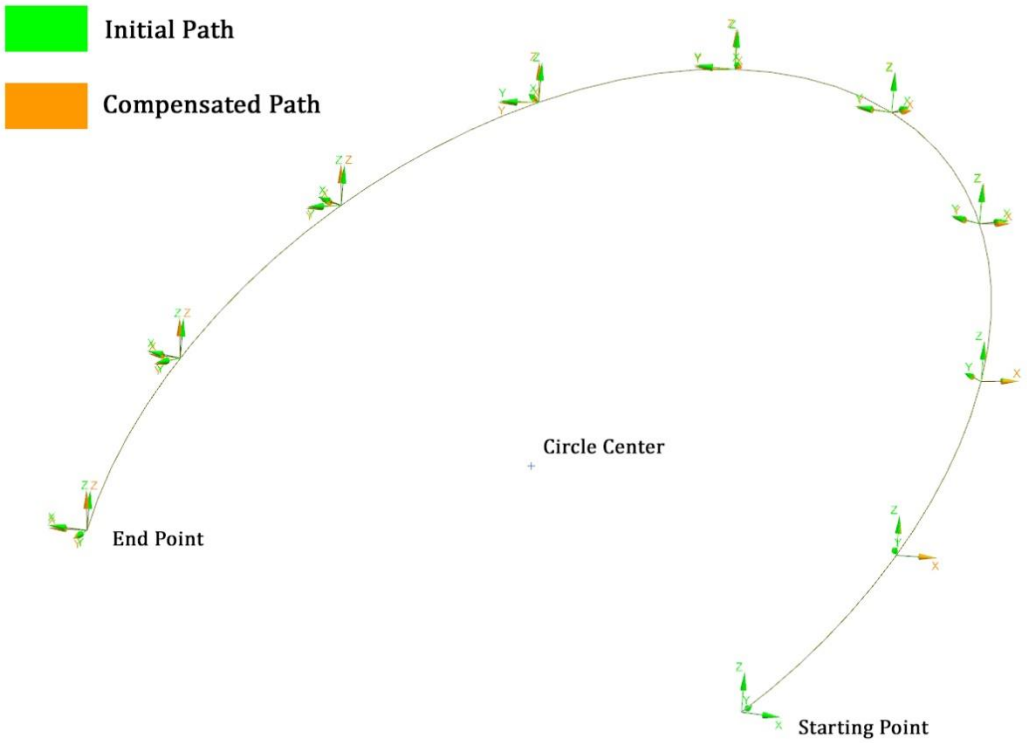

Figure 4.37 : Initial and compensated path representations: circular path with rotation about z-axis.

4.3 Process Speed Comparison

Here, we are going to test one of our example paths with different process speeds. As stated above, our example paths have the same process speed of 250 mm/s. We are going to increase to 1000 mm/s for the first case and decrease the 50 mm/s for the second.

First Case:

For the 1000 mm/s process speed, user-defined parameters are selected as below.

$$\eta_{accUsage} = \%20, \qquad \eta_{acc} = \%20, \qquad \eta_{vel} = \%80$$

The sample time is also decreased to $\Delta t = 0.00025\ s$ to obtain the same step count. Generated trajectories, position and orientation error results are given respectively in **Figure 4.38**, **Figure 4.39,** and **Figure 4.40.**

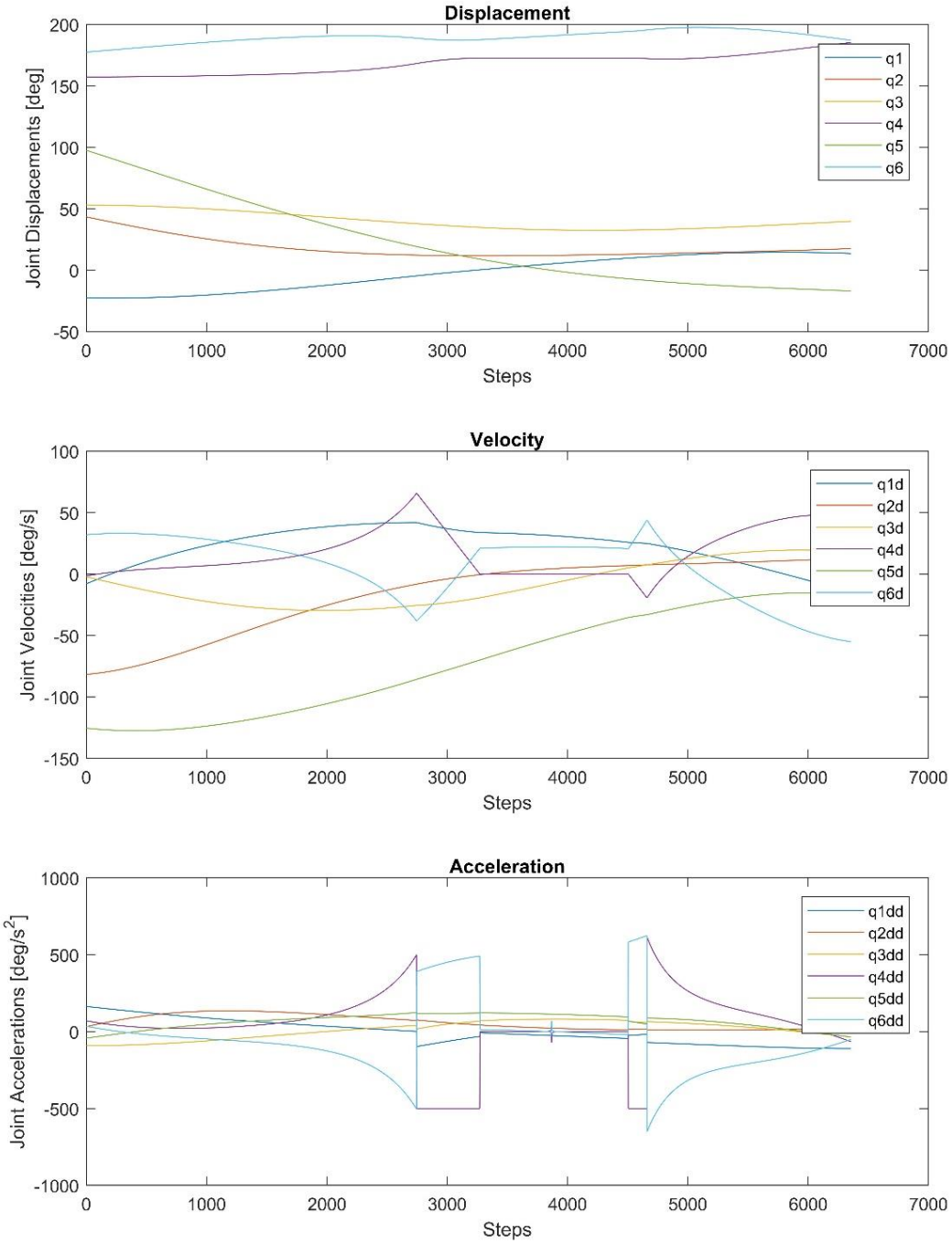

Figure 4.38 : Joint trajectories: high-speed circular path with rotation about y-axis.

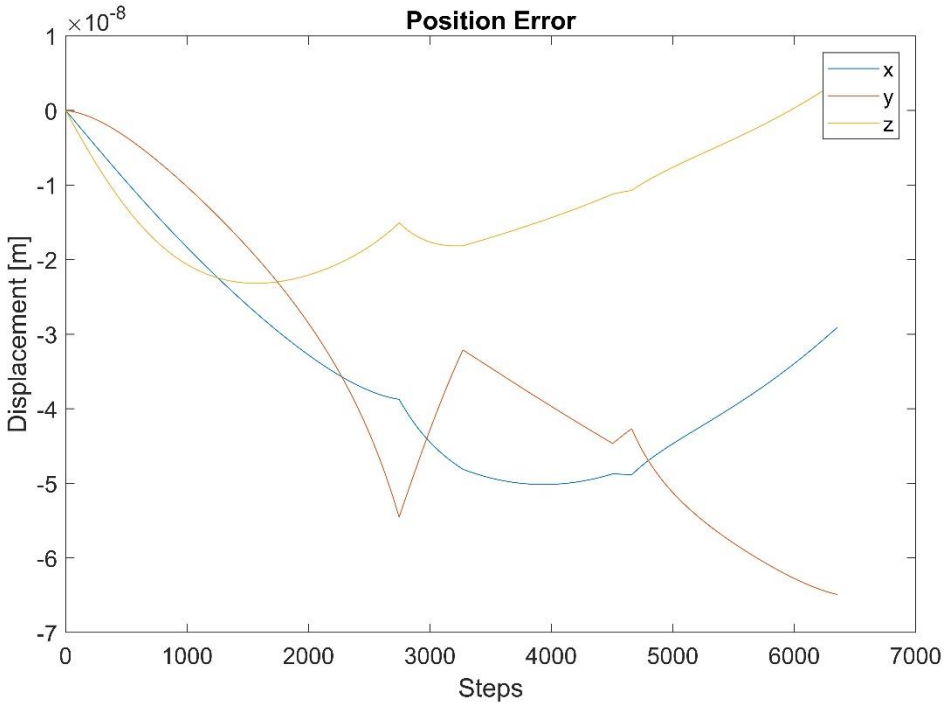

Figure 4.39 : Position error: high-speed circular path with rotation about y-axis.

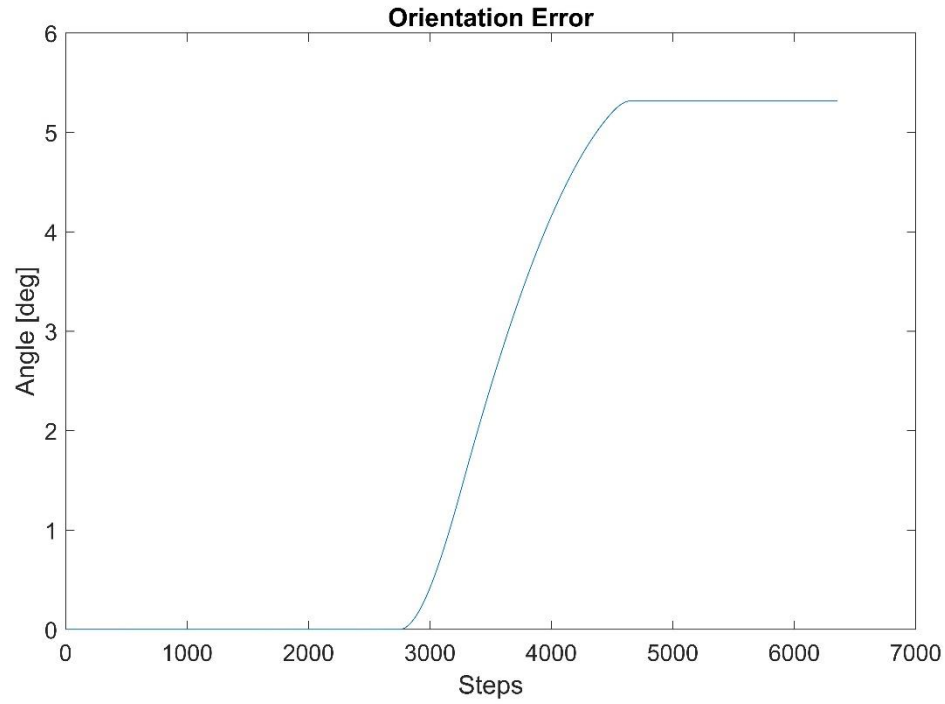

Figure 4.40 : Orientation error: high-speed circular path with rotation about y-axis.

Second Case:

For the 50 mm/s process speed, user-defined parameters are selected as below.

$$\eta_{accUsage} = \%1, \qquad \eta_{acc} = \%1, \qquad \eta_{vel} = \%4$$

The sample time is also increased to $\Delta t = 0.005\, s$ to obtain the same step count. Generated trajectories, position and orientation error results are given respectively in **Figure 4.41**, **Figure 4.42,** and **Figure 4.43.**

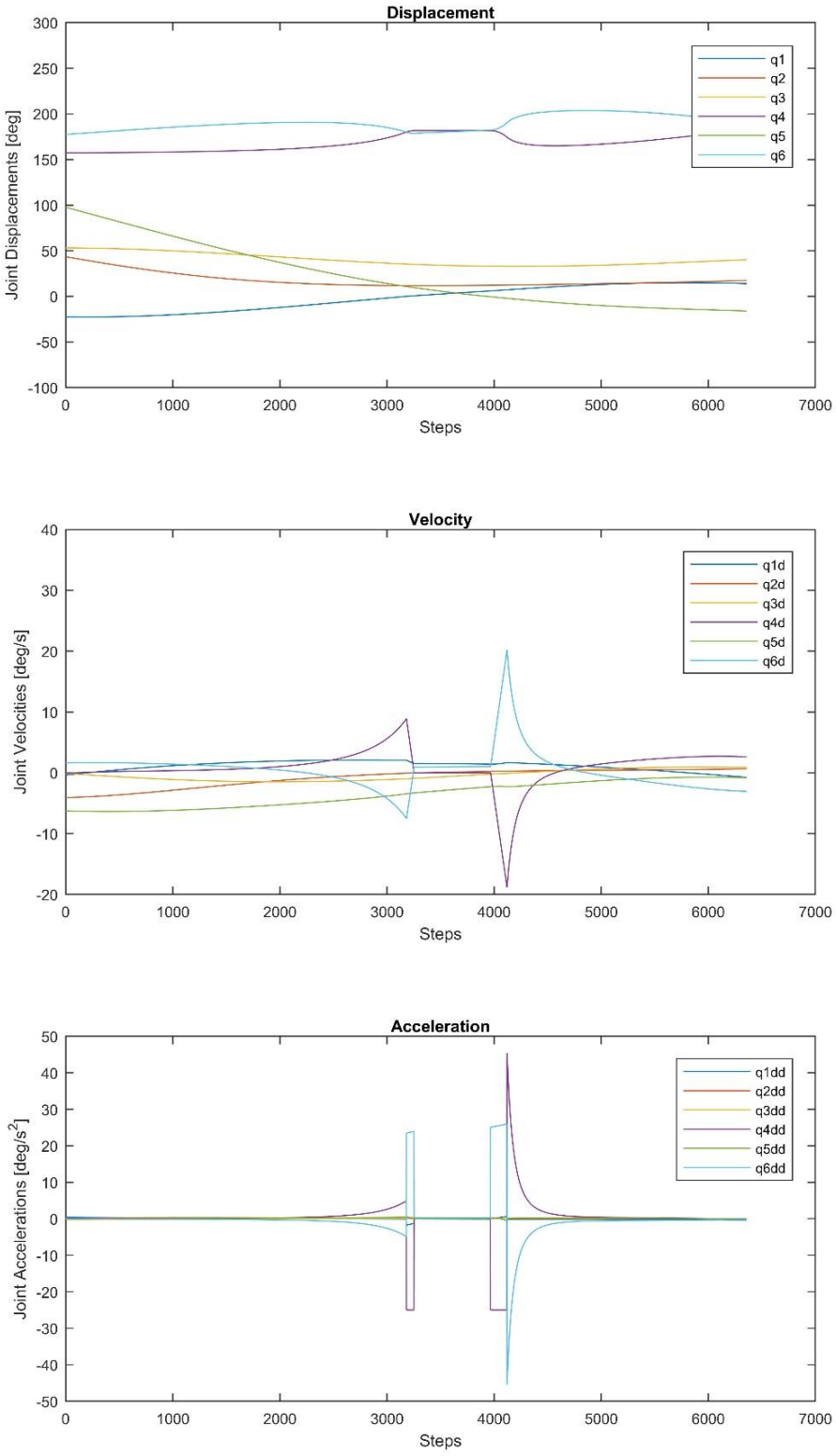

Figure 4.41 : Joint trajectories: low-speed circular path with rotation about y-axis.

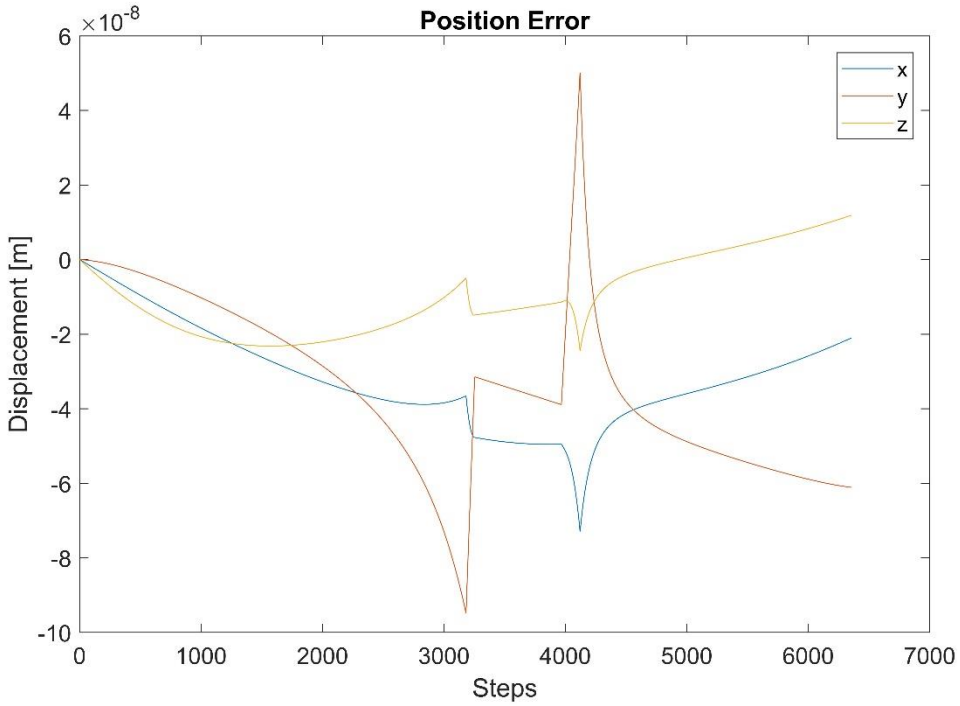

Figure 4.42 : Position error: low-speed circular path with rotation about y-axis

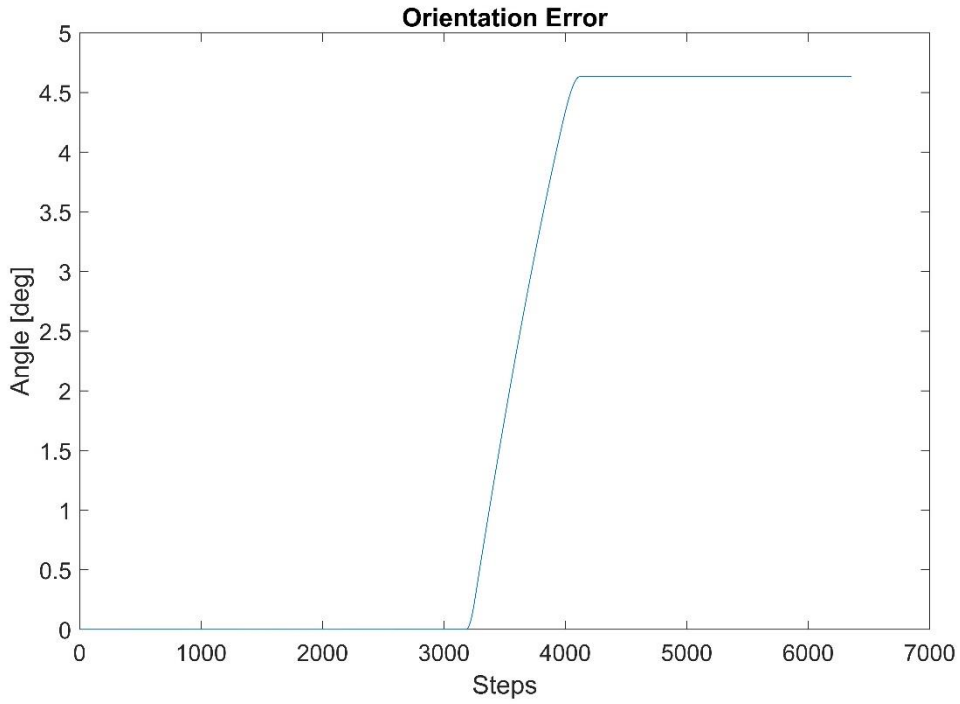

Figure 4.43 : Orientation error: low-speed circular path with rotation about y-axis.

As it can be seen on the trajectories, the main difference occurred in the acceleration and velocity levels of the joints. The obtained position and orientation errors are not very distinctive which makes our method to be used in any operation regardless of the process speed.

5. CONCLUSION

5.1 The Originality of Thesis

This thesis study inspired by several scholar's works and the related sources should be mentioned. In the introduction section, the definition of the redundancy is made almost the same as given by Huo and Baron [31]. Their work clarified the ambiguity over task-related and manipulator-related redundancy. Furthermore, using projection matrices to separate instantaneous angular velocity is given by them. The angle-axis rotation representation that is given in **Section 2.2** is mostly taken from Angeles's book [40]. However, taking it into the quaternions deeply, the transformation between rotation matrices and quaternions are our self-study. In robot modeling, we have continued with the angle-axis rotation representation for the FGM despite the generic DH method. The encouragement to work with this representation is provided to us firstly by Murray et al. [44]. Besides, Angeles used a similar framework. For the IGM, a standardized Pieper method has been used. For methodology, works from Brandstötter et al. [27] and Khalil et al. [28] were inspiring. In velocity and acceleration analysis sub-sections, main arrangements are taken from Angeles, whereas, quaternion inclusion belongs to us. Inverse kinematic solutions are originally possessed by Whitney, the methodology is taken from Angeles. At last, the path planning section is our self-study.

If we get to the originality of our method for wrist singularity, it actually combines studies of older methods with the task re-construction that we have newly proposed. The degenerate axis had been already defined by Aboaf and Paul [18]. However, separating the instantaneous angular velocities and accelerations of wrist joints as feasible and infeasible is our contribution to the literature. Then, eliminating the infeasible rotations by adding the compensation rotations into the task is also our original content to solve the wrist singularity of industrial robots.

5.2 Comparison of Our Method with Others

As we stated earlier, Jacobian related methods do not guarantee a specified accuracy when passing through wrist singularity. Moreover, they require lots of user-defined parameters to manipulate robot Jacobian e.g. damping factor, weighting matrix, etc. Hence, they are not well-accepted methods for passing through wrist singularity.

For the task-related methods, there has not been a method that uses the same robot jacobian when passing through wrist singularity until our method. Those methods show unnatural behaviors such as fixing joint velocities at a specified value or idle rotations through the redundant axis. Whereas, our method does not change any stages of the inverse kinematic solution, because it is not affected by ill-conditioned Jacobian. Elimination of the infeasible direction rotations from the task provides smooth path-tracking juts like being at the out of wrist singular areas. This is carried out by only rotation compensation through an axis. Hence, there is not any additional position error at the tracking stage.

However, orientation error exists when compensation is applied through the parallel infeasible rotation axis. Whereas, it is not much as the other methods do. For industrial applications, such as arc welding and sealing, some orientation variations can be tolerated as long as position accuracy is satisfied. In simulations, the highest orientation error was 7 degrees for a difficult task. Thus, we think our method is suitable for these applications.

On the contrary, if the redundant axis rotation is feasible enough, it would be possible to compensate through redundant rotation axis which can remove wrist singularity effects completely.

Our method requires only 4 user-defined parameters: acceleration usage, acceleration limit threshold, velocity limit threshold, and redundancy threshold which are very natural and easy to specify according to the robot we are using or the application we are doing. Henceforth, taking all comparisons into the account, our method is very advantageous rather than others.

5.3 Future Work

Consequently, we have managed to obtain successful results in simulations. Whereas, our method should be tested in more complex tasks. In our examples, we have created linear and circular position paths with a basic orientation path. It is required to test our method with the splines curves with highly varying orientations. The results should be compared and checked for industrial needs.

In acceleration trajectories, there are discontinuities because of the applied compensation. Mathematical models have been created for both the robot side and path planning side should be upgraded to the jerk level in order to compensate more smoothly. Further, it would help to obtain more realistic simulations.

There is also a need for determining boundaries of wrist singular areas automatically which requires further study. It can help the reduction of user-defined parameters into 2 from 4 because the joint limits threshold would be unnecessary.

Moreover, research can be continued to the robots that are having different kinematic designs. For instance, the following question should be answered: can our method work in the wrist structures that joint axes do not coincide at one point?

REFERENCES

[1] **Moran, M. E.** (2018). History of Robotic Surgery, *Robotics in Genitourinary Surgery 2nd Ed., Chp. 1.* Springer.

[2] **Selig, J. M.** (1992). *Introductory Robotics.* (p.1-6). Prentice Hall.

[3] **Tsai, L.** (1999). *Robot Analysis: The Mechanics of Serial and Parallel Manipulators.* John Wiley & Sons, Inc.

[4] **Engelberger, J. F.** (1999). Historical Perspective and Role in Automation, *Handbook of Industrial Robotics 2nd Ed., Chp. 1.* John Wiley & Sons, Inc.

[5] **Spong et al.** (2005). *Robot Modelling and Control.* (p.1-3). John Wiley & Sons, Inc.

[6] **Url-**1 <*https://www.blackdragonpress.co.uk/collections/sam-chivers/products/rossums-universal-robots*>, date retrieved 04.24.2019.

[7] **Url-**2 <*https://robotics.kawasaki.com/en1/anniversary/history/history_02.html*>, date retrieved 04.25.2019.

[8] **Gasparetto, A. & Scalera, L.** (2019). A Brief History to Industrial Robotics in the 20th Century. *Advances in Historical Studies, 8, 24-35*. https://doi.org/10.4236/cm.2019.81002

[9] **Url-**3 <*https://new.abb.com/products/robotics/home/about-us/historical-milestones*>, date retrieved 04.25.2019.

[10] **Hägele et al.** (2016). Industrial Robotics. *Handbook of Robotics 2nd Ed, 54, 1385-1388.* Springer.

[11] **Url-**4 <*https://ifr.org*>, date retrieved 04.25.2019.

[12] **Url-**5 <*https://new.abb.com/products/robotics/industrial-robots/irb-6700*>, date retrieved 04.25.2019.

[13] **Yoshikawa, T.** (1985). Manipulability and Redundancy Control of Robotic Mechanism. https://doi.org/10.1109/ROBOT.1985.1087283

[14] **Maciejewski, A. & Klein, C.** (1985). Obstacle Avoidance for Kinematically Redundant Manipulators in Dynamically Varying Environments. https://doi.org/10.1177/027836498500400308

[15] **Nakamura, Y. & Hanafusa, H.** (1986). Inverse Kinematic Solutions with Singularity Robustness for Robot Manipulator Control. ASME. *J. Dyn. Sys., Meas., Control.* 1986;108(3):163-171. DOI:10.1115/1.3143764.

[16] **Hollerbach, J. & Suh, K.** (1987). Redundancy Resolution of Manipulators Through Torque Optimization. IEEE Journal on Robotics and Automation, vol. 3, no. 4, pp. 308-316. DOI:10.1109/JRA.1987.1087111

[17] **Nakamura, Y., Hanafusa, H. & Yoshikawa, T.** (1987). Task-Priority Based Redundancy Control of Robot Manipulators. *The International Journal of Robotics Research*, 6(2), 3–15. https://doi.org/10.1177/027836498700600201

[18] **Aboaf, E. & Paul, R.** (1987). Living with the singularity of robot wrists. *IEEE International Conference on Robotics and Automation*, Raleigh, NC, USA, 1987, pp. 1713-1717. DOI: 10.1109/ROBOT.1987.1087792

[19] **Chiaverini, S. & Egeland, O.** (1990). A solution to the singularity problem for six-joint manipulators, *IEEE International Conference on Robotics and Automation*, Cincinnati, OH, USA, 1990, pp. 644-649 vol.1. DOI: 10.1109/ROBOT.1990.126056

[20] **Baron, Luc.** (2000). A Joint-Limits Avoidance Strategy for Arc-Welding Robots. *International Conference on Integrated Design and Manufacturing in Mechanical Engineering*, Montreal, Canada.

[21] **Khalil, W. & Dombre, E. (**2002). *Modeling, Identification and Control of Robots*. *3rd Ed*. p-68. Taylor & Francis, Inc. Bristol, PA, USA. ISBN: 1560329831.

[22] **Cheng, H. & Gupta, K.** (1989). A Historical Note on Finite Rotations. DOI: 10.1115/1.3176034.

[23] **Sarabandi S, Thomas F.** (2019). A Survey on the Computation of Quaternions From Rotation Matrices. *ASME. J. Mechanisms Robotics*.;11(2):021006-021006-9. DOI:10.1115/1.4041889.

[24] **Shoemake, K.** (1985). Animating rotation with quaternion curves. *SIGGRAPH '85 Proceedings of the 12th annual conference on Computer graphics and interactive techniques*. (p. 245-254). DOI:10.1145/325334.325242

[25] **Hanson, A.** (2006). *Visualizing Quaternions, 1st ed*. (p. 96). ISBN: 13: 978-0-12-088400-1.

[26] **Spong et al.** (2005). *Robot Modelling and Control.* (p. 47). John Wiley & Sons, Inc.

[27] **Brandstötter, M. et al.** (2014). An Analytical Solution of the Inverse Kinematics Problem of Industrial Serial Manipulators with an Ortho-parallel Basis and a Spherical Wrist. *Proceedings of the Austrian Robotics Workshop*.

[28] **Khalil, W. & Dombre, E. (**2002). *Modeling, Identification and Control of Robots*. *3rd Ed*. (p. 395-399). Taylor & Francis, Inc. Bristol, PA, USA. ISBN: 1560329831.

[29] **Graf, B.** (2007). Quaternions and Dynamics. arXiv:0811.2889. <https://arxiv.org/abs/0811.2889v1>.

[30] **Nakamura, Y.** (1991). *Advanced Robotics: Redundancy and Optimization*. Addison-Wesley Publishing Company, Inc. ISBN: 0-201-15198-7.

[31] **Huo, L. & Baron, L.** (2005). Kinematic Inversion of Functionally-Redundant Serial Manipulators: Application to Arc-Welding. *Transactions of the Canadian Society for Mechanical Engineering, 2005, 29:679-690,* https://doi.org/10.1139/tcsme-2005-0045

[32] **Campa, R., Camarillo, K. & Arias, L.** (2006). Kinematic Modeling and Control of Robot Manipulators via Unit Quaternions: Application to a Spherical Wrist. *Proceedings of the 45th IEEE Conference on Decision and Control*, San Diego, CA, 2006, pp. 6474-6479. DOI: 10.1109/CDC.2006.377155

[33] **Liégeois, A.** (1977). Automatic Supervisory Control of the Configuration and Behavior of Multibody Mechanisms. *IEEE Transactions On Systems, Man, And Cybernetics, Vol. Smc-7. No. 12*, pp. 868-871.

[34] **Yoshikawa, T.** (1984). Analysis and Control of Robot Manipulators with Redundancy. *Robotics Research: The First International Syposium, Brady and Paul, Eds.* MIT Press, Cambridge, MA, pp. 735-747.

[35] **Chiaverini, S., Siciliano, B. & Egeland, O.** (1994). Experimental Results on Controlling a 6-DOF Robot Manipulator in the Neighborhood of Kinematic Singularities. *Experimental Robotics III. Lecture Notes in Control and Information Sciences, vol 200*. Springer, Berlin, Heidelberg. https://doi.org/10.1007/BFb0027580

[36] **Url-**6 <*https://www.waybuilder.net/free-ed/SkilledTrades/Welding/10GMAW/10GMAW.asp*>, date retrieved 01.01.2020.

[37] **Mathers, G.** (2002). Resistance welding processes. *The Welding of Aluminium and Its Alloys*, pp. 166–180. https://doi.org/10.1533/9781855737631.166

[38] **Trinh, C. et al.** (2015). A Geometrical Approach to the Inverse Kinematics of 6R Serial Robots With Offset Wrists. *Volume 5C: 39th Mechanisms and Robotics Conference*. DOI: 10.1115/detc2015-47950

[39] **Jazar, R. N.** (2011). *Advanced Dynamics: Rigid Body, Multibody, and Aerospace Applications*. (p. 699). John Wiley & Sons.

[40] **Angeles, J.** (2014). *Fundamentals of Robotic Mechanical Systems: Theory, Methods, and Algorithms.* Springer International Publishing. DOI: 10.1007/978-3-319-01851-5.

[41] **Geu Flores, F. et al.** (2019). Generalization of the Virtual Redundant Axis Method to Multiple Serial-Robot Singularities. *ROMANSY 22 – Robot Design, Dynamics and Control. CISM International Centre for Mechanical Sciences (Courses and Lectures), vol 584.* Springer, Cham. https://doi.org/10.1007/978-3-319-78963-7_62

[42] **Geu Flores, F. et al.** (2016). Robust Inverse Kinematics at Position Level by Means of the Virtual Redundant Axis Method. *ROMANSY 21 - Robot Design, Dynamics and Control, CISM International Centre for Mechanical Sciences (Courses and Lectures), vol 569.* Springer, Cham. https://doi.org/10.1007/978-3-319-33714-2_3

[43] **Pires, J. N., Loureiro, A. & Bolmsjö, G.** (2010). *Welding robots: Technology, system issues and applications: With 88 figures*. London: Springer. DOI: 10.1007/1-84628-191-1.

[44] **Murray, R. M., Li, Z., & Sastry, S. S.** (1994). *A mathematical introduction to robotic manipulation*. Boca Raton, FL: CRC. ISBN: 978-0-8493-7981-9.

Cansın Akarsu

MECHANICAL ENGINEER (B.Sc.), MECHATRONICS ENGINEER (M.Sc.)

Summary

An engineer, have knowledge of mechatronic system design, integration, and production; experienced in various CAD software; love and be capable of coding, have started my professional career more than 4 years ago. I endeavor to progress in control and robotic areas academically, have individual studies on creating mathematical models of robotic systems, developing algorithms for controlling these, and making simulations. By considering the investment in myself, my prior career goal is working in R&D departments.

Contact

+90 533 425 3613
akarsuca@gmail.com
Çankaya, Ankara

Personal Info

Date of Birth: 22/10/1993

Birth Place: Kadıköy, Istanbul

Marital Status: Single

Nationality: Turkish

Military Service: Done

Driving License Type: B

Language

Turkish: C2

English: C1

German: A2

Education

Master's Degree:
09/2016 – 01/2020 **Mechatronics Engineering**, 3.06/4.00
İstanbul Technical University

Bachelor's degree:
09/2011 – 06/2016 **Mechanical Engineering**, 2.85/4.00
İstanbul Technical University

High School:
09/2007 – 06/2011 **Math and Science**, 75/100
İstanbul Köy Hizmetleri And. High School

Software Knowledge

Advanced: Tecnomatix Process Simulate, NX, Matlab, Simulink, Autocad, MS Office
Intermediate: Solidworks, C Programming, V-rep, VBA, Abaqus, Adams
Beginner: Catia, LaTex

Exams

E-YDS – ÖSYM, Score: 80 Date of exam: 10/2019
Bulats – Cambridge English, Score: 75 Date of exam: 01/2017
ALES – ÖSYM, Score: 78 Date of exam: 05/2016

Work Experience

- **R&D Engineer** – 07/2020 - Still
  Roketsan Missiles Inc., Lalahan, Ankara

- **System Design Engineer** – 04/2016 - 02/2019
  Altınay Robot Technologies Inc., Tuzla, İstanbul
  Tasks:
  - Designing robot integrated assembly lines in 3D, with respect to norms and standards
  - Calculation of the cycle time for each station for each process
  - Making kinematic analyses of robots and other equipment: models, reachability control, collision control etc.
  - Preparation of robot integrated process simulations
  - Determining process requirements and project follow-up
  - Determining equipment that are going to use in the process
  - Having offers for the supply equipment
  - Guiding mechanical designers for the equipment that are going to be designed and produced, controlling accordance with the process
  - Creating IDs for equipment and BOM lists on Teamcenter, sending them into IFS.
  - Support for commissioning works, providing risk analysis, fmea and other customer requirements.

  Main projects that I participate:
  - Martur – Front and back seat, Welding Assembly Line (Romania, Morocco)
  - Ford Sollers – Kuga, Framing Assembly Line (Russia)
  - Benteler – Torsion Profile, Welding and Sanding Assembly Line (Germany)
  - Benteler – Side Arm, Welding Cell (Germany)
  - Gestamp – Front Subframe, Welding Assembly Line (Gebze)
  - Renault – Rear Twist Beam, Welding Cells (Bursa)
  - Gestamp – Front Subframe, Welding Assembly Line (Gebze)
  - Tofaş – Egea, Bodyside Closures Assembly Line (Bursa)

- **Work and Travel** – 05/2015 - 09/2015
  Roma NY Pizza, Virginia Beach, USA

- **Engineering Intern** – 08/2014 - 09/2014
  Parsan Inc., Pendik, İstanbul
  Made my production internship here, reported following processes that I had observed.
  - Axle forging
  - Forging die production
  - Heat treatment
  - Hob screw-cutting
  - CNCs and other machining processes

Additional Info

Hobbies: History, Swimming sports, Bass guitar, Chess

Member of: ITU Water Polo Team