Modular and Analytical Methods for Solving Kinematics and Dynamics of Series-Parallel Hybrid Robots

von Shivesh Kumar

Dissertation

zur Erlangung des Grades eines Doktors der Naturwissenschaften
- Dr. rer. nat. -

Vorgelegt im Fachbereich 3 (Mathematik & Informatik)
der Universität Bremen
im September 2019

Datum des Promotionskolloquiums: 15. November 2019

Gutachter: Prof. Dr. Frank Kirchner (Universität Bremen)
Prof. Dr. Andreas Müller (Johannes Kepler Universität Linz)

Abstract

While serial robots are known for their versatility in applications, larger workspace, simpler modeling and control, they have certain disadvantages like limited precision, lower stiffness and poor dynamic characteristics in general. A parallel robot can offer higher stiffness, speed, accuracy and payload capacity, at the downside of a reduced workspace and a more complex geometry that needs careful analysis and control. To bring the best of the two worlds, parallel submechanism modules can be connected in series to achieve a series-parallel hybrid robot with better dynamic characteristics and larger workspace. Such a design philosophy is being used in several robots not only at DFKI (for e.g., Mantis, Charlie, Recupera Exoskeleton, RH5 humanoid etc.) but also around the world, for e.g. Lola (TUM), Valkyrie (NASA), THOR (Virginia Tech.) etc. These robots inherit the complexity of both serial and parallel architectures. Hence, solving their kinematics and dynamics is challenging because they are subjected to additional geometric loop closure constraints. Most approaches in multi-body dynamics adopt numerical resolution of these constraints for the sake of generality but may suffer from inaccuracy and performance issues. They also do not exploit the modularity in robot design. Further, closed loop systems can have variable mobility, different assembly modes and can impose redundant constraints on the equations of motion which deteriorates the quality of many multi-body dynamics solvers. Very often only a local view to the system behavior is possible. Hence, it is interesting for geometers or kinematics researchers, to study the analytical solutions to geometric problems associated with a specific type of parallel mechanism and their importance over numerical solutions is irrefutable. Techniques such as screw theory, computational algebraic geometry, elimination and continuation methods are popular in this domain. But this domain specific knowledge is often underrepresented in the design of model based kinematics and dynamics software frameworks. The contributions of this thesis are two-fold. Firstly, a rigorous and comprehensive kinematic analysis is performed for the novel parallel mechanisms invented recently at DFKI-RIC such as RH5 ankle mechanism and Active Ankle using approaches from computational algebraic geometry. Secondly, the general idea of a modular software framework called Hybrid Robot Dynamics (HyRoDyn) is presented which can be used to solve the geometry, kinematics and dynamics of series-parallel hybrid robotic systems with the help of a software database which stores the analytical solutions for parallel submechanism modules in a configurable and unit testable manner. HyRoDyn approach is suitable for both high fidelity simulations and real-time control of complex series-parallel hybrid robots. The results from this thesis has been applied to two robotic systems namely Recupera-Reha exoskeleton and RH5 humanoid. The aim of this software tool is to assist both designers and control engineers in developing complex robotic systems of the future. Efficient kinematic and dynamic modeling can lead to more compliant behavior, better whole body control, walking and manipulating capabilities etc. which are highly desired in the present day and future robotic applications.

Zusammenfassung

Während serielle Roboter für ihre Vielseitigkeit in verschiedensten Anwendungen einen größeren Arbeitsbereich, einfachere Modellierung und Steuerung bekannt sind, haben sie bestimmte Nachteile, wie z.B. eine begrenzte Präzision, geringere Steifigkeit und schlechte dynamische Eigenschaften im Allgemeinen. Ein paralleler Roboter kann eine höhere Steifigkeit, Geschwindigkeit, Genauigkeit und Nutzlastkapazität unter dem Nachteil eines reduzierten Arbeitsbereichs bieten, wie auch eine komplexere Geometrie, die eine sorgfältige Analyse und Regelung erfordert. Um die Vorteile aus parallelen und seriellen Robotern gemeinsam zu nutzen, können diese zu hybriden Strukturen kombiniert werden, die gute dynamische Eigenschaften bei möglichst große Arbeitsbereich aufweisen. Eine solche Designphilosophie kommt nicht nur am DFKI in mehreren Robotern zum Tragen (z.B. Mantis, Charlie, Recupera Exoskeleton, RH5 humanoid etc.), sondern wird weltweit in Robotern, wie z.B. Lola (TUM), Valkyrie (NASA), THOR (Virginia Tech.) etc. genutzt. Diese Roboter weisen die Komplexität sowohl serieller als auch paralleler Architekturen auf. Daher ist die Lösung ihrer Kinematik und Dynamik eine Herausforderung, da sie zusätzlichen geometrischen Einschränkungen geschlossener kinematischer Ketten unterliegen. Die meisten Ansätze in der Mehrkörperdynamik nutzen die numerische Lösung dieser Zwangsbedingungen im Interesse allgemeiner Lösbarkeit , können dabei aber größere Fehler bei längerer Rechenzeit aufweisen. Außerdem können sie mögliche Modularitäten des Roboters nicht ausnutzen. Darüber hinaus können geschlossene Systeme eine variable Mobilität aufweisen, wie auch verschiedene Montagemodi, was den Bewegungsgleichungen redundante Einschränkungen auferlegen kann und die Qualität vieler Solver für Mehrkörpersystem beeinträchtigt. Sehr oft ist nur eine lokale Sicht auf das Systemverhalten möglich. Daher ist es für Geometer oder Kinematiker interessant die analytischen Lösungen für geometrische Probleme zu untersuchen, welche einer bestimmten Art von Parallelmechanismus zugrundeliegen. Analytische Lösungen sind in ihrer Bedeutung gegenüber numerischen Lösungen unabdingbar . Techniken wie Schraubentheorie, algebraische Geometrie, Eliminierungs- und Fortführungsmethoden sind in diesem Bereich beliebt. Aber dieses domänenspezifische Wissen ist beim Entwurf modellbasierter Kinematik- und Dynamik-Software-Frameworks oft unterrepräsentiert. Die Beiträge dieser Arbeit sind zweigeteilt. Erstens wird eine rigorose und umfassende kinematische Analyse für neuartige, kürzlich am DFKI-RIC erfundene Parallelmechanismen wie dem RH5-Fußgelenkmechnismus und dem Active Ankle unter Verwendung von Ansätzen aus der algebraischen Geometrie und der Schraubentheorie durchgeführt. Zweitens wird die allgemeine Idee eines modularen und aufteilbaren Software-Frameworks namens Hybrid Robot Dynamics (HyRoDyn) vorgestellt, welches für die Lösung der Geometrie, Kinematik und Dynamik von seriell-parallelen, hybriden Robotersystemen mit Hilfe einer Software-Datenbank verwendet werden kann, dass die analytischen Lösungen für parallele Submodule in konfigurierbarer und einheitenprüfbarer Weise speichert. Der HyRoDyn-Ansatz eignet sich sowohl für Simulationen mit hoher Wiedergabetreue, als auch für die Echtzeit-Steuerung komplexer, seriell-paralleler Hybridroboter. Die Ergebnisse dieser Arbeit wurden auf zwei Robotersysteme angewendet, nämlich dem Recupera-Reha Exoskelett sowie dem Humanoiden RH5 . Ziel dieses Softwaretools ist es, sowohl Konstrukteure als auch Reglungstechniker bei der Entwicklung komplexer Robotersysteme der Zukunft zu unterstützen. Effiziente kinematische und dynamische Modellierung kann zu einem verbesserten nachgiebigen Verhalten, einer besseren Ganzkörperkontrolle, Geh- und Manipulationsmöglichkeiten usw. führen, die in der heutigen und zukünftigen Robotik sehr relevant sind und sein werden.

Danksagung

The work presented in this PhD dissertation is an outcome of continual work and draws support from various sources. Obligations thus incurred in completing the work have been many. It is therefore almost impossible to express adequately the debts owed to many persons who have been instrumental in imparting this work a successful status. It is however a matter of great pleasure to express my gratitude and appreciation to those who have been contributing to bring about this project.

Firstly, I would like to thank my first supervisor and institute director Prof. Dr. Frank Kirchner for giving me an opportunity to work on this interesting topic at the DFKI Robotics Innovation Center. His vision for the future of robotics has been truly influential for me and I borrow great inspiration from his “framework” thinking to not only give structure to various aspects of my work but also to put it in a bigger context. I am very grateful for his unconditional support and encouragement throughout the course of this thesis. As rightly said in German, he has lived the image of a perfect “doktorvater” for me. Also, I truly enjoyed both formal and informal discussions with him while at work or at the PhD retreats. Then, I would like to thank my second supervisor Prof. Dr. Andreas Mueller (Director, Institute of Robotics, Johannes Kepler University, Linz) for providing his continous guidance and constructive feedback during this thesis. I have borrowed much of the mathematical aspiration of my work from his work. I believe that his ascetic approach to scientific research and willingness to dedicate his valuable time in nurturing young minds without seeking much material benefit is a gift to many of us. He has lived the image of a true “guru” in my life. Also, thank you very much to Frank for letting me visit Andreas and to Andreas for hosting me in Linz in September 2017 and June 2019 both of which were extremely productive times in the context of my thesis.

Secondly, I would like to take this opportunity to thank my first two mentors Dr. Jose de Gea Fernandez and Dr. Bertold Bongardt (now at Institut für Robotik und Prozessinformatik, TU Braunschweig) at DFKI-RIC when I started to work here back in October 2015. Jose is the person who interviewed me during my job application and invited me to work at this institute. I discussed my first ideas with him to which he was very welcoming and always supported me at this institute in his role as a team leader by putting me in the correct projects which suited my interests and skillset and catered to the requirements of my PhD topic. Its a great pleasure to share the position of deputy team leader with him and I sincerely hope that we can do some great work together in the area of mechanics and control and solve some interesting problems in this area. Bertold is the person who mentored me very closely right from the very beginning, introduced me to his work and in general to many interesting works in the world of robot geometry and kinematics. I would always be thankful for all the time he spent with me. It was his abstract mathematical thinking which rekindled my interest in pure as well as applied mathematics. I authored my first papers with him and learnt the art of scientific writing from him. His feedback has always been crucial for improving my work. When I look back, I would not be where I am without the support of my early mentors, so once again my big thanks to both of you in nurturing my career at DFKI-RIC.

Thirdly, I would like to express my gratitude to my project leaders Elsa Andrea Kirchner (Recupera-Reha), Stefan Stiene and Peter Kampmann (D-RoCK), Sirko Straube and Hendrik Woehrle (Q-RoCK) who gave me the intellectual freedom and space to work on my ideas and use them to realize the project goals. I consider myself fortunate to be a part of these projects which provided me interesting problems to work upon, solve them and also disseminate my contributions to the scientific community through various mediums. None of this would be possible without these projects and their leaders who worked very hard to bring these projects to our institute. The work presented in this thesis started in the Recupera-Reha project, matured in D-RoCK and concluded in Q-RoCK project. These projects were funded by the German Aerospace Center (DLR) with federal funds from the Federal Ministry of Education and Research (BMBF) (Grant Numbers: 01-IM-14006A, 01IW15001 and FKZ 01IW18003 respectively). I would like to acknowledge their support.

Fourthly, I would like to thank various brilliant colleagues with whom I collaborated closely during the course of this thesis. I will start with the colleagues from the team system design. In fact, it is their creativity in developing novel parallel mechanisms which raised interesting questions about their geometry and motion. I would especially like to mention my former colleague Marc Simnofske who invented the Active Ankle mechanism which attracted me towards the topic of this thesis. He had a great intuition in design and I was always impressed with how he can come up with interesting mechanisms without having to mathematically study them. My current colleagues like Heiner Peters, Martin Mallwitz and Christopher Schulz follow his legacy and I truly enjoy working with them. I would also like to thank my Master thesis student turned colleague turned friend Christoph Stoeffler who took the time to proofread my thesis and provide me his valuable feedback. He also helped me setup the study group on “Differential Geometry” which brought together like-minded people from our institute such as Julius Martensen, Luis Octavio Arriaga, Laura Breitkopf, Daniel Harnack etc. to learn this amazing subject together. Then, I would also like to thank the three Indian colleagues Ajish, Rohit and Sankar who have been of great help right from the beginning. These were the people I could reach anytime and ask any questions regarding work or life in Germany. Also, they helped me a lot while learning the RoCK middleware and cleared my programming doubts. I have a genuine appreciation for their friendliness and patience. I also had the chance to start some external collaborations during my PhD. Especially, I would like to mention my former colleague and friend Abhilash Nayak who introduced me to algebraic geometry and took his time out to work with me on many problems. I hope we will continue to collaborate in this area in the future. Lastly, I would also like to thank my students in particular Jan Frederik Bode, Durgesh Salunkhe, Anirvan Dutta who all worked very hard and contributed to my work indirectly.

PhD life won’t be fun without the support of friends and family. I would like to thank my special friend Sahar Mahdie for sharing various ups and downs of a PhD life while walking in the Bürgerpark. I am also grateful to my friends in Bremen Anke, Rosa, Veronika, Ingo, Zeynep, Chandani, Vipul for creating some beautiful memories outside yet parallel to my PhD life. My parents have always supported me unconditionally, despite many hardships they had to face. I would be eternally grateful to my family (especially my mother) for their love and encouragement in shaping me into who I am today.

Contents

Part I

Introduction and State of the Art

Chapter 1

Introduction

This chapter is the entry point of this thesis and serves three main purposes: 1) it provides the motivation of the topic and illustrates its relevance, 2) it establishes the main objectives of this thesis and sets it into the context, 3) it gives a brief description of the methodology developed and highlights the main contributions. Further, it describes the structure of the thesis with a short summary of the content of each chapter which lets the readers easily navigate through the document. Also, it notes the list of scientific publications that were a result of this work.

1.1 Motivation

The thesis is generally motivated to provide a systematic treatment to the increasing complexity of robotic systems. In particular, last decades have witnessed the rise of series-parallel hybrid robot designs which inherit the complexity of both serial and parallel designs. Starting with an introduction to this design architecture in Section 1.1.1, the complexity of these robotic systems is highlighted (Section 1.1.2) along with the need to standardize their design and control with modern software architectures (Section 1.1.3).

1.1.1 Series-Parallel Hybrid Robots

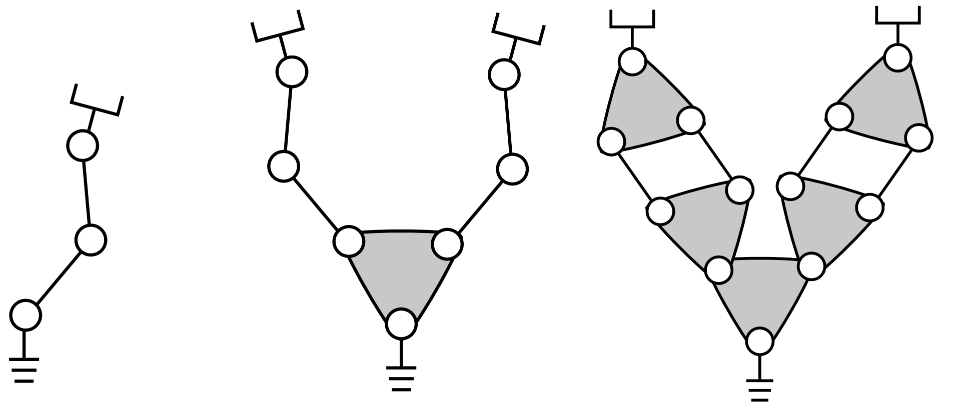
Definition 1 *A serial robot is a mechanical system that is designed as a series of links connected by motor-actuated joints that extend from a base to a single end-effector.*

Definition 2 *A tree-type robot is a mechanical system that is designed as a series of links connected by motor-actuated joints that extend from a base to multiple end-effectors.*

Serial and tree-type robots are well known for their versatility in applications, large workspace, simple modeling and control. Hence, they often represent the state-of-the-art in robotic systems. However, they generally feature only limited precision, low structural stiffness, and poor dynamic characteristics. For these reasons, robots based on a pure tree type topology suffer from speed and torque limitations. Fig. 1.1a and Fig. 1.1b show schematics of exemplary serial and tree-type systems respectively.

Definition 3 *A parallel robot is a mechanical system that uses several serial chains to support a single platform, or end-effector.*

In contrast to serial robots, parallel robots can provide higher stiffness, speed, accuracy, and payload capacity. On the downside, they possess a reduced workspace and a more complex geometry which requires careful analysis and control. Parallel robots have been traditionally used in more tailored use cases such as fast pick-and-place applications [Brinker and Corves, 2015], driving simulators [Stewart, 1965], fast orientation devices [Gosselin and Hamel, 1994] etc. Table 1.1 presents an overview of comparison between serial and parallel robots. It can be noticed that advantages and disadvantages of two designs are almost complemenatry.



(a) Serial robot (b) Tree-type robot (c) Series-parallel hybrid robot

Figure 1.1: Serial, tree-type and series-parallel hybrid robots

Definition 4 *A series-parallel hybrid robot is defined as a robot which is built from a serial or tree-type combination serial and parallel mechanisms.*

Series-parallel hybrid designs (see Fig. 1.1c) combining the advantages of serial and parallel topologies are common in the field of heavy machinery, e.g., cranes, excavator arms, bulldozers etc. for a long time. Such designs have also recently caught the attention of robotics researchers from the industry and

Table 1.1: Comparison of serial / tree-type robots and parallel robots

Serial / Tree-type Robots	Parallel Robots
+ State of the art, easier to control	– Complex geometry, difficult to control
+ Large workspace	– Limited workspace, singularities
– Inherent low stiffness	± High stiffness (inhomogeneous)
– Lower precision	+ Higher precision
– Lower payload capacity	+ Higher payload capacity
– Acceleration limits	+ High accelerations (up to 100 g)

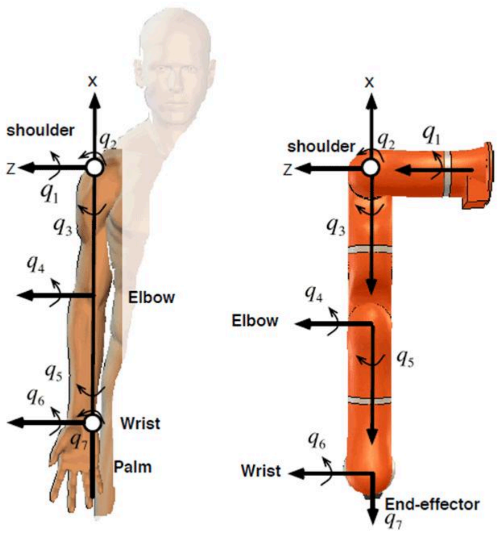
academia. For instance, the stiffness of an industrial manipulator can be significantly improved by including a simple parallelogram mechanism. In particular, industrial robots as ABB's IRB4400, IRB6660, KUKA's KR 40-PA., KR 50-PA., KR 700-PA robots, and Comau's Smart NJ series, SR400 utilize this design concept [To and Webb, 2012], [Gautier et al., 1995]. Logabex's LX4 robot piles four stewart platform modules in series to achieve a redundant series-parallel hybrid manipulator with large workspace and good payload (75 Kg) to weight (120 Kg) ratio [Mer, 2006]. In academics and R&D, the idea to use closed loop mechanisms and Parallel Kinematic Machines (PKM) has been utilized more liberally, giving rise to a number of biologically inspired lightweight robotic systems with good dynamic characteristics. Some prominent examples including Stewart platform, 2SPU+1U [1] [Serracín et al., 2012], double parallelogram linkage [Kumar et al., 2017a] etc. have been used in various hybrid robots like hominid CHARLIE [Kuehn et al., 2014], multi-legged robot MANTIS [Bartsch et al., 2016], AXO-SUIT [Bai et al., 2017] and humanoid robot LOLA [Lohmeier et al., 2006], THOR [Lee et al., 2014] etc. The design motivation of such hybrid robots is evident: use of PKM-based submechanisms helps designers to achieve lightweight, modular and compact design while enhancing the stiffness and dynamic characteristics of the robot.

1.1.2 Complexity in Kinematic and Dynamic Modeling

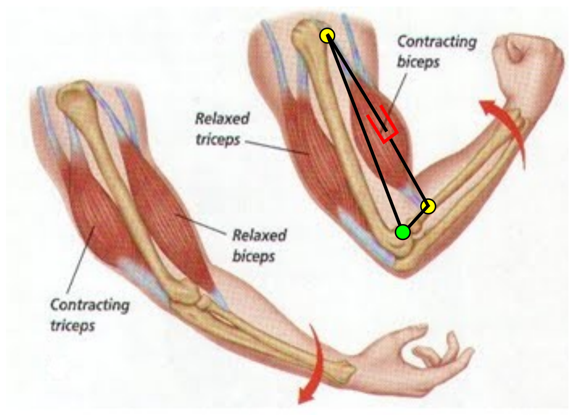
Our expectations from robotic systems have been steadily increasing. Present day robots are expected to be fast, agile, safe, soft, precise, compliant, predictable and intelligent at the same time which can also sometimes sound contradictory. To address these demands, the robotic systems need to have an intelligent design as the intelligence in behavior and control can not alone meet the growing expectations.

Nature has always captivated the attention of roboticists and inspired the design of robotic systems. Most industrial robotic arms (see Fig. 1.2a) include a shoulder-

[1] In mechanism theory, it is typical to identify mechanisms using their type. For details, see [Kong and Gosselin, 2007].



(a) Anthropomorphic Arm
[Wang and Artemiadis, 2013]



(b) Elbow joint in human
(Adapted from [Arts2Science, 2018])

Figure 1.2: Biologically inspired design in robotics

elbow-wrist architecture similar to humans abstracted with either seven revolute joints grouped in spherical (S) - revolute (R) - spherical (S) joint pairs or six revolute joints grouped in universal (U) - revolute (R) - spherical (S) joint pairs. An example of the former is KUKA iiwa LBR robot and the latter is Staubli RX90 robot. An advantage of this architecture is the principle solvability of its inverse kinematics problem. This kind of architecture is inspired from biology but mostly on the surface. Looking under the skin, one finds various muscle groups actuating a certain joint in rather a series-parallel architecture. Fig. 1.2b shows the elbow joint in human arm with the biceps and triceps muscles connected with the rigid skeleton with the help of tendons. This allows for intelligent placement of the actuators along the links and improve the payload to weight ratio of the arm. This is becoming the design inspiration in modern robotics and designers around the world are trying to abstract different kinematic joints with the help of parallel mechanisms (see right side of Fig. 1.2b).

On one hand series-parallel hybrid designs combine the advantages of serial and parallel architectures, on the other, they inherit the kinematic complexity of both designs. Observing this development in robotics, it can be noticed that robots are becoming increasingly complex for modeling and control. It is well known that the inverse kinematics problems for serial chains are difficult to solve and forward kinematics problems are difficult to solve for the parallel robots [Nielsen and Roth, 1999]. Hence, both forward and inverse kinematics problems are difficult to solve for series-parallel designs. While numerical tools for solving such problems exist, they often provide only a *local* view to their kinematic behavior. Tailored analytical solutions to these problems can provide deeper and *global* insights into their kinematics. Hence, it is interesting for geometers and kinematics researchers, to study the analytical solutions to geometric problems associated with a specific type of parallel mechanisms using methods from screw theory, computational algebraic geometry, distance geometry etc. Their importance over numerical solutions is irrefutable. A novel 3 DOF parallel mechanism called ACTIVE ANKLE [Simnofske, 2015] (see Fig. 1.3a) was invented to design a modular spherical joint for hip and ankle applications in a full body exoskeleton. Similarly, a novel 2 DOF orientational parallel mechanism as shown in Fig. 1.3b has been developed for ankle joint of a humanoid robot at DFKI-RIC. Forward and inverse problems related to their geometry and kinematics have not been studied previously.

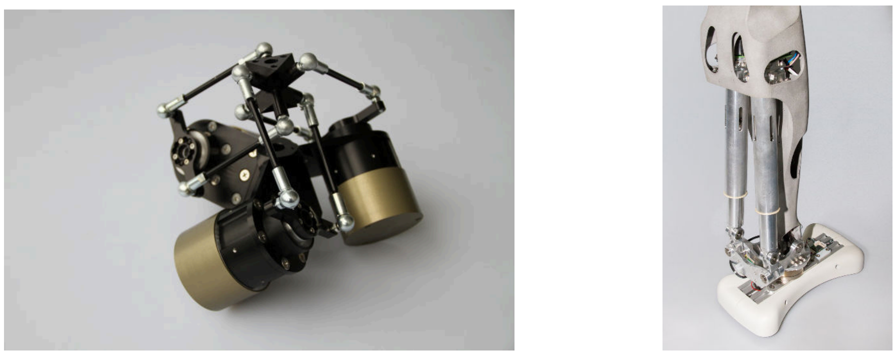

(a) 3 DOF ACTIVE ANKLE mechanism (b) 2 DOF RH5 ANKLE mechanism

Figure 1.3: Newly invented orientational parallel mechanisms at DFKI-RIC

Due to difficulties in modeling of series-parallel hybrid robots, their full dynamic model is not exploited. For example, in most cases these robots are often limited to position control (e.g. MANTIS, CHARLIE, SHERPATT [Cordes et al., 2018]) where a joint to actuator mapping is utilized to achieve a kinematic control of the robot. When equipped with real time dynamic control, an inverse dynamic model in generalized coordinates (neglecting the contribution from the closed loops) is often combined with an inverse static model in actuation space to compute the actuator forces [Hopkins et al., 2015], [Vonwirth, 2017], [Paine et al., 2015]. This approach is used in torque controlled series-parallel hybrid humanoids such as THOR, Valkyrie, Lola etc. An obvious advantage of this simplification is the reduced CPU time needed to solve the inverse dynamic model but it leads to unnecessary increase of PD gains for achieving the desired controller tracking performances. The trade offs of this model simplification have not been studied systematically.

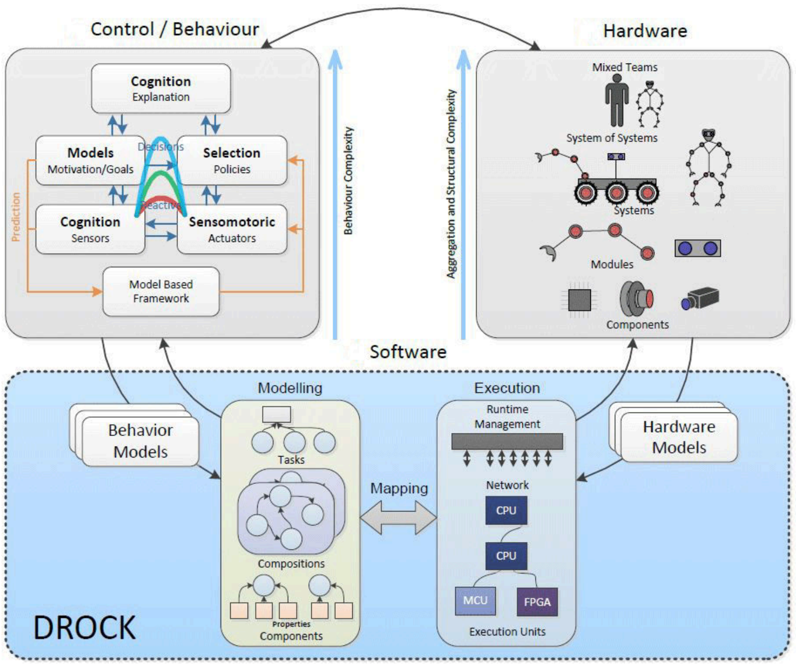


Figure 1.4: D-RoCK (Credits: Jakob Schwendner, DFKI GmbH)

1.1.3 Model Based Software Development of Complex Robots

Building a robot is a sophisticated process which requires knowledge from different domains like mechanics, electronics, computer science etc. Hence, there is a strong need to bring together domain specific knowledge from the experts and to make available to public use. The rise of open source robotics initiatives are serving to this purpose. A major challenge in robotics is to develop systems efficiently and cost effectively, which are able to deal with this complexity in a robust manner. Software plays a decisive role in dealing with the complexity of these systems. The x-RoCK project series[2] is concerned with tools and methods for developing software for robots. The concepts of modularization and modeling are used to make the process manageable. The modularity must be reflected in the design of hardware (mechanical structure, electronics) and software components (low level controllers, kinematics and dynamic modeling, behaviour modeling etc.) which constitute a robotic system. While modularization enables efficient reuse of components, modeling describes how these components can be used in a given context. RobMoSys [RobMoSys, 2018] is another European project in this direction.

The prime focus of the first project in x-RoCK project series namely D-RoCK is the aspect of model based software development in the context of mobile manipulation systems [D-RoCK, 2018]. The modeling comprises software, hardware and be-

[2]x-RoCK is a project series at DFKI-RIC involving D-RoCK [D-RoCK, 2018], Q-RoCK [Q-RoCK, 2019] etc.

havior of the systems. Behavior is mapped onto software, and software is mapped onto hardware. At run-time the system state is aligned with the models, to allow monitoring and dynamic system reconfiguration. Fig. 1.4 demonstrates the three different aspects i.e. hardware, software and behavior in the development of robots and the inter-connections between them. A kinematics and dynamics library is one of the most crucial links between the hardware and behavior models and is an integral part of the robot motion software domain. It is needed because it provides a bi-directional mapping between the task space and actuator space of the robots. While the task space is a natural setting to describe high level behaviors, the actuator space is necessary to perform low level control of the robot. Most kinematics and dynamics libraries do not exploit the modularity in robot design which is particularly important in the modeling and control of series-parallel hybrid robots. Further, the loop closure constraints are resolved numerically which may lead to inaccuracy and poor performance.

1.2 Objectives

The general objective of this thesis is to provide a systematic treatment of kinematic and dynamic modeling of series-parallel hybrid robotic systems so that the methods developed in this thesis can be used by designers and control engineers to develop and control the complex robotic systems of the future. First, an extensive survey on the various multi-dof joints used in series-parallel hybrid robots is performed to identify a notion of modularity. Based on the survey, two main objectives of this thesis are identified:

1. rigorous kinematic analysis of two novel parallel mechanisms using modern kinematics approaches
   (a) **Active Ankle** is a 3 DOF parallel mechanism which behaves in an almost spherical manner and can be used as an abstraction to spherical joints (see Fig. 1.3a). It finds applications in hip and ankle designs of the Recupera Exoskeleton.
   (b) **RH5 Ankle** is a 2 DOF parallel mechanism which can be used as an abstraction to universal joint (see Fig. 1.3b). It finds applications in the design of wrist, ankle and torso joints in the RH5 humanoid.
2. develop a software workbench for kinematic and dynamic modeling of series-parallel hybrid systems which allows
   (a) modular composition of multi-dof serial or parallel joints

(b) reuse of the closed form solutions derived for specific types of parallel mechanisms

(c) model abstraction by exploiting the hierarchy in robot design

The methods and software framework developed in this thesis are applied to the modeling and control of following two series-parallel hybrid robotic systems recently developed at DFKI-RIC.

1. **Recupera-Reha Exoskeleton** has 32 active DOFs and is built by combining several higher DOF joint modules: a Stewart platform of type 6-UPS in torso, a double parallelogram in shoulder for flexion–extension movement and ACTIVE ANKLE mechanism as a 3 DOF joint in hip and ankle. Its modular design allows it to be used in either the wheelchair configuration (see Fig. 1.5a) or the full body configuration (see Fig. 1.5b).

2. **RH5 Humanoid**, as shown in Fig. 1.5c, is a lightweight and biologically inspired humanoid robot which uses linkages and PKM modules for most of its joints for e.g. hip flexion-extension, knee, ankle, torso and wrist. The robot also has 32 active DOFs.

An outlook of this work is to study the model simplification in the dynamic control of series-parallel hybrid robots. Overall, this work aims to address various aspects related to the kinematic and dynamics analyses of series-parallel hybrid robots.

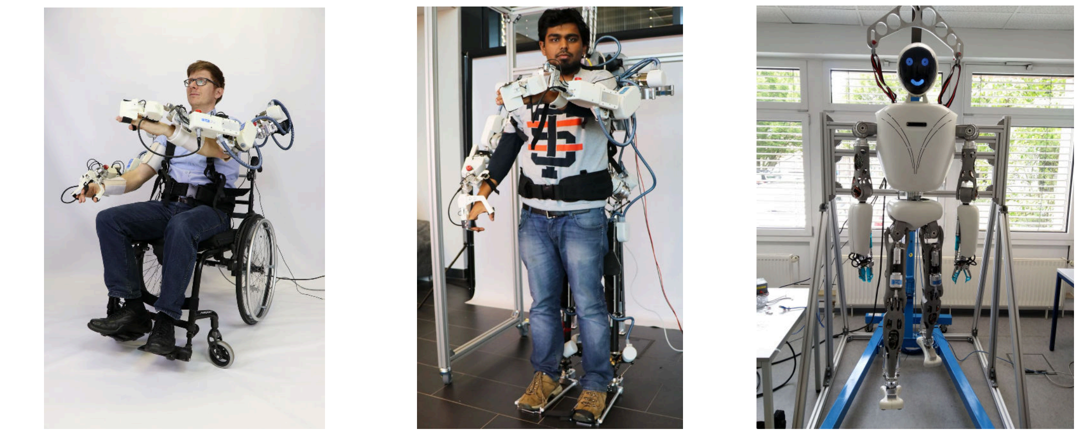

(a) Recupera Wheelchair Exo (b) Recupera Full Body Exo (c) RH5 Humanoid

Figure 1.5: System applications: Recupera Exoskeleton and RH5 Humanoid

1.3 Contributions

To identify the notion of modularity, a systematic survey of various series-parallel hybrid robotic systems developed in the field of legged robotics, exoskeletons and industrial automation is performed in mechanics, electronics and software domains [Kumar et al., 2019c]. It was found that most of these hybrid robots utilize parallel submechanisms as an abstraction to a higher degree of freedom active joint (for e.g. universal, spherical, six dof). Further, they let the designers exploit the non-linear transmission, enhance the stiffness and dynamic capabilities of the robot and produce a light weight modular design.

The first main contribution of the work includes the study of two novel parallel mechanisms, namely RH5 ANKLE [Kumar et al., 2018c] and ACTIVE ANKLE [Simnofske et al., 2016], which provide an abstraction to universal and spherical joints respectively. Their designs were optimized for their tailored use cases and a rigorous kinematic analysis was performed using approaches from modern kinematics. Their forward and inverse kinematics problems were solved and their configuration spaces and work spaces are characterized [Kumar et al., 2018a, Kumar et al., 2018c]. In particular, methods from computational algebraic geometry were used to answer questions regarding their global kinematic behavior [Kumar et al., 2018b, Kumar et al., 2018c]. Further, loop closure functions (LCF) for these mechanisms were derived based on such analyses.

The second main contribution of this work is the development of an analytical and modular software workbench called Hybrid Robot Dynamics (HyRoDyn) [Kumar et al., 2018d] which can be used to solve the kinematics and dynamics of series-parallel hybrid robots. The overall concept of HyRoDyn software framework is shown in Fig. 1.6. While numerical resolution of loop closure constraints can always be chosen as a first approach for studying the kinematic behavior of a parallel mechanism, to get a full understanding of its behavior requires rigorous kinematic analysis. Once, the mechanism has been analysed, this knowledge is made reusable with the help of subsystem software database. The main idea here is to store the analytically derived loop closure functions (LCF) in a *configurable* mechanism library which is identified by its type (for e.g. RH5 ANKLE [Kumar et al., 2018c]). Based on submechanisms defined in a hybrid robot, it can modularly compose the LCF of the overall system in an automated way. The resulting loop closure Jacobian has a block diagonal structure that can be exploited in the computation of various forward and inverse kinematics and dynamics algorithms. The software is implemented in C++ and utilizes $O(n)$ multi-body dynamics algorithms for tree type systems from RBDL [Felis, 2017](based on [Featherstone, 2008]). Presently, closed form solutions to mechanisms such as 1-RRPR, 2-SPU+1U, 2-SPRR+1U, 6-RUS, 6-UPS, parallelogram chains are available in its submechanism libraries and the software can be used to analytically solve the kinematics and dynamics of arbitrary series-parallel hybrid robots composed of these submechanism modules. Actuation of the robot can be arbitrarily selected. The software has been successfully used in the analysis and control of some complex series parallel hybrid robots such as the Recupera-Reha exoskeleton [Kumar et al., 2019b] and the RH5 humanoid. Further, it can also be used to generate Look Up Tables (LUT) for different submechanism modules which allow for distributed computation of robot's kinematics.

As an outlook, this research also addresses the model simplification tradeoff for series-parallel hybrid systems by introducing some error and performance metrics for variable fidelity modeling of these systems [Kumar et al., 2019a].

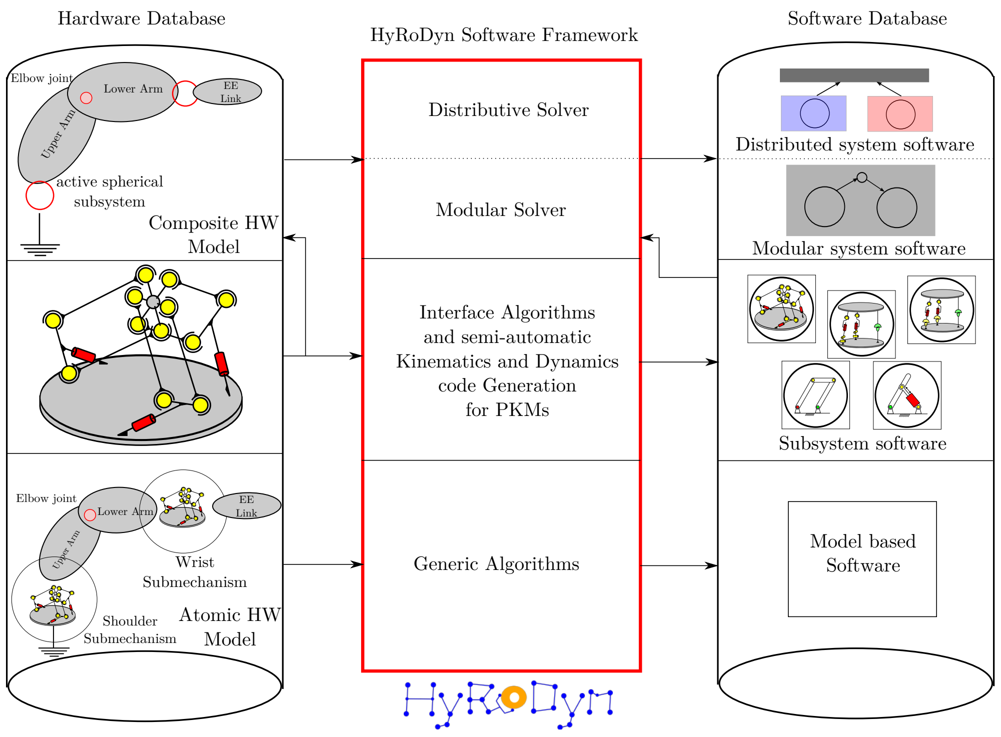


Figure 1.6: HyRoDyn concept: HyRoDyn software framework acts between a hardware database (which stores robot descriptions as models) and software database (which stores closed form solutions to loop constraints of various parallel submechanism modules). Based on the modular description of the hardware model, it can modularly compose the loop closure function of the hybrid composition and transfer them to various kinematics and dynamics algorithms.

1.4 Structure

This thesis is organized in a total of 10 chapters which are categorized into 5 parts namely, I. Introduction and SOTA, II. Geometric Analysis, III. Kinematics and Dynamics, IV. HyRoDyn Design and Applications and V. Conclusion. Fig. 1.7 shows the overall outline of this thesis. In the following, a summary of the each chapter is provided which lets the readers easily navigate through the thesis document.

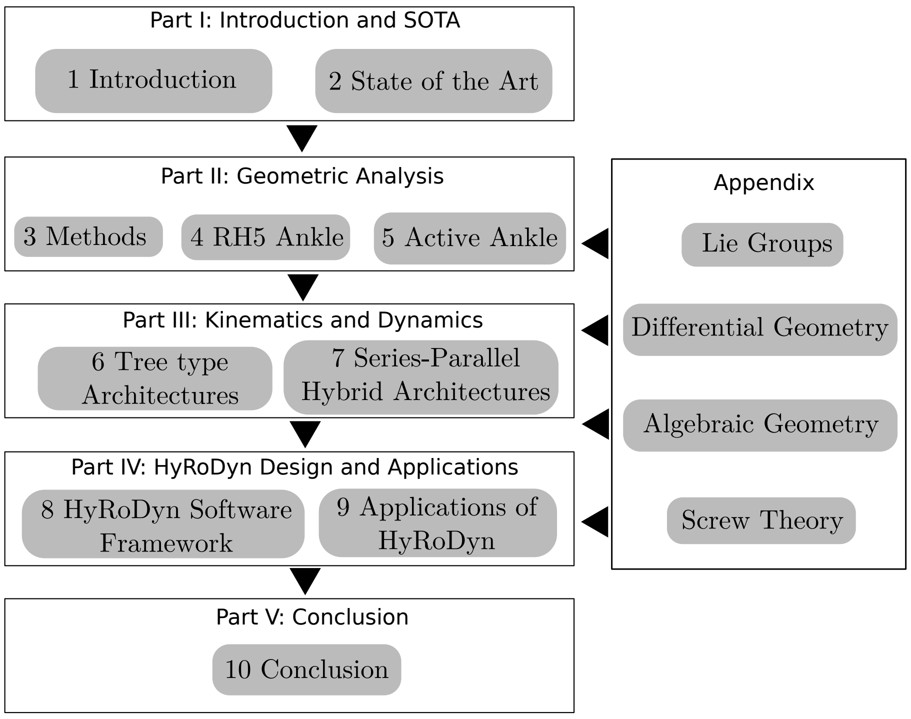

Part I: Introduction and SOTA
1 Introduction
2 State of the Art
Part II: Geometric Analysis
3 Methods
4 RH5 Ankle
5 Active Ankle
Part III: Kinematics and Dynamics
6 Tree type Architectures
7 Series-Parallel Hybrid Architectures
Part IV: HyRoDyn Design and Applications
8 HyRoDyn Software Framework
9 Applications of HyRoDyn
Part V: Conclusion
10 Conclusion
Appendix
Lie Groups
Differential Geometry
Algebraic Geometry
Screw Theory

Figure 1.7: Thesis Outline

Part I: Introduction and SOTA collects the Chapters 1 & 2 of this thesis which provide an introduction and study of state of the art in this area.

Chapter 1 (Introduction) serves as an entry point to the thesis and presents the motivation, objective and contributions of this work. It also provides details about the structure of the thesis and how the scientific results from this thesis are disseminated.

Chapter 2 (State of the Art) provides a state of the art survey on various series-parallel hybrid robots in various application domains. Further, it studies the modular and distributive aspect of their design in both hardware and software domains. It also provides a survey on various approaches used in their modeling and control.

Part II: Geometric Analysis collects the Chapters 3, 4 and 5 of this thesis as they essentially present the geometric analyses of parallel mechanisms.

Chapter 3 (Modern Approaches in Geometric Analysis) provides an overview of modern approaches in the geometric analysis of closed loop mechanisms.

Chapter 4 (Study of 2SPRR+1U Device for Abstraction of Universal Joint) presents a comprehensive geometric analysis of a two DOF ankle mechanism which can be used as an abstraction to universal joint. The chapter introduces the mechanim's architecture, solutions to forward and inverse geometric problems and workspace analysis using tools from computational algebraic geometry. Variants of this mechanism are used in RH5 humanoid in the ankle, wrist and torso designs.

Chapter 5 (Study of 3R-[2SS] Device for Abstraction of Spherical Joint) presents a thorough geometric analysis of a three DOF ankle mechanism which can be used as an abstraction to a spherical joint. The chapter introduces the mechanism's architecture, solutions to forward and inverse geometric problems and workspace analysis using tools from computational algebraic geometry. A novel technique for solving its rotative inverse geometric problem using the concept of virtual joints is also discussed which enables the real time control of this mechanism. This mechanism has been used in Recupera Exoskeleton in hip and ankle designs.

Part III: Kinematics and Dynamics collects the Chapters 6 and 7 which together describe the modular recursive methods for solving the kinematics and dynamics of series-parallel hybrid robots.

Chapter 6 (Screw Theory and Lie Group Methods for Tree Systems) presents the screw theory and Lie group methods for multi-body dynamics of tree type systems. The provided treatment is recursive in nature and rooted in graph theoretic description of such systems. Its application in solving the forward and inverse dynamics problems is presented.

Chapter 7 (Modular and Analytical Methods for Series-Parallel Hybrid Systems) presents the modular and analytical methods for solving the kinematics and dynamics of series-parallel hybrid systems. The notion of the modularity is motivated from the robot design and is reflected in the modular enumeration of the graph. This allows a modular composition of loop closure constraints which can be used for modular and recursive computation of kinematics and dynamics. The explanation of the approach is aided with an example of a series-parallel humanoid leg.

Part IV: HyRoDyn Design and Applications is a collection of Chapter 8 and 9 which describe the modular software workbench called HyRoDyn and presents its application in modeling and control of Recupera Exoskeleton and RH5 humanoid.

Chapter 8 (HyRoDyn Software Architecture) presents the architecture of HyRoDyn software. It presents how a robot can be described in HyRoDyn and how such a description can be automatically generated using a visual editor called Phobos. Further, it presents the design of HyRoDyn library in C++ and its integration in RoCK middleware.

Chapter 9 (Results and Applications) presents the results from the application of HyRoDyn in the simulation, analysis and real time control of series-parallel hybrid robots. In particular, these results are obtained by solving various kinematics and dynamics problems for the Recupera Exoskeleton and RH5 humanoid. As an outlook, it also presents some insights into the possibility of model simplification in rigid body dynamics computations in case of series-parallel hybrid robots. Finally, its real world application in the area of robotic rehabilitation is presented.

Part V: Conclusion contains the last and concluding chapter of this thesis.

Chapter 10 (Conclusion and Outlook) presents the summary of the thesis, its scientific results and identifies the future research directions.

Part VI: Appendix collects the four appendices on the mathematical foundations required to properly understand the rigid body motion and analytical mechanics.

Appendix A (Lie Groups) presents an introduction to group theory with a motivation to introduce Lie groups which are important objects in the study of rigid body motion.

Appendix B (Differential Geometry) presents an introduction to differential geometry with a motivation to introduce Lie algebra which is often exploited in kinematic and dynamic modeling presented in Chapter 6 and 7.

Appendix C (Algebraic Geometry) presents an introduction to algebraic geometry which is useful in understanding the algebraic methods for solving loop constraints. Its applications are presented in Chapter 4 and 5.

Appendix D (Screw Theory) presents an introduction to screw theory which is again useful in various formulations presented in Chapter 6 and 7.

Disclaimer on Content Reuse Single sentences as well as entire sections of this thesis are taken from my own publications without explicit quotation either because I am the main author or I contributed the used part to them. The same applies to figures and tables. However, the papers from which the content has been borrowed are clearly cited in the beginning of the chapters or sections. The text from this thesis may be reused in my upcoming publications.

1.5 Dissemination of scientific results

The main scientific results along with their applications from this thesis have been disseminated through various channels like peer-reviewed publications in journals and conferences, poster presentations and invited guest lectures at various research institutes.

1.5.1 Journals

1. Shivesh Kumar, Bertold Bongardt, Marc Simnofske, Frank Kirchner. “Design and kinematic analysis of the novel almost-spherical mechanism Active Ankle”. In: Journal of Intelligent and Robotic Systems (JINT), Springer Nature.

2. Shivesh Kumar, Hendrik Woerhle, Mathias Trampler, Marc Simnofske, Heiner Peters, Martin Mallwitz, Elsa Andrea Kirchner, Frank Kirchner. “Modular Design and Decentralized Control of the Recupera REHA Exoskeleton for Stroke Rehabilitation”, In: MDPI Applied Sciences 2019, 9, 626.

3. Anirvan Dutta, Durgesh Salunkhe, Shivesh Kumar, Arun Dayal Udai, Suril V Shah. “Sensorless Full Body Active Compliance in a 6-DOF Parallel Manipulator”, In: Robotics and Computer-Integrated Manufacturing, Elsevier.

4. Shivesh Kumar, Hendrik Woehrle, Jose de Gea Fernandez, Andreas Mueller, Frank Kirchner. “A Survey on Modularity and Distributivity in Series-Parallel Hybrid Robots”. In: Mechatronics: The Science of Intelligent Machines, Elsevier. [under revision]

5. Shivesh Kumar, Andreas Mueller. “An Analytical and Modular Software Workbench for Solving Kinematics and Dynamics of Series-Parallel Hybrid Robots”. In: Journal of Mechanisms and Robotics, ASME. [under review]

1.5.2 Book Chapters

1. Shivesh Kumar, Abhilash Nayak, Bertold Bongardt, Andreas Mueller, Frank Kirchner. “Kinematic analysis of Active Ankle using computational Algebraic

Geometry". In: Computational Kinematics, Saïd Zeghloul, Lotfi Romdhane, and Med Amine Laribi (Eds.). Springer International Publishing, Cham, 117–125.

2. Shivesh Kumar, Abhilash Nayak, Heiner Peters, Christopher Schulz, Andreas Mueller, Frank Kirchner, Kinematic analysis of a novel parallel 2SPRR+1U ankle mechanism in humanoid robot, In: Advances in Robot Kinematics (ARK 2018) Springer Proceedings in Advanced Robotics, vol 8. Springer, Cham.

1.5.3 Conferences

1. Marc Simnofske, Shivesh Kumar, Bertold Bongardt, Frank Kirchner. "Active Ankle - an Almost-Spherical Parallel Mechanism". In: 47th International Symposium on Robotics (ISR 2016). Munich, Germany, June 2016.
2. Elsa Andrea Kirchner, Niels Will, Marc Simnofske, Luis Manuel Vaca Benitez, Bertold Bongardt, Mario Michael Krell, Shivesh Kumar, Martin Mallwitz, Anett Seeland, Marc Tabie, Hendrik Wöhrle, Mehmed Yüksel, Anke Heß, Rüdiger Buschfort, Frank Kirchner. "Recupera-Reha: Exoskeleton Technology with Integrated Biosignal Analysis for Sensorimotor Rehabilitation". In 2. Transdisziplinäre Konferenz Technische Unterstützungssysteme, die die Menschen wirklich wollen, Hamburg, VDE, pages 504-517, Dec/2016.
3. Shivesh Kumar, Marc Simnofske, Bertold Bongardt, Andreas Mueller, Frank Kirchner. "Integrating Mimic Joints into Dynamics Algorithms - Exemplified by the Hybrid Recupera Exoskeleton". In: Proceedings of AIR 2017. ACM, NY, USA, 6 pages. [2nd Prize in Best Oral Presentations, Springer]
4. Christoph Stoeffler, Shivesh Kumar, Heiner Peters, Olivier Bruels, Andreas Mueller, Frank Kirchner. "Conceptual Design of a Variable Stiffness Mechanism in a Humanoid Ankle using Parallel Redundant Actuation", In: IEEE Humanoids 2018, 06.11.-09.11.2018, Beijing, China.
5. Shivesh Kumar, Andreas Mueller. "An Analytical and Modular Software Workbench for Solving Kinematics and Dynamics of Series-Parallel Hybrid Robots". In: 43rd Mechanisms and Robotics Conference, 2019 ASME IDETC/CIE, Anaheim, California.
6. Shivesh Kumar, Julius Martensen, Andreas Mueller, Frank Kirchner. "Model Simplification For Dynamic Control of Series-Parallel Hybrid Robots - A Representative Study on the Effects of Neglected Dynamics". In: IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS-2019), Macau, China.

1.5.4 Posters

1. Shivesh Kumar, Bertold Bongardt, Marc Simnofske, Frank Kirchner. “Task space controller for the novel Active Ankle mechanism”. In: International Conference on Robotics and Automation for Humanitarian Applications, (IEEE RAHA-16).

2. Shivesh Kumar, Kai Alexander von Szadkowski, Andreas Mueller, Frank Kirchner. “HyRoDyn: A Modular Software Framework for Solving Analytical Kinematics and Dynamics of Series-Parallel Hybrid Robots”, In: IEEE/RSJ International Conference on Intelligent Robots and Systems, (IROS-2018).

1.5.5 Invited Guest Lectures

1. Design, analysis and control of a novel almost spherical mechanism Active Ankle, In Kinematics, Dynamics and Mechatronics in Motion Technology - Seminar RWTH Aachen, 06 December 2017 at Institut für Getriebetechnik, Maschinendynamik und Robotik, RWTH Aachen, Germany.

2. Modular and Distributable approach towards Kinematic and Dynamic modeling of series-parallel Hybrid robots, 11 September 2018 at Institut für Robotik und Prozessinformatik, TU Braunschweig, Germany.

3. A Modular Software Workbench for Kinematic and Dynamic Modeling of Complex Series-Parallel Hybrid Robots, 26 June 2019 at Institute of Robotics (ROBIN), Johannes Kepler University, Linz, Austria.

4. A modular approach for kinematic and dynamic modeling of complex robotic systems using algebraic geometry, 13 July 2019 at the mini-symposium on Algebraic geometry for kinematics and dynamics in robotics at SIAM Conference on Applied Algebraic Geometry, Bern, Switzerland (co-authored with Prof. Andreas Mueller).

5. Overview of Robotics Activities at DFKI-RIC and A Software Architecture for Solving Kinematics and Dynamics of Series-Parallel Hybrid Robots, 05 September 2019 at NASA Jet Propulsion Laboratory, Pasadena, United States.

1.5.6 Tutorial

1. Lie Group Modeling of Robot Kinematics and Dynamics – A Hands-on Introduction at ASME IDETC 2019 conference, Anaheim, United States (co-authored with Prof. Andreas Mueller).

Chapter 2

State of the Art

This chapter presents the state of the art in design and control of series-parallel hybrid robots based on [Kumar et al., 2019c]. The chapter is organized as follows. Section 2.1 provides a survey on various series-parallel hybrid robots in various application domains. Section 2.2 presents the notion of modularity and distributivity in these hybrid robots from a hardware and software perspective. Section 2.3 discusses the modeling and control methods for these robots while highlighting the theoretical challenges and adopted solution approaches.

2.1 Series-Parallel Hybrid Designs

Series-parallel hybrid designs that combine the advantages of serial and parallel topologies are common in the field of heavy machinery, e.g., cranes, excavator arms, bulldozers etc. for a long time. It is only in the last two decades that the robotics community has started to adopt these designs for real applications. The survey presented in this section studies the evolution of series-parallel hybrid robots primarily in various domains: humanoids/bipeds, multi-legged robotic systems, exoskeletons or physical man-machine interfaces and industrial automation.

2.1.1 Humanoids

Humanoids are bipedal robots with upper body, arms and head to mimic human anatomy. These are complex mechatronic systems with highly interdependent technical aspects. The last few decades of research in humanoid robotics have shown that humanoids designed for high dynamic performance require a stiff structure and good mass distribution [Stasse and Flayols, 2019]. Both of these properties can be achieved by utilizing parallel mechanisms in the design. Fig. 2.1a shows the bipedal robot LOLA [Lohmeier et al., 2006] developed in 2006, which is probably the first humanoid robot designed using a modular parallel joint concept. The non-linear transmission between the actuator space and output space in a parallel mechanism can be utilized advantageously by adjusting its design parameters to the torque-speed characteristics of typical movements of the robot [Lohmeier, 2010]. Moreover, utilizing multi-DOF parallel mechanisms can also help in achieving a more uniform load distribution on the actuators. The design of NASA Valkyrie humanoid [Radford et al., 2015] as shown in Fig. 2.1b followed a similar design concept by utilizing PKM modules for its wrist, torso and ankle joints. Similarly, the robot AILA [Lemburg et al., 2011] also used PKM modules for her wrist, neck and torso joints. The torque controlled humanoid TORO from DLR [Englsberger et al., 2014] and TALOS [Stasse et al., 2017] (PAL Robotics) are largely tree type systems but utilize a simple parallelogram linkage in their ankle for creating the pitch movement (see Fig. 2.1c). Humanoid robots THOR [Lee et al., 2014], [Lee, 2014] and SAFFiR [Lahr et al., 2016] from Virginia Tech. use 2 DOF PKM for hip roll-yaw and ankle joints and utilize Hoeken's mechanism for hip pitch and knee joints. LARMbot [Cafolla et al., 2016] is a recently reported humanoid robot prototype which utilizes two linearly actuated tripod parallel mechanisms as legs and a cable driven parallel mechanism (CPDR) for its torso design. CARL is a compliant walking leg designed using parallel redundant actuation which mimics the behaviour of the human muscles [Schütz et al., 2017]. RH5 is a lightweight and biologically inspired humanoid robot recently developed by DFKI-RIC which uses linkages and PKM modules for most of its joints for e.g. hip flexion-extension, knee, ankle, torso and wrist [Peters et al., 2017] (see Fig. 2.1d). It weighs 64 kg and is designed for a payload capacity of 5 kg in each arm.

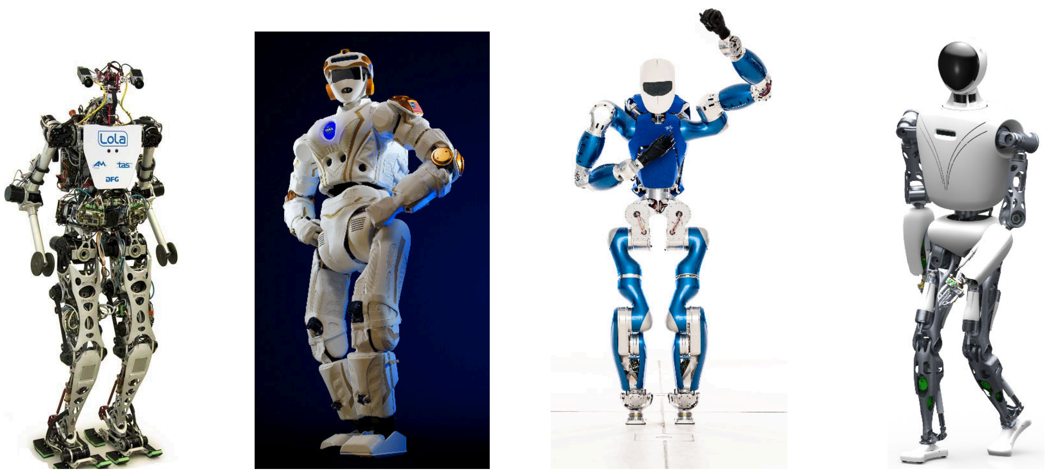

(a) Lola (TUM) [Lohmeier et al., 2006] (b) Valkyrie (NASA) [Radford et al., 2015] (c) TORO (DLR) [Englsberger et al., 2014] (d) RH5 (DFKI) [Peters et al., 2017]

Figure 2.1: Humanoid robots with series-parallel hybrid design

2.1.2 Multi-Legged Robots

Closed loop linkages and parallel mechanisms are increasingly being used in multi-legged robots which are specifically designed for high-payload applications. With their use, certain joints can be strengthened without comprimising the overall leg inertia. The multi-legged robot Mantis [Bartsch et al., 2016] depicted in Fig. 2.2a contains PKMs of type 2-SPU+1U [Serracín et al., 2012] in its ankle joints and slider-crank mechanisms that drive certain revolute joints in its legs and torso. Fig. 2.2b shows the hominid robot Charlie [Kuehn et al., 2014], featuring a Stewart platform of type 6-RUS as a six DOF active joint module in spine and neck. It also utilizes another parallel mechanism in the ankle joint. This robot supports both bipedal and quadrupedal walking gaits. SherpaTT rover, which is a wheeled-leg hybrid robot uses a closed loop linkage in inner and outer leg joints [Cordes et al., 2018], [Cordes et al., 2017]. HeritageBot [Ceccarelli et al., 2018] from University of Cassino is a hybrid robot which is capable of flying and legged locomotion on ground. Its modular design makes use of four tripod parallel mechanisms for the leg design.

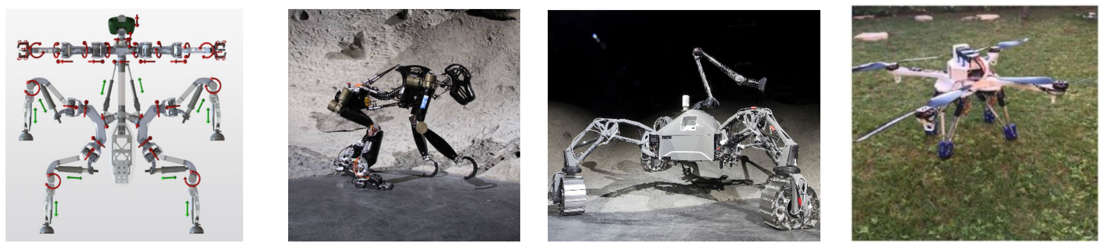
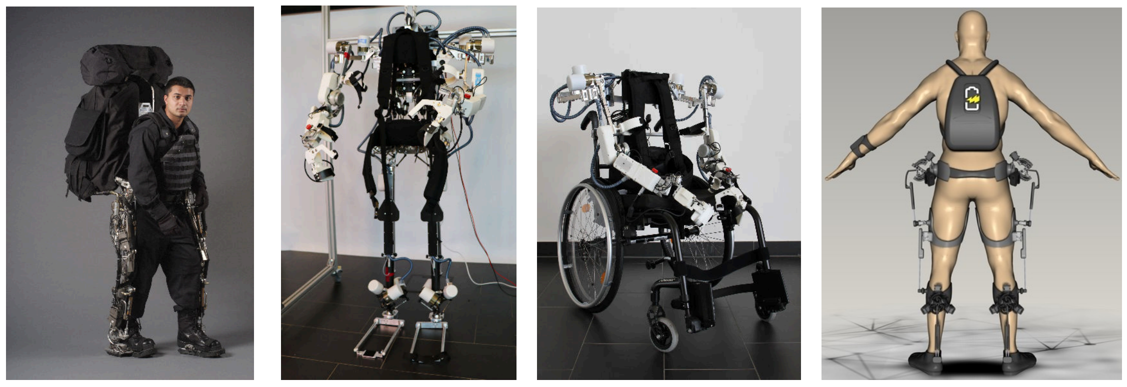

(a) Mantis (DFKI) [Bartsch et al., 2016] (b) Charlie (DFKI) [Kuehn et al., 2014] (c) SherpaTT (DFKI) [Cordes et al., 2017] (d) HeritageBot [Ceccarelli et al., 2018]

Figure 2.2: Multi-legged robots with series-parallel hybrid design

2.1.3 Exoskeletons

In exoskeletons or physical man-machine interfaces, most joints require a limited range of motion because most of the human joints like the wrist or ankle are not able to perform a 360° rotation movement. Hence, in exoskeletons based on serial kinematic chain, a physical limitation of joint movements is necessary for safety reasons. Software based joint limits may fail, hence, additional mechanical end stops are required at each joint. The use of parallel mechanisms in exoskeletons can not only lead to a lightweight design but also their limited workspace may be exploited as an additional safety feature. An early exoskeleton design utilizing closed loop linkage is Bleex [Zoss et al., 2006] as shown in Fig. 2.3a. Fig. 2.3b demonstrates a highly modular light weight Recupera full-body exoskeleton [Kirchner et al., 2016, Kumar et al., 2019b] with 32 active DOFs which is built by combining several higher DOF joint modules: a Stewart platform of type 6-UPS in torso, a double parallelogram [Kumar et al., 2017a] in shoulder for flexion–extension movement and Active Ankle mechanism [Simnofske et al., 2016] as a 3 DOF joint in hip and ankle. Due to high modularity of its mechanics and electronics design, the upper body including the two arms can be mounted on a wheelchair for rehabilitation applications (see Fig. 2.3c). A passive double parallelogram mechanism has also been employed in the design of AXO-SUIT [Bai et al., 2017] and a parallelogram mechanism has been used in Harmony dual arm exoskeleton [Kim and Deshpande, 2017]. Another recent development is the Spherical Exo Suit as shown in Fig. 2.3d which employs Agile Wrist mechanism as a 3 DOF spherical joint module at hip and ankle joints [Sadeqi et al., 2017].



(a) Berkeley Lower Extremity Exo(UCB) [Zoss et al., 2006]

(b) Recupera full body Exo (DFKI) [Credits: Meltem Y.]

(c) Recupera wheel chair Exo (DFKI) [Credits: Meltem Y.]

(d) Spherical Exo Suit (SFU, Canada) [Sadeqi et al., 2017]

Figure 2.3: Exoskeletons with series-parallel hybrid design

2.1.4 Industrial Automation

Hybrid serial-parallel robots are also gaining recognition in the industrial applications. For high stiffness applications, serial robot designs are often complemented by a parallelogram mechanism in the design (see Fig. 2.4b). Industrial robots such as ABB's IRB4400, IRB6660, KUKA's KR 40-PA, KR 50-PA, KR 700-PA robots, and Comau's Smart NJ series, SR400 utilize parallelograms for increasing stiffness of one or several joints [To and Webb, 2012], [Gautier et al., 1995]. Fig. 2.4a shows Logabex's LX4 robot which piles four Stewart platform modules in series to achieve a redundant series-parallel hybrid manipulator with large workspace and good payload (75 Kg) to weight (120 Kg) ratio. The control of such a robot is difficult [Mer, 2006]. The 3-DOF Delta robot has gained high popularity for fast pick and place operations in the industry. Soon, a demand for mounting an active wrist was realized and several such robots with 1, 2 and 3 DOF wrists were developed. A comparitive study of various series-parallel delta robot designs can be found in [Brinker et al., 2017]. FANUC M-3iA/6A Delta Robot is an interesting six axis series-parallel hybrid robot designed for pick and place operations in the food industry (see Fig. 2.4c). It is also available in 4 axis version with a single axis wrist design.

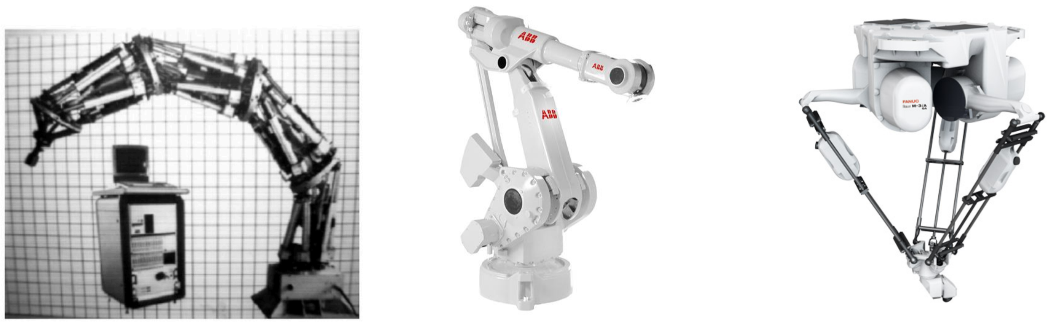

(a) Logabox LX4 [Mer, 2006] (b) ABB IRB4400 [Courtesy: ABB] (c) 6-axis DELTA robot [Courtesy: Fanuc]

Figure 2.4: Some series-parallel hybrid robots used in industrial automation

2.2 Modular and Distributive Aspects

In this section, we present the notion of modularity and distributivity in the series-parallel hybrid robots from a hardware and software perspective. The subsection Hardware is further subdivided into Mechanics and Electronics design. The subsection Software discusses the software architectures used in the control of such robotic systems.

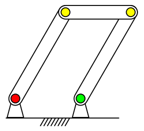
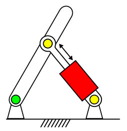
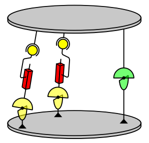
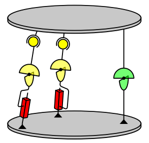
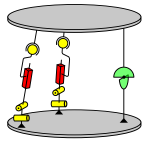
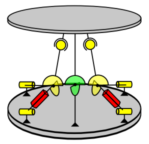
2.2.1 Hardware

Hardware of any robotic system can be broadly categorized into mechanics and electronics domain. In the following, we discuss these aspects of series-parallel hybrid robots.

2.2.1.1 Mechanics

Hybrid robots are robots that are composed of a series of serial or parallel submechanisms. Table 2.1 and Table 2.2 present a list of closed loop linkages and parallel submechanism modules which have been utilized in series-parallel hybrid robot designs discussed in Section 2.1. In all cases, these submechanisms are used as a mechanical generator of m-dimensional motion subspaces of $SE(3)$, i.e., they are used as an abstraction to either an active lower pair joint (for e.g. revolute joint, spherical

Table 2.1: Linkages and PKM modules in series-parallel hybrid robots where platform coordinates can be measured directly (red, green and yellow colors denote active, EE and other passive joints)

Mechanism			Practical Applications	
Schematic	Type	Mobility (m)	Joint Module	Hybrid Robot
	1-$\underline{\text{R}}$RRR	1 DOF rotational	Ankle pitch Series-elastic leg Ankle pitch Shoulder flexion-extension Joint 2 and/or Joint 3	TORO humanoid [Englsberger et al., 2014] ATRIAS [Hubicki et al., 2016] TALOS humanoid [Stasse et al., 2017] RECUPERA-REHA [Kumar et al., 2017a], HARMONY [Kim and Deshpande, 2017] ABB IRB4400, KUKA KR 40-PA etc.
	1-RR$\underline{\text{P}}$R	1 DOF rotational	Hip, Torso Hip flexion-extension, Knee Inner and Outer leg joints Hip-Knee, Knee-Ankle joints	MANTIS [Bartsch et al., 2016] RH5 Humanoid [Peters et al., 2017], HADE leg [Garcia et al., 2011] SHERPATT rover [Cordes et al., 2017] CARL [Nejadfard et al., 2018]
	2-S$\underline{\text{P}}$U+1U	2 DOF universal	Wrist, Torso Ankle Ankle Hip Roll-Yaw, Ankle Thurster steering	RH5 humanoid [Peters et al., 2017] MANTIS [Bartsch et al., 2016] LOLA humanoid [Lohmeier et al., 2006] THOR [Lee et al., 2014], SAFFIR [Lahr et al., 2016] AUV Leng, EurEx [Hildebrandt et al., 2013]
	2-$\underline{\text{P}}$US+1U	2 DOF universal	Wrist, Ankle and Torso	VALKYRIE humanoid [Radford et al., 2015]
	2-S$\underline{\text{P}}$RR+1U	2 DOF universal	Ankle	RH5 Humanoid [Kumar et al., 2018c]
	2-SU[1-RR$\underline{\text{P}}$R]+1U	2 DOF universal	Ankle	CHARLIE [Kuehn et al., 2014]

joint etc.) or universal joint which are building blocks of most robotic systems. It can be immediately noticed from these tables that parallel submechanism modules are mostly used as abstractions to orientational joints, exceptions are their application as six dof joint in CHARLIE, RECUPERA-REHA exoskeleton and 2R1T wrist in R1 humanoid [Parmiggiani et al., 2017]. The most popular abstractions are discussed in the following.

- 1 DOF Revolute joint: Abstraction of a revolute joint is basically done using variants of a four bar linkage. For a simple $1:1$ transmission, a parallelogram [Englsberger et al., 2014], [Stasse et al., 2017] or double-parallelogram linkage [Kumar et al., 2017a] is often employed. In comparison to direct drive joints, the main motivation here is to reduce the resulting inertia of the robot or to create a virtual center of rotation. The slider-crank mechanism (identified as 1-RR$\underline{\text{P}}$R in Table 2.1) is used to exploit the non-linear transmission properties of the mechanism. Also, the prismatic actuation gives the possibility to mount the actuator along the link longitudnally which is advantageous from a construction perspective.

- 2 DOF Universal joint: Abstraction of a universal joint is very useful when the joint requires only a limited range of rotational motion and has to be placed away from the base of the robot. Hence, the most common application is the design of ankle joint in humanoids or legged robots. They also have been used in the design of wrist and torso mechanisms. All the orientational parallel manipulators that have been used in this context are equipped with a passive leg containing the universal joint [Kuo and Dai, 2013]. Such designs are very advantageous because they do not have output singularities and provide good stiffness to the moving platform. Also, it is easy to install rotary encoders to measure the orientation of the platform directly so that forward kinematics of the mechanism is not required to be solved in real time.

- 3 DOF Spherical joint: Since the workspace of 3 DOF orientational parallel manipulators is limited, they have been mostly used in the design of exoskeletons for the abstraction of hip and ankle joints. AGILE EYE [Gosselin and Hamel, 1994] which was originally developed as a fast camera orienting device is used as hip and ankle module in SPHERICAL EXO SUIT concept. The disadvantage of this design is that it requires all the revolute joint axes to intersect at one point which is not good for high payload or impact applications. ACTIVE ANKLE [Kumar et al., 2018a] is an interesting parallel mechanism which overcomes this problem by utilizing spatial quadilateral mechanism in each leg where only push-pull forces can exist in each leg. However, it is an almost spherical mechanism and is only suited for applications where small translation of the EE point can be tolerated.

- Six DOF free joint: Many joints in humans like spine are actually very complicated and not easy to abstract using lower kinematic pairs. To provide a 6 DOF active spine to human-like robots, variants of STEWART GOUGH platform have been utilized. In this survey, one could find designs with both rotary (6-$\underline{R}$US) and prismatic actuation (6-U$\underline{P}$S). The active spine can also be used as a 6 DOF force-torque sensor and improves the workspace of limbs and lowers the velocity requirements of other limb joints [Kuehn et al., 2018]. The disadvantage of using such mechanisms is the complicated nature of their forward kinematic problem.

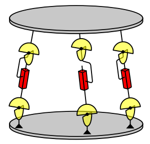
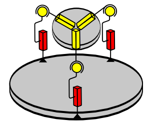
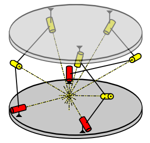
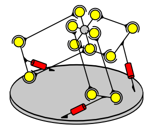
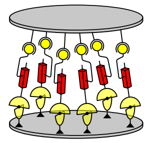
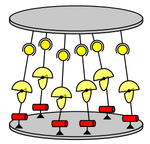
Table 2.1 shows the list of linkages and parallel submechanisms where the platform coordinates (highlighted with green) are a subset of coordinates used to describe the mechanism. Since, it is well known that it's difficult to solve the forward kinematics of the parallel robots in real time, there is a tendency to choose parallel mechanisms where additional sensors can be integrated to measure the platform coordinates. Table 2.2 shows the list of parallel submechanisms where the platform coordinates are not a subset of mechanism's coordinates. Here, its also not possible to put extra sensors to measure the platform coordinates directly but in some cases they may be integrated in other passive joints to simplify the calculation of platform coordinates from actuator states. Hence, the use of such parallel submechanisms is less common in the design of series-parallel hybrid robots in comparison to the ones presented in Table 2.1. Overall, two observations can be made from this survey: submechanism modules are used as a mechanical generator of a motion subspace (revolute, universal, spherical, free joint etc.) and the same type of submechanism with different physical parameters is utilized as a module to serve different purposes (ankle, wrist, torso joints etc.) in the same robot.

2.2.1.2 Electronics

One important development in robot electronics, that is especially helpful for parallel and hybrid robots, is the use of decentralized or distributed processing and control architectures.

In classical i.e. centralized control systems, all actuators are connected to one central host computing system (Fig. 2.5 (a)). In contrast to these, distributed processing systems (Fig. 2.5 (b, c)) consist of multiple computing devices (often called *processing elements* or *nodes*) that are connected via a network to exchange data with one another. Decentralized processing systems can be used to implement lo-

Table 2.2: PKM modules in series-parallel hybrid robots where platform coordinates can not be sensed (red and yellow colors denote active and passive joints respectively)

Mechanism			Practical Applications	
Schematic	Type	Mobility (m)	Joint Module	Hybrid Robot
	3-U$\underline{\text{P}}$U	3 DOF translational	Leg	LARMBOT [Cafolla et al., 2016], HERITAGEBOT [Ceccarelli et al., 2018]
	3-$\underline{\text{P}}$SP	3 DOF 2 Rotational 1 Translation	Torso [Fiorio et al., 2017], Wrist [Sureshbabu et al., 2017]	R1 humanoid [Parmiggiani et al., 2017]
	3-$\underline{\text{R}}$RR	3 DOF spherical	Hip, Ankle	SPHERICAL EXO SUIT [Sadeqi et al., 2017]
	3-$\underline{\text{R}}$[2-SS]	3 DOF almost spherical	Hip, Ankle	RECUPERA-REHA full-body exo [Kumar et al., 2018a]
	6-U$\underline{\text{P}}$S	6 DOF free	Torso	RECUPERA-REHA full-body exo [Kirchner et al., 2016, Kumar et al., 2019b]
	6-$\underline{\text{R}}$US	6 DOF free	Torso, Neck	CHARLIE homonid [Kuehn et al., 2014]

cal controllers that control multiple joints in a synchronized fashion (see Secs. 2.2.2 and 2.3.4), which is especially important for the control of parallel and, hence, hybrid systems.

In both centralized and distributed systems, a communication network is required to exchange data between sensors, actuators, and controllers. This is accomplished by using field-bus systems that are also often used in the industrial or automotive fields like CAN, CAN-FD or higher-level protocols like PROFIBUS [PROFIBUS & PROFINET International (PI), a]. Nowadays, different modified Ethernet-based protocols with realtime support [Gutiérrez et al., 2018] are becoming increasingly popular, for example PROFINET [PROFIBUS & PROFINET International (PI), b], EtherCAT [EtherCAT Technology Group, 2003] or SERCOS [Sercos International e. V., 2005]. Although the required bandwidth to coordinate several joints is quite low (depending on the control strategy, the exchanged data consists of simple parameters, e.g., desired position, velocity, torque), the response time should be as short as possible.

The nodes contribute to the overall processing of the system by performing specific processing tasks and providing additional capabilities, for example peripherals. Distributed architectures are important for modularity, since they provide the possibility to build independent substructures of robot systems. Local controllers implemented in the distributed hardware can be used with high sensor sampling rates and a low delay in comparison to a centralized approach.

A hierarchy with three types of modularity can be defined as follows: (1) joint-level modules, i.e., highly integrated, fully operational modules that include an actuator, sensors, and electronics for signal processing, communication and control. Commercial examples are the Robotis Dynamixel and Dynamixel Pro series [Robotis, b, Robotis, a], the Kinova KA-58 and KA-75+ actuators [Kinova, 2018], the Schunk PowerCube, PR 2, PRH, PDU 2 and PSM 2 series, TREE actuators [TREE, 2018] or the ANYdrive [ANYbotics, 2016]. However, reusable joint modules are not only used commercially, but also developed by research facilities to build novel and experimental robotic systems, such as the DLR LWR III robot [Schaeffer et al., 2007] or DFKI Charlie [Kuehn et al., 2016] joints. Further examples support often additional features, such as joint-modules with series elastic actuation as used in the WALK-MAN [Negrello et al., 2015] and RoboSimian [Hebert et al., 2015] robots, an integrated inertial measurement unit [Rader et al., 2017], or even space compliance such as the DFKI-X joint [Sonsalla et al., 2017]. However, it is important to note that many research joint modules usually do not have integrated, but closely attached electronic controllers.

Series-parallel hybrid robots that contain reusable joint modules are the CHARLIE [Kuehn et al., 2014], MANTIS [Bartsch et al., 2016] and RECUPERA [Kirchner et al., 2016] systems. In these examples one or a few actuators are combined with a processing system that is responsible for the first level control of an actuator. Joint-specific limits (position, velocity, torque) can also be implemented in the joint-level controller for a hierarchical safety implementation.

Joint level modules can be used to create systems with (2) submechanism-level modularity, i.e., more complex subsystems that need to be controlled as a whole. Examples are classical serial mechanisms like limbs or parallel mechanisms as shown in Tab. 2.1 and 2.2. Examples for such independently working modules are limbs in the ECCE humanoid robot [Jäntsch et al., 2010] or the ROBONAUT hand [Bridgwater et al., 2012] or the spine in CHARLIE [Kuehn et al., 2014]. One important point to note is that the use of these mechanisms requires to solve the (forward/inverse) kinematics and dynamics of the corresponding subsystem, which can be performed locally to maintain modularity. This constitutes the second-level control of the robotic system.

The next modularity level is (3) the complete robot system itself (which can in turn build teams or swarms with other robots). Regarding the processing architectures, it can be observed that even in robots that consist of independent modules, the distributed computing located in the modules are usually connected to a central host that is responsible for overall control. Different devices can be used to implement a distributed robotic control system, e.g., microcontrollers as in ECCE [Jäntsch et al., 2010] or FPGA/processor combinations as in the ROBONAUT [Ambrose et al., 2000]/ROBONAUT 2 [Diftler et al., 2011] and VALKYRIE [Radford et al., 2015] robots or CHARLIE [Kuehn et al., 2014], MANTIS [Bartsch et al., 2016] and RECUPERA [Kirchner et al., 2016].

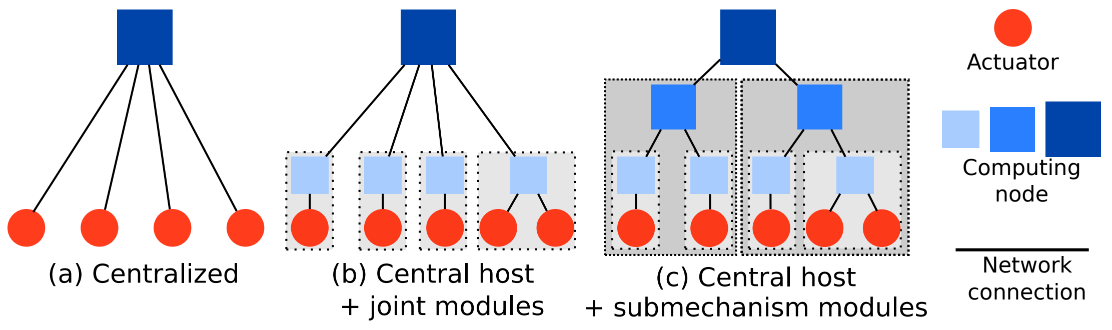

(a) Centralized
(b) Central host + joint modules
(c) Central host + submechanism modules
Actuator
Computing node
Network connection

Figure 2.5: Different types of centralized and modular processing/control architectures.

2.2.2 Software

Software is important for modular robots in especially two aspects: (1) during the design phase for simulation and design optimization, and (2) during the runtime/application phase.

It is widely acknowledged that the construction of a multi-DOF robot is a highly challenging task. Hence, to simplify this design process, reusable modules can be used. This leads to the development of tools like H-ROS [Zhang et al., 2012] and HRIM [Zamalloa et al., 2018]. Especially for modular series-parallel hybrid robots like the systems introduced in Sec. 2.1, the use of appropriate design and modelling software is important (see Sec. 2.3.1). For modular robots, frameworks like D-Rock [D-RoCK, 2018] can be used, which provide tools for a model-based robot development process. For the remainder of this section, however, we will focus on the software frameworks that are used during the runtime/application phase.

In modular robots with distributed control, it is important to enable different components to exchange data in order to synchronize the control of several joints. This is especially important for modular robots that contain PKMs (see Sec. 2.3.4). In this case, the overall operation of the robot is distributed across multiple processors to maintain modularity and, hence, several actuators have to be coordinated. Accordingly, some sort of communication middleware is required [Mohamed et al., 2009, Elkady and Sobh, 2012]. The middleware should provide a coherent interface to the different hardware components and provide mechanisms of data transfer. Many different robot middleware frameworks are available for the research community, the most popular examples probably the Robot Operating System (ROS) [Quigley et al., 2009] or Yet Another Robot Platform (YARP) [Metta et al., 2006]. Each framework has specific advantages and core application areas (e.g., perception, manipulation, locomotion for YARP, and navigation, planning for ROS [Aragão et al., 2016]).

All of these frameworks support some kind of modularity. In ROS, for example, the functionality is implemented inside so-called *nodes*. Each node runs inside a different process, the communication between nodes is mediated by an anonymous publish-subscribe middleware. However, for modular hybrid robotic systems, ROS has a number of shortcomings. ROS, for instance, has neither real time capabilities nor is it possible to use ROS on small *bare metal* systems like microcontrollers in actuator modules (see Sec. 2.2.1.2). Finally, ROS does not support fieldbus or embedded communication systems like Ethercat or even CAN directly. Hence, it is not possible to, e.g., instantiate ROS nodes that run on actuator modules to control a specific robot joint directly or even build a complete system out of them. Similar limitations are prevalent in most other robot frameworks: it is possible to build modular distributed systems on a software level, but not on a bare-metal hardware level.

Synchronization of different actuators and, hence, realtime capabilities, are of high importance for parallel and hybrid systems. A limited support of these capabilities are offered by the realtime variant of Orocos, Orocos RT [Bruyninckx, 2001], [Bruyninckx et al., 2003] and, hence, the derived Robot Construction Kit (RoCK) [Joyeux, 2010]. However, similar to ROS, these frameworks do not support low-level systems.

As a consequence, one aim of recent developments is to include bare-metal controllers and support field-busses. Following this approach will also enhance the mechanism-level modularity in series-parallel hybrid robots, since in this case multiple joints need to be controlled in combination (see Sec. 2.2.1.2). Examples where this design principle has been used are the computation of the inverse kinematics of a Stewart platform representing the spine on a softcore CPU in CHARLIE [Kuehn et al., 2014]) to satisfy constraints regarding size and power consumption which was built using NDLCom to communicate between the controllers [Zenzes et al., 2016]. Another recent example is Finroc [Reichardt et al., 2012], which has been used to, e.g., implement a distributed controller for a series-parallel hybrid leg with redundant actuation [Reichardt et al., 2017].

Nevertheless, with increasing requirements and the need to use small embedded systems, new robot software frameworks like ROS 2 [Open Source Robotics Foundation (OSRF), 2016], are being developed. They support technologies like Protocol Buffers [Google, 2017], ZeroMQ [iMatix, 2017] and the Data Distribution Service (DDS) [Pardo-Castellote, 2003, Schlesselman et al., 2004] and might satisfy the constraints of mechanism-level modularity in future applications. Although ROS 2 uses the DDS, which does not result in a performance benefit of ROS 2 over the plain TCP or UDP used in ROS [Maruyama et al., 2016], it supports realtime requirements and guarantees different Quality of Service (QoS) levels.

Another important development that has to be mentioned for completeness is the OPC Unified Architecture (UA), which is an industrial machine-to-machine communication protocol [Mahnke et al., 2009]. It not only provides mechanisms to transfer, but also semantically annotates the data. However, until now, OPC UA is mainly used in industrial robotics but rarely in research.

One important point to notice is the communication overhead that results from a decentralized approach. Clearly, if software modules are distributed across different hardware modules, the time to transfer data between central controller and leaf modules must be considered. As discussed in Sec. 2.2.1.2, many different fieldbus or custom systems [Zenzes et al., 2016, Zhang et al., 2012] are used in robotics. They can help to optimize the system regarding throughput, realtime requirements and QoS level on the physical and data link layers and are required to fulfill these requirements on the higher layers. Hence, the choice of appropriate communication hardware is essential to build distributed control systems.

2.3 Modeling and Control

Multi-body kinematics and dynamics has been an area of extensive research during the past decades. While the term kinematics encompasses problems of position, velocity and acceleration analysis, the term dynamics refers to problems associated with the study of forces and torques and their effect on motion of multi-body systems. Kinematics and dynamics essentially forms the basis of behaviour modeling and control of any robotic system. The usage of parallel submechanisms in a robot's design introduces a new level of complexity to their description, kinematics, dynamics and control. In this subsection, we discuss these domain specific difficulties and present some practical approaches used in controlling series-parallel hybrid robots.

2.3.1 Modeling

For describing serial robots, Denavit–Hartenberg (DH) parameters [Hartenberg and Scheunemann, 1955], and their modifications [Khalil and Dombre, 2002b], have become the de-facto standard: they specify each coordinate transformation by only four parameters instead of six parameters, due to the particular placement of local coordinate systems at specific locations. Since the placement of these coordinate systems requires manual effort, work has been invested to extract the DH parameters automatically from computer aided design (CAD) models of the serial manipulators [Rajeevlochana et al., 2012]. In case of tree type robots and robots with closed loops, the traditional notion of DH parameters cannot be used and hence various extensions have been proposed in the literature [Khalil and Kleinfinger, 1986]. A comparison of various robot parameterization techniques with the Sheth-Uicker parameters can, for example, be found in [Bongardt, 2013]. Due to the dependence of the frame placement on the link geometries, the modeling becomes unintuitive in particular for complex link shapes (for example in exoskeletons or human-machine interfaces). For these reasons, standard open source robot description formats, such as `URDF`[1] (`ROS`), `COLLADA`[2] (`OpenRAVE`), or `SDF`[3] (`Gazebo`), do not rely on DH parameters (or extensions) for

[1]`http://wiki.ros.org/urdf/XML`
[2]`https://www.khronos.org/collada/`
[3]`http://sdformat.org/`

representing the coordinate transforms and, instead, store the required transformations by six parameters. These coordinate transforms requested by these description formats can be automatically extracted from CAD environments by programs such as `CAD-2-SIM` [Bongardt, 2011] and `SolidWorks to URDF Exporter`[4]. Even with the presence of such tools, it can be very time consuming to create complete robot description models for series-parallel hybrid robots because most formats do not allow the possibility of a modular description. Further, `URDF` does not allow for proper definition of closed loops, that often leads to complicated work-arounds when used for description of contemporary, complex robots. Other formats, such as `SDF`, allow the definition of parallel linkages, but do not further provide the functionality to explicitly define a spanning tree of a looped graph, necessary for a standardized tree representation of a model.

2.3.2 Kinematics

Kinematics is often regarded as the study of *geometry of motion*. Direct or inverse geometric problems generally result in a set of non-linear algebraic equations regardless of the method of problem formulation. For serial robots, inverse geometric problem and for parallel robots, forward geometric problem are easy to formulate but difficult to solve. The three most useful solution techniques to deal with such problems are polynomial continuation, Gröbner bases, and elimination method [Nielsen and Roth, 1999]. J. P. Merlet in [Merlet, 1999] presented a list of open problems for parallel robots which include forward kinematics, workspace and singularity analysis, singularity free trajectory planning, optimal design etc. A rigorous kinematic analysis of generic series-parallel hybrid robots is also an open problem because they carry kinematic complexity of both serial and parallel topologies. It is still quite appropriate to quote Nielsen and Roth [Nielsen and Roth, 1999] in this context: "Yet, a lot remains to be done before the subject of kinematic position analysis in robotics can be considered closed. Large structural classes, such as hybrid series and in-parallel systems, are just beginning to be treated in a systematic manner. Mainly, such studies are still done on a case-by-case basis, without a general theory and framework." An example of such an analysis of 3-RPS-3-SPR type series-parallel hybrid manipulator can be found in [Nayak et al., 2018].

Nevertheless, a Jacobian based numerical solution to the geometric problems in singularity free regions is usually possible, which forms the basis of several multibody simulation software. Also, developments in the field of modern kinematics [McCarthy, 2013] such as screw theory and computational algebraic geometry has helped researchers a great deal to study specific families of parallel or series-parallel

[4]`http://wiki.ros.org/sw_urdf_exporter`

hybrid mechanisms. Also, it is important to note that comprehensive kinematic analysis of all the mechanisms listed in Table 2.1 and Table 2.2 is readily available in the literature.

2.3.3 Dynamics

Notable works in the field of robot dynamics include Newton–Euler [Featherstone, 2008], [Khalil and Dombre, 2002b], and Lagrangian [Khalil and Dombre, 2002b] formulations, the Decoupled Natural Orthogonal Compliment (DeNOC) formulation [Rao et al., 2006], and Kane's method [Lesser, 1992],[Buffinton, 2005]. Traditionally, the equations of motion were described using 3D vectors – which quickly yields a large amount of equations for systems of connected bodies [Luh JS, 1980]. To address this issue, alternative compact and user-friendly formulations have been developed based on screw theory [Featherstone, 2008],[Jain, 2011b] and Lie group theory [Müller, 2017],[Müller, 2018] which can easily be transformed into program code for modern computers.

Robots containing closed loops are subjected to additional geometric loop closure constraints which are difficult to resolve at position level and hence most multi-body dynamics software frameworks try to resolve them at velocity and acceleration levels and arrive at position constraints numerically. The Rigid Body Dynamics Library (`RBDL`) [Felis, 2017] and `OpenSim` [Delp et al., 2007] are some examples of open source libraries that implement such algorithms and import robot descriptions using various robot description formats. Table 2.3 presents a survey on multi-body software that can deal with kinematic and dynamic analysis of robots with closed loops. Another issue with series-parallel robots is the considerably large number of their spanning tree DOFs. Let n be the number of DOFs of the spanning tree representing a series-parallel hybrid robot, m be the total mobility of the robot, p be the number of actuators in the system and c be the number of independent closed loops. RH5 humanoid which only contains relatively simple parallel mechanism modules (with less than 3 DOF) has 32 DOF ($m = p = 32$), $c = 15$ independent closed loops and $n = 76$ DOF in its spanning tree. For a complicated hybrid robot such as Recupera-Reha exoskeleton, the robot has $m = p = 30$, $c = 29$ and $n = 128$. Hence, it can be very challenging to solve the full dynamic model of such systems in real time. In most practical applications reported in literature, full dynamic model of the series-parallel hybrid robots is not exploited [Hopkins et al., 2015], [Schütz et al., 2017], [Nejadfard et al., 2018].

Library	**Developer / Institution**	**Language / Platform**	**Licensing**	**Supported Mechanisms**	**Robot Description**	**Robot Parameterization**	**Algorithm / Approach**
ADAMS	MSC ADAMS	Cross	Commercial	Everything	CAD	Proprietary	Proprietary
CATIA / DELMIA	Dassault Systems	Windows	Commercial	Everything	CAD	Proprietary	Proprietary
GEAR	Junggon Kim, RI CMU	C++/Cross	Open Source	Serial, Tree type, Closed loop	C++ file	Featherstone	RNEA
Neweul-m2	Univ of Stuttgart	MATLAB/Windows	Partners only	Serial, parallel, tree	.m file	Proprietary	NE
OpenSYMORO	Prof. Khalil, IRCCYN	Python / Linux	Open Source	Serial, Tree type, Closed loop	PAR file, MDH input in GUI	Khalil-Kleinfinger	NE, Lagrangian
Pinnochio	Nicolas Mansard, LAAS-CNRS	C++/Python / Linux	Open Source	Serial, Tree type	URDF/Lua	Featherstone	Featherstone
PyMbs	TU Dresden	Python/Linux	Open Source	Serial, certain closed loops	Python script	Proprietary	NE
RBDL	M Felis, Univ. of Heidelberg	C++/Linux	Open Source	Serial, Tree type	URDF/Lua	Featherstone	RNEA
OpenSIM	NCSRR, Stanford University	C++/Linux	Open Source	Everything	URDF	Rodriguez and Jain	RNEA
REDYSIM	Mechatronics Lab, IIT Delhi	MATLAB / Windows	Protected Freeware	Serial, Tree type, Closed loop	.m file	Khalil-Kleinfinger	DeNOC
RobCoGen	IIT, Genoa	C++/Linux	Open Source	Serial, Tree type	Kinematics-DSL	Featherstone	Featherstone
Simmechanics	MathWorks	MATLAB / Cross	Commercial	Everything	CAD	Proprietary	Proprietary
V-Rep	Coppelia Robotics	C++/Cross	Dual Licensing	Serial, Tree type, Parallel	CAD	Proprietary	Vortex/NE

Table 2.3: Comparision of different robot modeling tools

2.3.4 Control

As previously stated, parallel robots can provide higher stiffness, speed and accuracy as their serial counterparts. However, oftentimes those theoretical capabilities are not easy to achieve in practical applications due to different error sources. While higher speeds have been vastly shown in parallel systems and high accuracies in many of them, a broad sense of higher accuracy and stiffness control capabilities are more difficult to be generically achieved in practical applications [Paccot et al., 2009, Abdellatif and Heimann, 2010]. One reason is that accuracy is affected by different sources: on the one side, parallel systems might require more joints than a serial one for the same task, and thus the kinematics errors due to assembly inaccuracies or simplified kinematics might negatively affect the final system's accuracy. On the other side, the stiffness control is more complex since it requires of a workspace analysis to locate optimal areas for higher stiffness (some areas are of very low stiffness) [Merlet, 2000]. Both accuracy and stiffness control are additionally affected by the larger amount of singularities present in parallel robots and which need to be removed, basically by highly-redundant actuation [Cheng et al., 2003][Liu et al., 2001][Muller, 2005]. Fortunately, many of those issues can be solved by using more complete and efficient real-time kinematics and dynamics models (instead of simplified ones), and by proper experimental identification of the parameters of those models (to deal, for instance, with unmodelled inaccuracies). Having achieved that by using more accurate real-time models, hybrid systems can deliver high performances in a broader spectrum than those of only serial or parallel nature.

Notwithstanding, more complex control strategies can also significantly contribute to having better dynamic performance, i.e. closer performance to the expected theoretical one. However, historically the same classical control strategies used in serial robots have been reused for parallel robots and there is relatively few specific literature on control of parallel devices [Ghorbel et al., 2000] in comparison with their serial counterparts. The approaches followed basically fall into two categories: (a) the model-free control schemes such as PID control [Khosravi and Taghirad, 2014, Su et al., 2004], fuzzy control [Zi et al., 2008], use neural networks to learn dynamics without a priori knowledge of the system [Mirza et al., 2017] or based on force feedback such as in the seminal work in [Merlet, 1988], and (b) model-based control schemes such optimal control [Liu et al., 2001] or the use of machine learning methods [Abdellatif and Heimann, 2010]. The use of same control methods as in serial robots leads to some limitations inherent to the nature of those methods. For instance, in serial robots it is a de facto standard to use joint space control, which is not the most suitable strategy for parallel robots and, consequently, not for hybrid systems. A parallel robot is described by its end-effector pose rather than by joint configuration as in serial systems. For that reason, Cartesian (or task) space control is more suitable for complex hybrid mechanisms, especially multipurpose systems which use redundancy to perform a number of tasks simultaneously. However, some task controllers will likely need information from the position in abstract joint space (ref. to Fig 2.6), which for parallel systems is much more complex problem, usually solved numerically. The fact that forward kinematics is often not of closed form means that a certain joint configuration can lead to different end-effector poses.

Additionally, in rigid serial robots the assumption is that each actuator (abstract joint as in Fig. 2.6) can be considered almost dynamically decoupled from the others ones (especially a good assumption in actuators with high gear ratios) and thus, typically a linear PID controller will be used at each axis. The assumption does not hold true for lightweight serial robots moving at high speeds because the nonlinear dynamics cannot be properly compensated. Parallel robots would suffer the same issues, as they are highly nonlinear and highly-coupled systems by nature. Fortunately, model-based control approaches such as computed-torque control [Shang et al., 2012] can alleviate some of those problems and compensate nonlinear dynamics, provided, once more, that there is an accurate robot model. Some other techniques such as robust control [Shang and Cong, 2014] can also be used with simplified dynamics or any kind of nonlinear controllers [Su et al., 2004][Shang et al., 2010]. In any case, for parallel systems the optimal strategy would be to work in Cartesian space, although that would mean to face the problem of directly measuring the end-effector pose, since using numerical estimation and forward kinematics to compute the end-effector pose could degrade again the dynamic performance of the robot.

2.3.5 Adopted Practices

Series-parallel hybrid robots are highly complex mechatronic systems and generic treatment of such robots remains an open problem. Hence, there is always a trade-off between modeling depth, accuracy and computational efficiency. However, modularity in robot design allows for certain abstractions which simplifies their modeling and control. Such abstractions are shown in Fig. 2.6. While Fig. 2.6(a) captures the true complexity of the robot, due to absence of generic methods to model and control such systems, three different abstractions are adopted to simplify the modeling and control. In the following, we discuss the practices used in design, modeling and control of series-parallel hybrid robots.

- **Design**: It is common practice to avoid any switch of assembly mode [5] in the design of parallel submechanism modules for hybrid robots. It is

[5] Assembly modes are different solutions to forward kinematics problem.

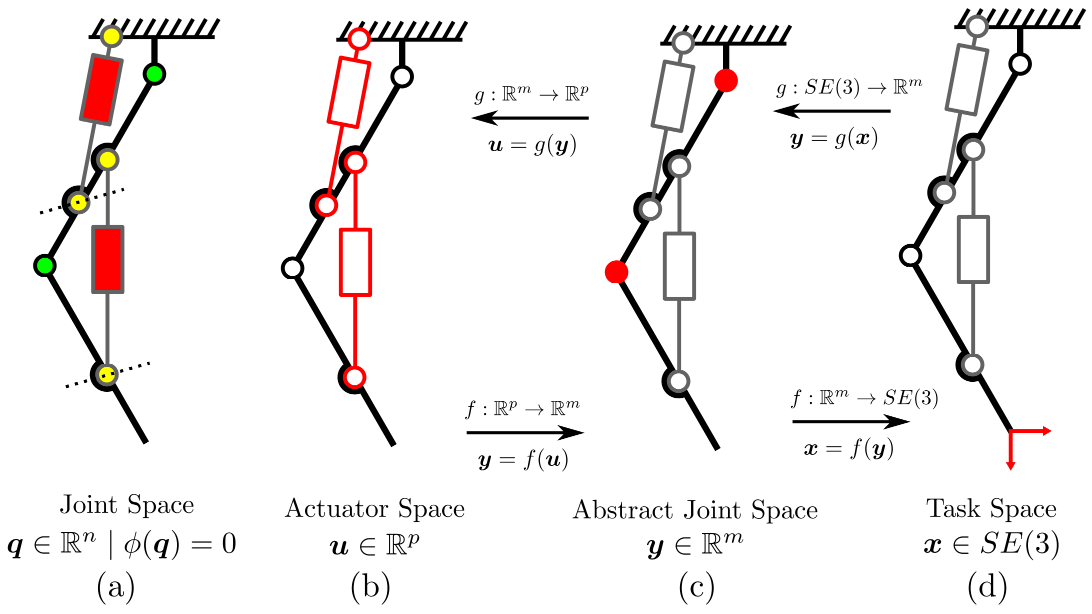

$g : \mathbb{R}^m \rightarrow \mathbb{R}^p$
$\boldsymbol{u} = g(\boldsymbol{y})$
$g : SE(3) \rightarrow \mathbb{R}^m$
$\boldsymbol{y} = g(\boldsymbol{x})$
$f : \mathbb{R}^p \rightarrow \mathbb{R}^m$
$\boldsymbol{y} = f(\boldsymbol{u})$
$f : \mathbb{R}^m \rightarrow SE(3)$
$\boldsymbol{x} = f(\boldsymbol{y})$
Joint Space
$\boldsymbol{q} \in \mathbb{R}^n \mid \phi(\boldsymbol{q}) = 0$
(a)
Actuator Space
$\boldsymbol{u} \in \mathbb{R}^p$
(b)
Abstract Joint Space
$\boldsymbol{y} \in \mathbb{R}^m$
(c)
Task Space
$\boldsymbol{x} \in SE(3)$
(d)

Figure 2.6: Different abstractions used for modeling and control of hybrid robots

achieved by choosing appropriate design parameters and physically restricting the movement of the joints in the parallel submechanism module. This ensures a unique forward kinematics solution for any given actuator input which makes the behaviour of submechanism modules similar to serially connected joints and greatly simplifies the modeling and control of such systems. However, it comes at a cost of workspace restriction as certain kind of singularities can be crossed using appropriate trajectory planning in case of parallel robots [Pagis et al., 2015].

- **Kinematics**: Forward and inverse kinematics of the submechanism module is usually solved to provide a bi-directional map between actuation space and abstract joint space (see Fig. 2.6(b) & (c)). Forward and inverse kinematics of parallel submechanism modules can be solved on local controllers either analytically or in resource-constrained systems with the help of Look Up Tables (LUTs). Analytical solutions are preferred when embedded hardware includes a microcontroller with a Floating Point Unit (FPU) (e.g. parallel joints in THOR [Hopkins et al., 2015]) or in cases when parallel submechanism modules bear more than two DOF (e.g. 6 DOF spine joint in Charlie [Kuehn et al., 2014]). As an alternative, LUTs can be used for systems without FPUs or FPGA-based local controllers (e.g. 1 or 2 DOF parallel joints in MANTIS [Bartsch et al., 2016] and 2 DOF ankle in Charlie [Kuehn et al., 2014]). Once a mapping is available, the robot can be treated purely as a serial or tree type structure for which forward and inverse kinematics problems are easy to solve on the main controller (see Fig. 2.6(d)). Many series-parallel hybrid robots such as SherpaTT, MANTIS and Charlie are kinematically controlled and compliance is realized only with the help of force/torque measurements. Further, it is not common practice to compute the full kinematic state of the spanning tree (see Fig. 2.6(a)) since such calculations can be computationally expensive.

- **Dynamics**: As pointed out before, the computation of full inverse dynamic model for hybrid robots can be computationally very expensive due to the large size of their spanning trees and the large number of loop closure constraints to be resolved. The moving parts inside a parallel submechanism module may have relatively small contribution to the overall dynamics of the system which is essentially due to dynamics of link segments, and joint friction etc [Buschmann et al., 2013]. Hence, an inverse dynamic model in abstract joint space is often combined with an inverse static model in actuation space to compute the actuator forces [Hopkins et al., 2015], [Vonwirth, 2017]. This approach is used in torque controlled series-parallel hybrid humanoids such as THOR, Valkyrie, Lola etc. To the best knowledge of the author, the tradeoff between the complete dynamic model and simplified dynamic model has not been reported in the literature.

2.4 Conclusion

This chapter presents the state of the art in design and control of series-parallel hybrid robots. Despite their kinematic complexity, such designs are becoming increasingly popular due to the mechanical advantage. Overall, one could conclude that by adopting certain practices in design, modeling and control, it is possible to use such designs in various robotics applications. Modularity in kinematics and dynamics algorithms and their distributed implementation can make it easy to deal with high complexity of series-parallel hybrid robots. However, there is a lack of general framework for analysing and modeling such systems. This is addressed in the later chapters of this thesis.

Part II

Geometric Analysis

Chapter 3

Modern Approaches in Geometric Analysis

This chapter builds the fabric of the Geometric Analysis part of the thesis. It starts with a small description on the historical background in the study of mechanisms (based on [Selig, 2005]) and then lays down the fundamental problems that one comes across in the geometric analyses of the robotic mechanisms (Section 3.2). Subsequently, two modern approaches namely screw theory (Section 3.4) and algebraic geometry (Section 3.3) are introduced and how they lead to the local and global analyses of these systems are described. Further, the advantages and disadvantages of these methods are highlighted. Lastly, a simple example of a slider crank mechanism is provided in Section 3.5 and these approaches are applied in its study.

3.1 Historical Interplay between Mathematics and Robotics

Modern industrial manipulators have a history of a little more than half-century but their ancestors which are mechanisms and linkages have existed for millennia. The human ability to lift heavy loads with the help of cranes or other lifting devices has been a key ingredient in the advancement of human civilization. These mechanisms consist of rigid links and are connected by various kinematic joints to transfer the motion and forces from one part to another. A geometric intuition is required to build such devices and hence it has caught the attention of various mathematicians, in particular, geometers in the last few centuries. In 1875, Alfred Bray Kempe, an amateur British mathematician, proved that all algebraic curves can be traced by linkages [Kempe, 1875]. Later, Koenigs proved a similar theorem for curves in space[1]. At around this time, many mathematicians were already interested in the theory of mechanisms – Chebychev, Schönflies and Darboux, to name a few. The most intensively studied mechanism has been the four bar linkage till this date. This device is ubiquitous in mechanical engineering, as it is an extraordinary design element. Some questions concerning the geometrical capabilities of this mechanism still remain unanswered [Selig, 2005].

Mathematicians were also interested in more general problems. In 1829, French mathematician Évariste Galois laid the foundation of group theory by studying the roots of a polynomial and characterizing the polynomial equations that are solvable by radicals in terms of properties of the permutation group of their roots. For example, he was able to explain why it is not possible to solve quintic polynomials with simple algebraic operations (addition, subtraction, multiplication and division) and application of radicals (square roots, cube roots, etc). Felix Klein, a German mathematician, developed his Erlangen program in 1872 which cemented the relationship between geometry and group theoretic ideas. Almost simultaneously, a Norwegian mathematician named Sophus Lie, developed the theory of continuous groups. In 1884, Clifford developed geometric algebras, modeled on Hamilton's quaternions; he was able to include translations as well as rotations. Around the turn of the 20th century, Ball developed Screw theory, which dealt with infinitesimal rigid body motions and was mainly used to look at problems concerning statics and dynamics [Ball, 1876]. A little later, Eduard Study looked at the geometry of the set of all finite rigid body motions. His work relied heavily on Clifford's dual quaternions.

After the first world war, mathematicians seem to have turned away from the study of mechanisms. Mechanical engineers also neglected mechanisms during this period. The first digitally operated and programmable robot, called Unimate, was invented by George Devol in 1954 and laid the foundations of the modern robotics industry. The first industrial robot was installed by General Motors in 1961. In 1968, Kawasaki bought a licence from Unimation to manufacture robots in Japan. In 1979, SCARA (selective compliance assembly robot arm) was introduced in Japan for assembly of printed circuit boards. All these developments caused a renewed interest of mechanisms community in the study and design of mechanisms and robots. In that vein, K.H. Hunt mastered screw theory for the analysis of spatial mechanisms [Davidson and Hunt, 2004] and Roger Brockett introduced the product of exponentials formula using matrix exponential mapping which gives the connection between a matrix Lie algebra $se(3)$ and the corresponding Lie group $SE(3)$ [Brockett, 1984] (see Appendix A and Section B.3 of Appendix B for an abstract understanding of Lie groups and Lie algebra). Due to the advances in the field

[1]A modern treatment can be found in [Li et al., 2018]

of numerical algebraic geometry and the advent of powerful computational algebra software, the algebraic description of the constraint equations became equally viable and their solution provided some deep insights into the global kinematic behavior of the mechanisms (for e.g. 6R serial chain and Stewart Gough platform). Moreover, electrical engineers became interested in the problem of controlling robots and computer scientists saw robots as vehicles for testing their ideas on artificial intelligence.

Loosely speaking, robots and mechanisms differ in the sense that the former are designed as general purpose machines with many degrees of freedom and the latter are designed for particular functionality and hence only a few, normally just one degree of freedom. From a mathematical perspective, the analyses of mechanisms and robots are the same to all intents and purposes. In the following section, we discuss the key terms and fundamental problems that one comes across in the geometric analyses of the robotic mechanisms. Later, we look at these problems through the eyes of screw theory and algebraic geometry and highlight how these approaches from modern kinematics help in a better understanding of those problems.

3.2 Problem Description

A robot is mechanically constructed using a set of links and joints. These joints constrain the kinematic motion between these links. The robot might be subjected to additional constraints from its environment or high level task specification. Thus, to analyze a robot one must study how different rigid bodies behave under a set of kinematic constraints. Fig. 3.1 provides a classification of kinematic constraints that typically occur in robotics. The first distinction is made between equality and inequality constraints. The former arise from a permanent physical contact between two bodies and the latter arise from situations where two bodies are allowed to make a contact and separate. Example of inequality constraints include phenomena such as collision, bouncing and loss of contact. The equality constraints (also called as Pfannian constraints) could be further divided into holonomic and non-holonomic constraints. The former are the constraints on the position variables and typically arise from sliding contact while the latter are constraints on the velocity variables and typically occur in rolling contact. At velocity and acceleration levels, these constraints are the same but the non-holonomic constraints can not be integrated i.e. they are not a derivative of any function. An immediate consequence of this is that a system subjected to non-holonomic constraints has more positional degrees of freedom than velocity degrees of freedom. The final distinction is done between scleronomic constraint, which is a function of position variables only, and rheonomic constraints which is also a function of time. In the scope of this thesis, we focus on the analysis of series-parallel hybrid robots, and hence, will restrict our attention to scleronomic constraints.

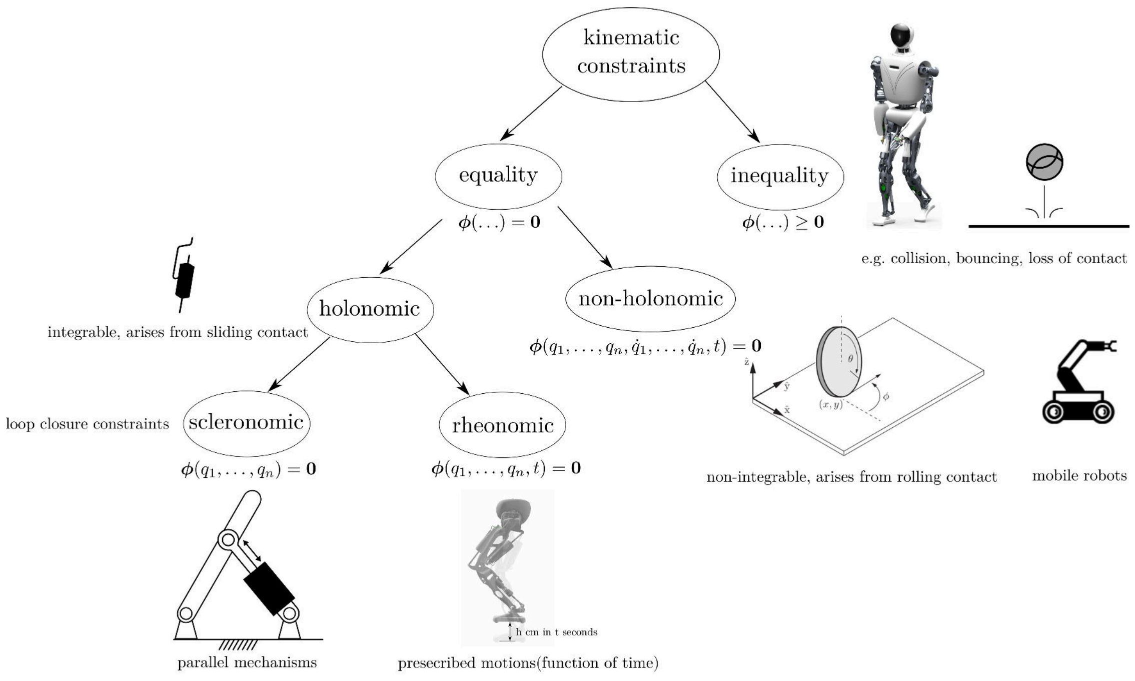

kinematic constraints
equality
$\phi(\ldots) = \mathbf{0}$
inequality
$\phi(\ldots) \geq \mathbf{0}$
e.g. collision, bouncing, loss of contact
holonomic
integrable, arises from sliding contact
non-holonomic
$\phi(q_1, \ldots, q_n, \dot{q}_1, \ldots, \dot{q}_n, t) = \mathbf{0}$
loop closure constraints
scleronomic
$\phi(q_1, \ldots, q_n) = \mathbf{0}$
rheonomic
$\phi(q_1, \ldots, q_n, t) = \mathbf{0}$
non-integrable, arises from rolling contact
mobile robots
parallel mechanisms
h cm in t seconds
prescribed motions(function of time)

Figure 3.1: Classification of kinematic constraints in multi-body systems based on [Featherstone, 2008]

Perhaps, one of the most fundamental questions one may ask is, where is the robot? The answer is *completely* given by the robot's configuration which is defined in the following.

Definition 5 (Configuration & c-space) *The configuration q of a robot is complete specification of position of every point on the robot. The number of real-valued coordinates n required to represent the configuration is the degrees of freedom of a robot. The space containing all the possible configurations of the robot is called configuration space (c-space) denoted by $\mathcal{Q}$ [Lynch and Park, 2017].*

Since, the configuration space contains the full description of the mechanism, understanding its shape can provide deep insights into the geometric behaviour of a robot. The shape of the c-space is described using the c-space topology which is a fundamental property of space itself and is independent of the choice of coordinates in the space. For example, the c-space of a single DOF prismatic joint is isomorphic to $\mathbb{R}^1$. Similarly, the c-space of a single DOF revolute joint is a unit circle $\mathbb{S}^1$ as the self-connectedness of the rotational movement can not be captured by $\mathbb{R}^1$. The c-space of a serial robotic system can usually be obtained by taking a cartesian product of c-spaces of individual joints. For example, the c-spaces of a rotating sliding knob and a 2R planar robot arm are a cylinder $\mathbb{R}^1 \times \mathbb{S}^1$ and a two dimensional torus $\mathbb{T}^2 = \mathbb{S}^1 \times \mathbb{S}^1$ respectively. More examples can be found in Fig. 3.2. The c-space of a n DOF robot with n_r revolute joints and n_p prismatic joints can be expressed as $\mathbb{V}^n = \mathbb{T}^{n_r} \times \mathbb{R}^{n_p}$. The configuration of a parallel robot is admissible if and only if it satisfies the geometric loop closure constraints. Let us define an implicit loop closure function $\phi : \mathbb{V}^n \mapsto \mathbb{R}^r \mid \phi(\boldsymbol{q}) = \boldsymbol{0}$ where r is the number of independent loop closure constraints acting on the system. Then the configuration space of the parallel robot, being the set of all admissible configurations, is

$$\begin{aligned} V &:= \{\boldsymbol{q} \in \mathbb{V}^n \mid \phi(\boldsymbol{q}) = \boldsymbol{0}\} \\ V &= \phi^{-1}(\boldsymbol{0}) \end{aligned} \tag{3.1}$$

The c-space V is a real variety in $\mathbb{V}^n$ and locally (close to a regular configuration $\boldsymbol{q}$)

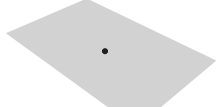
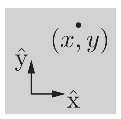
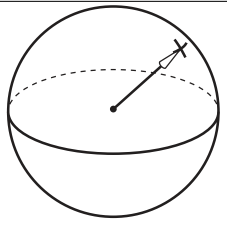
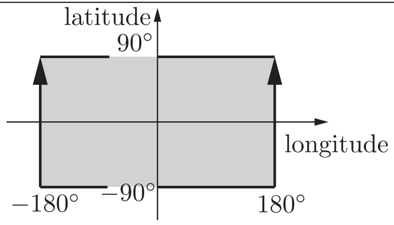
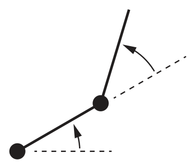
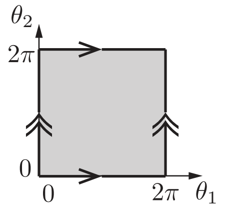
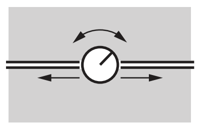
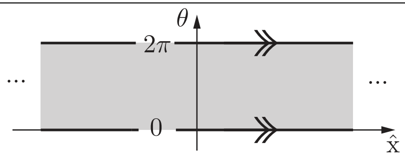
system	topology	sample representation
 point on a plane	$\mathbb{E}^2$	 $\mathbb{R}^2$
 spherical pendulum	S^2	 $[-180°, 180°) \times [-90°, 90°]$
 2R robot arm	$T^2 = S^1 \times S^1$	 $[0, 2\pi) \times [0, 2\pi)$
 rotating sliding knob	$\mathbb{E}^1 \times S^1$	 $\mathbb{R}^1 \times [0, 2\pi)$

Figure 3.2: Four topologically different c-spaces and example coordinate representations [Lynch and Park, 2017]

a smooth manifold [Müller, 2013]. V comprises several connected smooth manifolds (subspaces like smooth curves or surfaces) that are separated by singular points, indicating *non-smoothness* of V at these points. The mobility of a parallel robot hence depends on the c-space topology. The local DOF of the robot is given by the local dimension of the variety $\dim_q V$. The general mobility of a parallel robot is given by $m = n - r$ and can be estimated by Kutzbach-Gruebler's criteria:

$$d_s(\mathcal{M}) = s(n - m - 1) + f = s(-c) + f \ , \tag{3.2}$$

where

- s – motion parameter ($= 3$ for planar and spherical mechanisms, $= 6$ for spatial mechanisms)
- n – Number of links in the mechanism
- m – Total number of joints
- c – Number of independent closed loops
- f – sum of DOF of each joint.

Practically, one is often interested in only knowing the position of the robot's end-effector and whether it is able to perform the required task. The output motion produced by a robot's end-effector can be described in subspace of $SE(3)$ and can be parameterized by appropriate choice of coordinates. For e.g. the end effector configuration of a spatial 6R serial manipulator can be described by a homogenous transformation matrix $\boldsymbol{P}_E \in SE(3)$. The output mapping $f_O : \mathbb{V}^n \mapsto SE(3)$ yields the end effector configuration $\boldsymbol{P}_E = f_O(\boldsymbol{q})$ as a function of joint space configuration $\boldsymbol{q}$.

Definition 6 (Workspace) *The workspace $\mathcal{W}$ is a set of all the configurations that the end-effector can reach.*

$$\mathcal{W} := \{f_O(\boldsymbol{q}) \forall \boldsymbol{q} \in V\} \subset SE(3) \tag{3.3}$$

The workspace is usually determined by the robot's structure and chosen end-effector but independently of the task.

Definition 7 (Task space) *The task space $\mathcal{T} \subset SE(3)$ is a space in which the robot's task can be naturally expressed. The task space is defined by the nature of the task independently of the robot.*

The robot is called task-redundant if $\dim\mathcal{T} < \dim\mathcal{W}$ and $\mathcal{T} \subseteq \mathcal{W}$, and task-deficient if $\mathcal{W} \subset \mathcal{T}$. It must be noted that the task space, the robot's workspace and its c-space are different from each other. A point in the task space might not be feasible in the workspace of the robot. When feasible in the workspace, it may correspond to more than one robot configuration meaning that the point is not a complete specification of robot's configuration.

The robot's motion is determined by the motion of its actuators - the mechanical input. The relation between the actuator input and robot's motion is expressed by an input mapping $f_I : \mathbb{V}^n \mapsto \mathcal{U}$ that assigns any feasible robot configuration to the admissible input. This relation may not be unique as there may be different inputs corresponding to the same configuration of the robot. If there are p actuators in the robot, $\mathcal{U}$ is p-dimensional.

Definition 8 (Actuation space) *The actuation space $\mathcal{U}$ is the set of all admissible actuator configurations in a robotic system. The actuation space is also dependent on robot's structure.*

$$\mathcal{U} := \{f_I(\boldsymbol{q})\forall \boldsymbol{q} \in V\} \subset \mathbb{V}^n \tag{3.4}$$

For a fixed base serial mechanism, the actuation space is the same as configuration space i.e. $p = m = n$. However, in a parallel architecture, the actuation space is only a subspace of $\mathbb{V}^n$ i.e. $p < n$. The robot is said to be fully actuated when $p = m$, redundantly actuated when $p > m$.

$$\begin{tikzcd}
& \overset{\text{Configuration space}}{\mathcal{Q}} \arrow[dl, "f_I"'] \arrow[dr, "f_O"] & \\
\underset{\text{Actuation space}}{\mathcal{U}} \arrow[rr, shift left, "f_{FK}"] & & \underset{\text{Workspace}}{\mathcal{W}} \arrow[ll, shift left, "f_{IK}"]
\end{tikzcd}$$

Figure 3.3: Scheme of different mappings and spaces in robotics

Fig. 3.3 shows the different mappings and spaces that we commonly encounter in robotics. It is clear that c-space is the central object geometrically representing the robot. The input and output mappings yields the input i.e. actuator configuration and output i.e. end effector's configuration, respectively, corresponding to a given robot configuration. These mappings are not one to one for parallel robot in general and for redundant parallel mechanisms in particular. The notion of different spaces, input and output mappings above provides us a good abstract understanding of robot's motion. However, in practice, it is easier to work with direct mappings between the input and output spaces hiding the complete internal state of the robot. In the following, we introduce the notion of direct and inverse kinematic mappings.

Definition 9 (Direct Kinematic Mapping) *The direct kinematic mapping yields the end-effector configuration $\boldsymbol{P}_E \in \mathcal{W}$ from the actuator configuration $\boldsymbol{u} \in \mathcal{U}$ of the robot.*

$$\boldsymbol{P}_E = f_{FK}(\boldsymbol{u}) \tag{3.5}$$

Generally, it is very straight-forward to solve this problem for a serial robot and the mapping is one to one in nature. For parallel architectures, this problem is very difficult to solve and often many solutions to the forward kinematic problem exist corresponding to an actuator configuration.

Definition 10 (Inverse Kinematic Mapping) *The inverse kinematic mapping yields the actuator configuration $u \in \mathcal{U}$ from the end-effector configuration $\boldsymbol{P}_E \in \mathcal{W}$ of the robot.*

$$\boldsymbol{u} = f_{IK}(\boldsymbol{P}_E) \tag{3.6}$$

On the contrary, it is easy to solve the inverse kinematic problem for a parallel robot and difficult to solve it in case of serial architectures.

There are essentially two approaches that differ in the way the kinematics is modeled: one uses joint angles and displacements, and the other uses an algebraic parameterization of the motion of the links, such as dual quaternions or Study parameters. In the first case, the c-space V is an *analytic* variety, and in the latter it is an *algebraic* variety which are defined as the following:

- **Analytic Variety** is defined locally as the set of common zeros of finitely many analytic functions. An analytic function is a function that is locally given by a convergent power series.
- **Algebraic Variety** is defined as the set of solutions of a system of polynomial equations. A polynomial is an expression consisting of variables and coefficients, that involves only the operations of addition, subtraction, multiplication, and non-negative integer exponents of variables.

In the next two sections, we will discuss the general tools for algebraic (Section 3.3) and analytic (Section 3.4) formulations and an example demonstrating the use of the two methods is provided later in Section 3.5.

3.3 Algebraic Geometry

The configuration space of a large class of mechanisms can be defined using polynomial equations. At the core of it, this is possible due to two main reasons: 1) rigid body transformations are algebraic, 2) all lower pair joints except for helical joint are algebraic in nature [Wampler and Sommese, 2013]. An introduction to algebraic geometry can be found in the Appendix C. Two important tools for formulating algebraic constraint equations for any mechanism are 1) Study's kinematic mapping and its variants, 2) tangent half angle substitution which are described in the following.

3.3.1 Study's Kinematic Mapping

Study's Kinematic Mapping maps every displacement in $SE(3)$ to a point in a 7-dimensional projective space $\mathbb{P}^7$ [Husty and Schröcker, 2013, Husty et al., 2007]. If the homogeneous coordinate vector of $\boldsymbol{x}$ is $[x_0 : x_1 : x_2 : x_3 : y_0 : y_1 : y_2 : y_3]^T$, the kinematic pre-image of $\boldsymbol{x}$ is the displacement $\boldsymbol{T} \in SE(3)$ described by the transformation matrix:

$$
\begin{aligned}
\boldsymbol{T} = \frac{1}{\Delta}\begin{bmatrix} \Delta & 0 & 0 & 0 \\ p & {x_0}^2+{x_1}^2-{x_2}^2-{x_3}^2 & -2\,x_0x_3+2\,x_1x_2 & 2\,x_0x_2+2\,x_1x_3 \\ q & 2\,x_0x_3+2\,x_1x_2 & {x_0}^2-{x_1}^2+{x_2}^2-{x_3}^2 & -2\,x_0x_1+2\,x_3x_2 \\ r & -2\,x_0x_2+2\,x_1x_3 & 2\,x_0x_1+2\,x_3x_2 & {x_0}^2-{x_1}^2-{x_2}^2+{x_3}^2 \end{bmatrix} \\
\Delta = {x_0}^2+{x_1}^2+{x_2}^2+{x_3}^2 \\
p = -2\,x_0y_1+2\,x_1y_0-2\,x_2y_3+2\,x_3y_2 \\
q = -2\,x_0y_2+2\,x_1y_3+2\,x_2y_0-2\,x_3y_1 \\
r = -2\,x_0y_3-2\,x_1y_2+2\,x_2y_1+2\,x_3y_0\;.
\end{aligned} \tag{3.7}
$$

The parameters $x_i,\ y_i,\ i \in \{0,...,3\}$ are called as the *Study's parameters*. An Euclidean transformation can be represented by a point $p \in \mathbb{P}^7$ if and only if the following equations are satisfied:

$$g_1 := x_0y_0 + x_1y_1 + x_2y_2 + x_3y_3 = 0 \tag{3.8}$$

$$g_2 := {x_0}^2 + {x_1}^2 + {x_2}^2 + {x_3}^2 - 1 = 0 \tag{3.9}$$

All the points that satisfy the Equation (3.8) belong to the 6-dimensional *Study quadric*, S_6^2. Equation (3.9) ensures that the points do not lie on the *exceptional generator*, $x_0 = x_1 = x_2 = x_3 = 0$. The points on S_6^2 are called *kinematic image points* of the corresponding displacement, and the seven-dimensional projective space is called *kinematic image space*. Its variants for describing rigid body displacement in the planar motion group $SE(2)$ (also known as Blaschke mapping [Bottema and Roth, 1990]), 3 dimensional rotational motion group $SO(3)$ also exist. Rigid body displacement in $SO(2)$ can be described using complex numbers.

3.3.2 Tangent Half-Angle Substitution

While setting up the constraint equations, one typically encounters trignometric terms, typically due to the motion of revolute joints, which can be made algebraic using the tangent half-angle substitutions or Weierstrass substitution. For any point $(\cos\theta, \sin\theta)$ on the unit circle $\mathbb{S}^1$, draw a line passing through it and the point $(-1, 0)$. This point crosses the y-axis at some point $y = t$. Using simple geometry, it is trivial to show that $t = \tan(\theta/2)$ (see Fig. 3.4). The equation for the drawn line is $y = (1+x)t$. The equation for the intersection of the line and circle is then a quadratic equation involving t. The two solutions to this equation are $(-1, 0)$ and $(\cos\theta, \sin\theta)$. This allows us to write the latter as rational functions of t as shown below.

$$\sin(\theta) = \frac{2t}{1+t^2}\,, \cos(\theta) = \frac{1-t^2}{1+t^2}\,, \tan(\theta) = \frac{2t}{1-t^2} \tag{3.10}$$

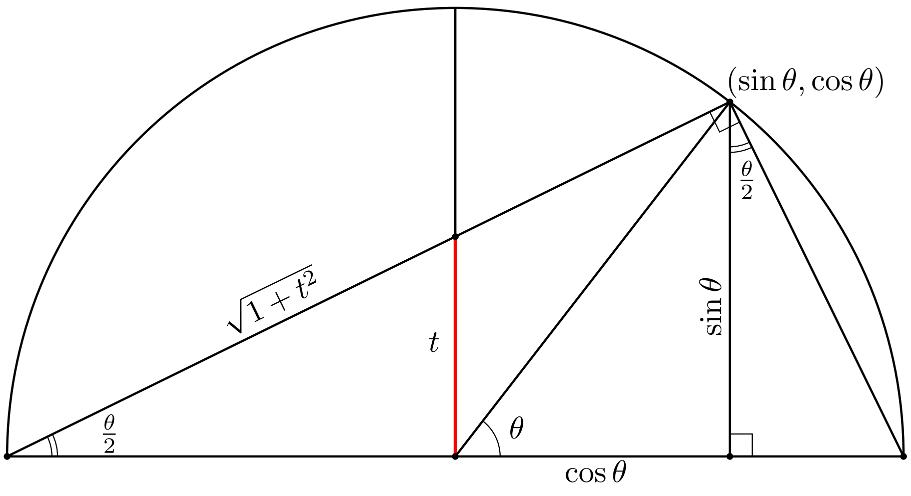


Figure 3.4: Geometric proof of tangent half-angle substitution

3.3.3 Towards Global Kinematics

Analytic descriptions of kinematic chains lead to parametric and implicit representations. These are easy to set up but difficult to solve. Very often only a single numerical solution is obtained. Complete analysis and synthesis needs all solutions. An algebraic description of constraint equations allows the use of powerful methods and algorithms from algebraic geometry. An important first task is to find the simplest algebraic constraint equations that describe the chains. There exists always a best-adapted coordinate system for a mechanism or at least for one kinematic chain in a more complicated mechanism. When a kinematic chain is represented in its "best"-adapted coordinate system, then it is called canonical chain. Geometric and algebraic preprocessing is needed before elimination, Gröbner basis computation or numerical solution process starts. Algebraic constraint equations yield answers to the overall behavior of a kinematic chain which provides insights into *global kinematics* of the mechanism [Husty, 2017b]. Solutions to the inverse kinematics of a general 6R serial chain robot [Raghavan and Roth, 1993] and to the direct kinematics of the general Stewart-Gough platform [Husty, 1996], which yields polynomials of degree 16 and degree 40, respectively, are major advances in the last century. The disadvantage of this approach is that such an analysis is not always straight forward and the involved algorithms are NP-hard.

3.4 Screw Theory

Screw theory has proven to be a powerful mathematical tool for the local analysis of complex mechanisms and robots. It provides a quick and efficient way to describe the configuration of a system at any given instant. There are two core reasons behind it: 1) the rigid body transformations can be analytically described in screw coordinates using exponential mapping, 2) most kinematic joints can be described as a combination of 1-DOF screw joints. An introduction to screw theory can be found in the Appendix D.

3.4.1 Matrix Exponential and Matrix Logarithm Maps

Any rigid body configuration can be achieved by starting from a fixed reference frame and integrating a twist for a specified time. Such a motion resembles the motion of a screw, rotating about and translating along the same fixed screw axis. The observation that all the configurations can be achieved a screw motion motivates a six parameter representation of the configuration called the *exponential coordinates* [Lynch and Park, 2017].

Definition 11 (Matrix Exponential) *Let $\boldsymbol{S} = (\boldsymbol{\omega}, \boldsymbol{v})$ denote the screw coordinates. The matrix exponential is defined as $\exp : [\boldsymbol{S}]\theta \in se(3) \rightarrow \boldsymbol{T} \in SE(3)$. If $\|\omega\| = 1$ then for any distance $\theta \in \mathbb{R}$ traveled along the screw axis or any angle $\theta \in \mathbb{R}$ rotated about the screw axis,*

$$\boldsymbol{T} = \exp[\boldsymbol{S}]\theta = \begin{bmatrix} \exp[\boldsymbol{\omega}]\theta & (\boldsymbol{I}\theta + (1-\cos\theta)[\boldsymbol{\omega}] + (\theta - \sin\theta)[\boldsymbol{\omega}]^2)\boldsymbol{v} \\ \boldsymbol{0} & 1 \end{bmatrix} \in SE(3) \tag{3.11}$$

where,

$$\exp[\boldsymbol{\omega}]\theta = \boldsymbol{I} + \sin\theta[\boldsymbol{\omega}] + (1-\cos\theta)[\boldsymbol{\omega}]^2) \in SO(3)\ . \tag{3.12}$$

If $\|\omega\| = 0$ and $\|v\| = 1$, then

$$\boldsymbol{T} = \exp[\boldsymbol{S}]\theta = \begin{bmatrix} \boldsymbol{I} & \boldsymbol{v}\theta \\ \boldsymbol{0} & 1 \end{bmatrix} \in SE(3)\ . \tag{3.13}$$

On the contrary, given an arbitrary pose $(\boldsymbol{R}, \boldsymbol{p}) \in SE(3)$, one can always find a screw axis $\boldsymbol{S} = (\boldsymbol{\omega}, \boldsymbol{v})$ and a scalar θ representing it. The matrix $[\boldsymbol{S}]\theta$ is called as the *matrix logarithm* of the pose $(\boldsymbol{R}, \boldsymbol{p})$.

Definition 12 (Matrix Logarithm) *The matrix logarithm is defined as* $\log : \boldsymbol{T} \in SE(3) \to [\boldsymbol{S}]\theta \in se(3)$. *If*

$$\boldsymbol{T} = \exp{[\boldsymbol{S}]\theta} = \begin{bmatrix} \boldsymbol{R} & \boldsymbol{p} \\ \boldsymbol{0} & 1 \end{bmatrix} \tag{3.14}$$

then, the matrix

$$[\boldsymbol{S}]\theta = \begin{bmatrix} [\boldsymbol{\omega}]\theta & \boldsymbol{v}\theta \\ \boldsymbol{0} & 0 \end{bmatrix} \tag{3.15}$$

is the matrix logarithm of $\boldsymbol{T} = (\boldsymbol{R}, \boldsymbol{p})$.

The algorithm to compute the matrix logarithm can be found in [Lynch and Park, 2017] and is skipped here for brevity. The exponential mapping is a key ingredient in the product of exponentials (POE) formula [Brockett, 1984] which can be used to compute the forward kinematics of serial kinematic chains or to set up geometric loop closure constraint equations for parallel mechanisms. The exponential mapping is analytic i.e. the function is given by a convergent power series. The variety thus obtained from loop closure constraints determined using POE formula is also analytic in nature. Another advantage here is that the use of the exponential mapping in terms of screw coordinates gives rise to explicit closed-form formulae for its derivatives.

3.4.2 Screw Representation of Joint Motion

Lower kinematic pairs with one DOF allow the interconnected bodies to perform screw motions between each other with a certain pitch h. Revolute joints can be modeled with zero pitch screws $h = 0$ and prismatic joints are modeled with infinite pitch screws $h = \infty$. If $\boldsymbol{S}$ denotes the unit screw axis, then for different 1 DOF lower kinematic pairs, $\boldsymbol{S}$ is given by:

$$\boldsymbol{S}^{\text{revolute}} = \begin{bmatrix} \hat{\boldsymbol{s}} \\ \boldsymbol{0} \end{bmatrix} \qquad \boldsymbol{S}^{\text{screw}} = \begin{bmatrix} \hat{\boldsymbol{s}} \\ h\hat{\boldsymbol{s}} \end{bmatrix} \qquad \boldsymbol{S}^{\text{prismatic}} = \begin{bmatrix} \boldsymbol{0} \\ \hat{\boldsymbol{s}} \end{bmatrix} \tag{3.16}$$

where $\hat{s}$ denotes the unit vector along the joint axis resolved in the joint frame.

3.4.3 Towards Local Analysis

One of the biggest advantages of using methods from screw theory is that it allows an easy set up of implicit constraint equations. Further, the derivatives of these equations can be derived in closed form. If a feasible configuration of the mechanism V_q is known, an exhaustive local analysis of the c-space geometry as well as finite curves passing through this point can be performed. It provides a powerful setting for studying the mechanism behavior and classification of its singularities [Müller, 2019]. A

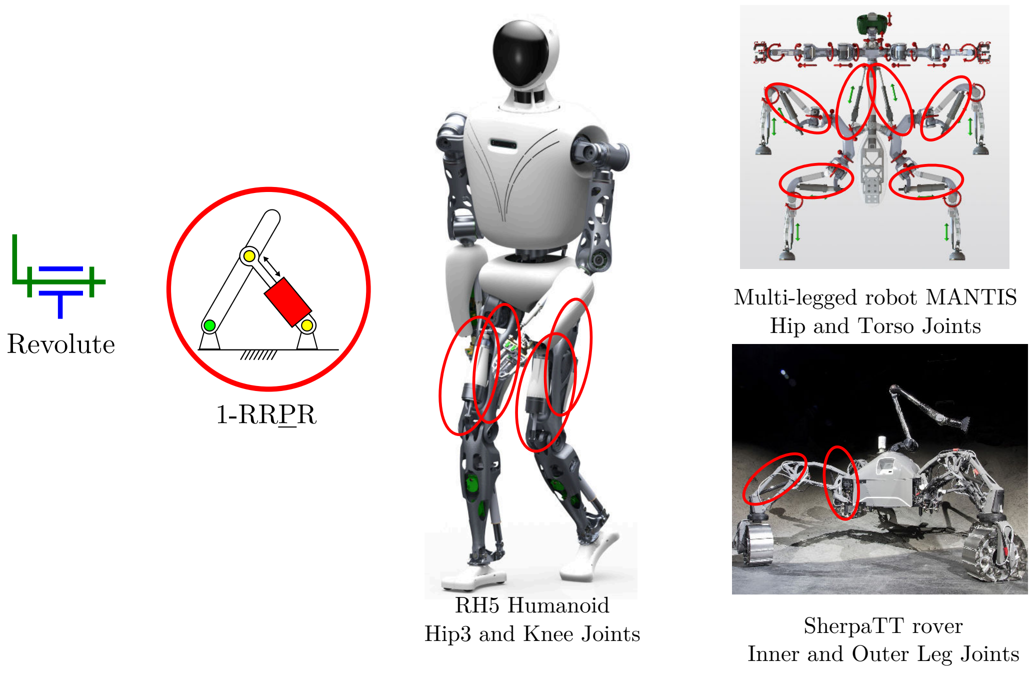


Figure 3.5: Revolute joint abstractions with Lambda mechanism in robots at DFKI-RIC

disadvantage of this approach is that the mobility and singularity analysis requires a prior knowledge of the actual solution set V. This disadvantage can be leveraged by an algebraic parameterization of the constraint equations after geometric preprocessing and the use of powerful tools from computational algebraic geometry to solve them.

3.5 Example: Lambda mechanism

A Lambda mechanism is basically a planar mechanism with triangular geometry as shown in Fig. 3.6. Body B1 forms a one-link arm, while B2 and B3 are the cylinder and piston, respectively, of a linear actuator. Joints 1, 2 and 3 participate in the kinematic loop, with joint 3 being the actuated prismatic joint. This section presents the study of this 1-RRPR mechanism which has been used for the abstraction of a revolute joint in various robot designs (see Fig. 3.5 for its applications in robots at DFKI-RIC).

3.5.1 Mobility Analysis

The general mobility of this mechanism can be calculated with the help of Kutzbach-Grübler criteria, see Equation 3.2. Since it is a planar mechanism, $s = 3$. Hence, the mobility of this mechanism is $d_s(\mathcal{M}) = 3(4 - 4 - 1) + 4 = 1$.

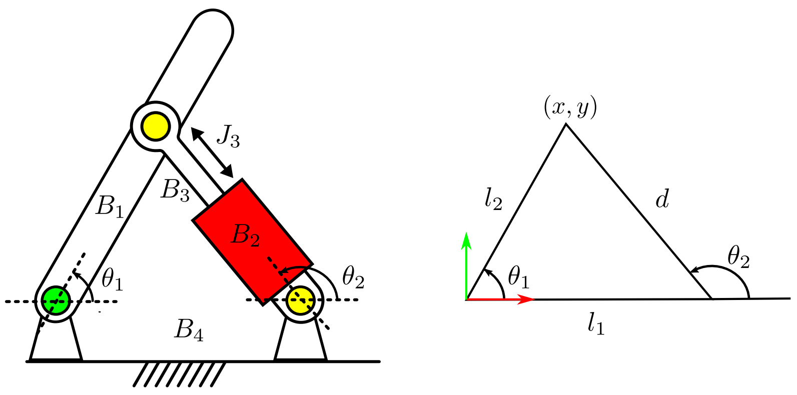
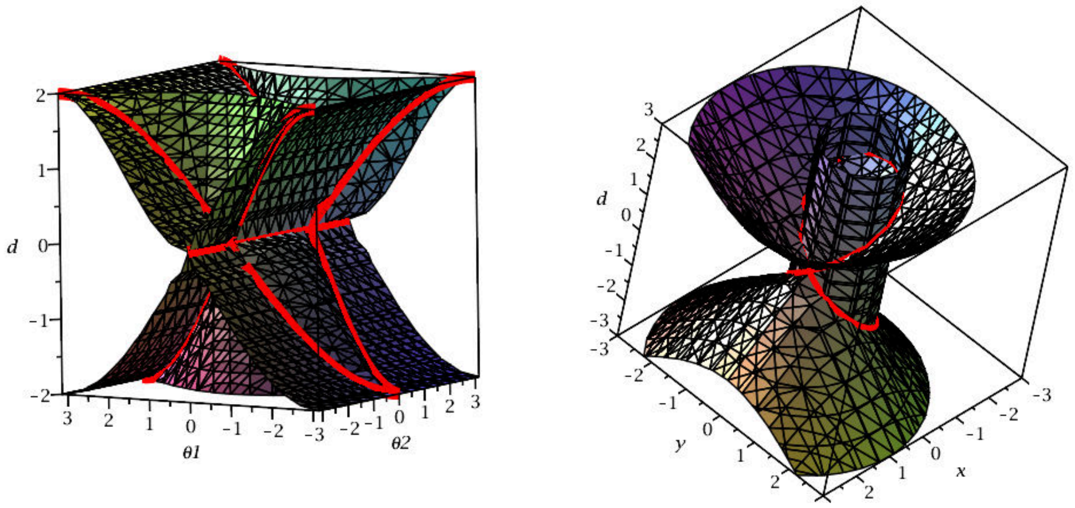


Figure 3.6: Lambda mechanism and its triangular geometry



(a) Intersection of analytic surfaces given by Equation 3.17 and Equation 3.18

(b) Intersection of cylinder (Equation 3.20a) and double cone (Equation 3.20b)

Figure 3.7: C-space of lambda mechanism

3.5.2 Geometric Analysis

In the following, we will perform the geometric analysis of this mechanism using both the approaches described previously in this chapter. First, we develop an analytic formulation using joint angles and displacements, and then develop an algebraic formulation using the polynomial description of the geometric constraint equations. We set the link parameters as $l_1 = l_2 = 1$ and hence the corresponding geometry is that of an equilateral triangle.

3.5.2.1 Analytic Formulation

The displacement of joints J_1 and J_2 can be parameterized using angles θ_1 and θ_2 and the movement of the slider can be parameterized with linear displacement variable d. These are measured as absolute coordinates in the reference frame defined in Fig. 3.6. Hence, the c-space of this mechanism can be described using the choice of coordinates (θ_1, θ_2, d) and is an analytic variety in $V^n = \mathbb{T}^2 \times \mathbb{R}^1$. Using the law of cosine in trigonometry, one can establish the constraint equation for this mechanism:

$$d^2 = l_1^2 + l_2^2 - 2l_1l_2\cos\theta_1 \tag{3.17}$$

To calculate the passive joint angle (θ_2), one can use the formula below:

$$d\cos\theta_2 = l_2\cos\theta_1 - l_1 \tag{3.18}$$

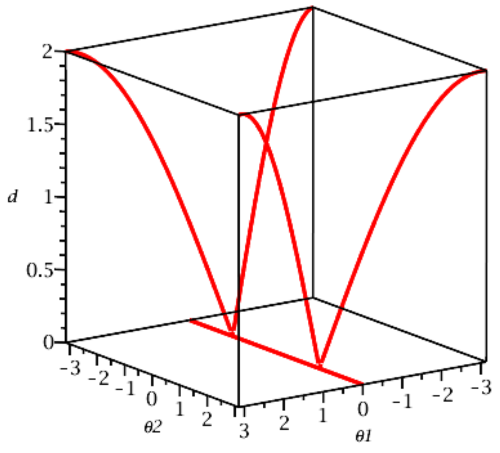
Equation 3.17 and Equation 3.18 are analytic surfaces and their intersection is an analytic variety as shown in Fig. 3.7a. However, it is to be noted that d should always be greater than or equal to 0 and hence, the part of the curve where d is negative should be disregarded. The final analytic variety of this mechanism is shown in Fig. 3.8a. It can be noticed that when $\theta_1 = 0$, then $d = 0$ and θ_2 is undefined and hence, represents a c-space singularity.

From an input-output viewpoint, d is the input variable and θ_1 is the output variable. It immediately follows that Equation 3.17 also provides an explicit closed form solution to the inverse geometric (or kinematic) problem. Rearranging the Equation 3.17, a closed form solution to the direct geometric (or kinematic) problem can be derived. There are two solutions to the forward geometric problem as shown in Equation 3.19.

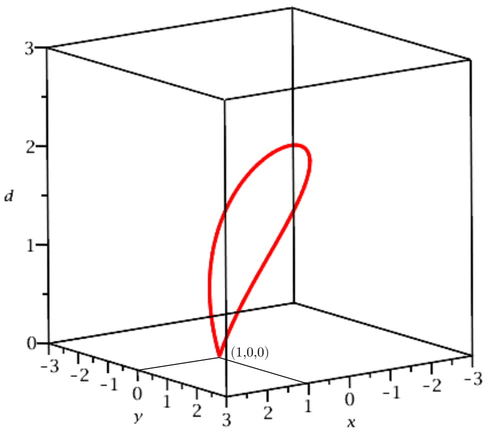
$$\begin{aligned} \theta_1 &= \arccos\frac{l_1^2 + l_2^2 - d^2}{2l_1l_2} \\ \theta_1 &= \operatorname{atan2}(\pm\sin\theta_1, \cos\theta_1) \end{aligned} \tag{3.19}$$

3.5.2.2 Algebraic Formulation

Since, it is a planar mechanism, one could formulate the constraint equations in the form of a set of polynomials by choosing an algebraic parameterization of the links in the XY plane. To this end, let $(x = l_2\cos\theta_1, l_2\sin\theta_1)$ denote the coordinates of the crank link B_1 and $(l_1, 0)$ denote the coordinates of the point on the ground where



(a) Analytic variety



(b) Algebraic variety

Figure 3.8: C-space of lambda mechanism with $d \geq 0$

cylinder B_2 is attached. The loop constraint equations are given as the following:

$$x^2 + y^2 = l_2^2 \tag{3.20a}$$

$$(x - l_1)^2 + y^2 = d^2 \tag{3.20b}$$

In this case the c-space coordinates are $(x, y, d) \in \mathbb{R}^3$ and any feasible values of these coordinates satisfying Equation 3.20 fully describe the mechanism. The first constraint equation represents the surface of a cylinder (Equation 3.20a) while the second equation represents a double cone (Equation 3.20b). The c-space is in this case is the set of all points lying on the intersection of these two surfaces which looks like a bent infinity shaped curve (or lemniscate curve) in 3-space of c-space coordinates. However, it should be noted that $d \geq 0$ so the part of the curve where d is negative should be disregarded. The algebraic variety representing the c-space of the mechanism is shown in Fig. 3.8b. A cusp can be noticed at the point $(x, y, d) = (1, 0, 0)$ which shows the singular configuration of the c-space as V is not a smooth manifold at this point.

From an input-output viewpoint, d is the input variable and the pair (x, y) is the output variable. In the algebraic formulation, the solution to the inverse kinematics problem can be derived by substituting Equation 3.20a in Equation 3.20b and eliminating the variable y. The solution is unique and is given by:

$$\begin{aligned} (x - l_1)^2 + l_2^2 - x^2 &= d^2 \\ d &= \sqrt{l_1^2 + l_2^2 - 2 l_1 x}\ . \end{aligned} \tag{3.21}$$

Similarly, the forward geometric problem can be solved by manipulating the constraint equations to write the variables (x, y) as a function of d. As noted earlier, it can be noted that this problem has two solutions given algebraically by Equation 3.22.

$$\begin{aligned} x(d) &= \frac{l_1^2 + l_2^2 - d^2}{2l_1} \\ y(d) &= \pm\sqrt{l_2^2 - \left(\frac{l_1^2 + l_2^2 - d^2}{2l_1}\right)^2} \end{aligned} \tag{3.22}$$

The above equation is also a parametric equation describing the c-space variety in $\mathbb{R}^3$ as shown in Fig. 3.8b.

3.6 Conclusion

This chapter presents a summary of modern geometric approaches in the analysis of robots and mechanisms. Two approaches namely, screw theory and algebraic geometry, are briefly discussed and their corresponding advantages in the local and global kinematic analysis of the mechanisms are highlighted. Lastly, a simple one DOF planar mechanism which converts the linear motion of an actuator to the rotary motion is studied with both analytic and algebraic approaches. It can be observed that for simple cases, like that of the lambda mechanism, the two approaches are equivalent in terms of their ease of use and the insights they provide in the mechanism analysis. In the upcoming chapters, where more complex mechanisms have been studied, we will take the geometric approach which suits better to the particular geometry or type of the mechanism being studied.

Chapter 4

Study of 2SPRR+1U Device for Abstraction of Universal Joint

This chapter presents the study of the novel 2SPRR+1U mechanism and its existing variant 2SPU+1U mechanism which has been used for the abstraction of a universal joint in various robot designs (see Fig. 4.1). The chapter is organized as follows: Section 4.1 presents the motivation for the mechanism's design and highlights its novelty. Section 4.2 presents the manipulator's architecture and constraint equations. Section 4.3 presents the solutions to the direct and inverse kinematic problems by utilizing tools from computational algebraic geometry. Section 4.4 presents the workspace characterisation, description of its singularity curves and performance analysis and Section 4.6 concludes this chapter. The content of this chapter is based on [Kumar et al., 2018c].

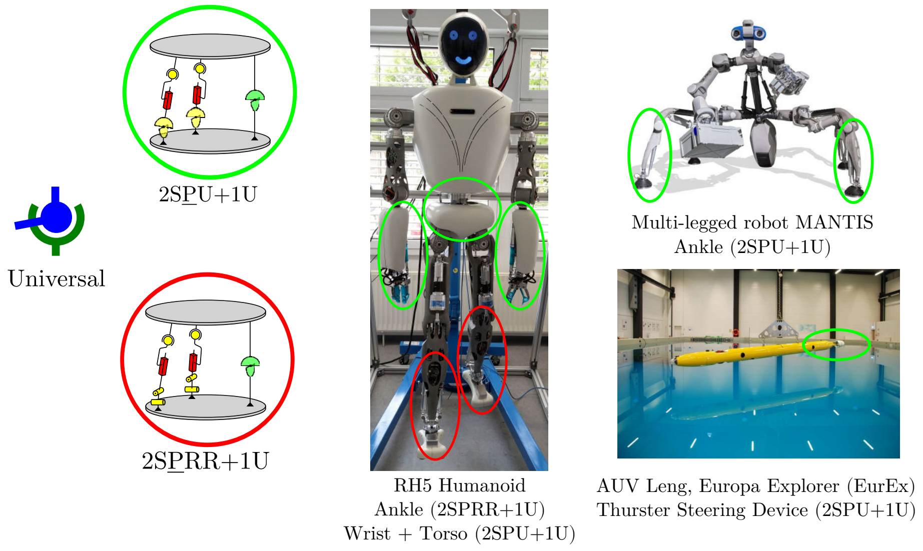

2SPU+1U
Universal
2SPRR+1U
RH5 Humanoid
Ankle (2SPRR+1U)
Wrist + Torso (2SPU+1U)
Multi-legged robot MANTIS
Ankle (2SPU+1U)
AUV Leng, Europa Explorer (EurEx)
Thurster Steering Device (2SPU+1U)

Figure 4.1: Universal joint abstractions in robots at DFKI-RIC

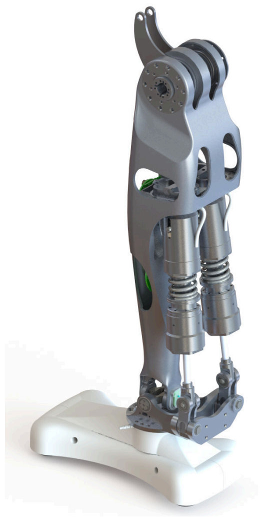
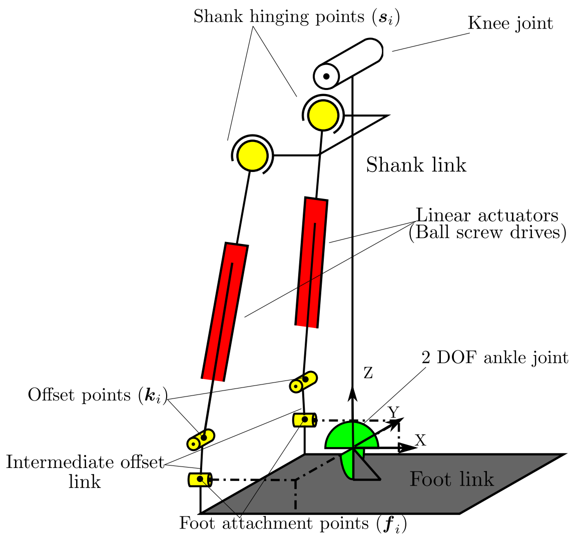


Figure 4.2: CAD prototype of Ankle joint


Shank hinging points ($\boldsymbol{s}_i$)
Knee joint
Shank link
Linear actuators (Ball screw drives)
2 DOF ankle joint
Z
Y
X
Offset points ($\boldsymbol{k}_i$)
Intermediate offset link
Foot link
Foot attachment points ($\boldsymbol{f}_i$)
Figure 4.3: Scheme of the mechanism

4.1 Design Motivation

Fig. 4.2 shows the novel two degrees of freedom orientational parallel mechanism of type 2SPRR+1U which is used as an ankle joint in the RH5 humanoid robot recently developed at DFKI-RIC. The kinematic actuation principle of this mechanism comprises of a motion constraint generator leg with a universal joint (U) and two auxiliary actuation legs of type SPRR i.e. they containt a spherical (S), prismatic (P) and two revolute (RR) joints in series as shown in Fig. 4.3. It is well known that during walking, the torque required for the pitch movement is larger than the torque required for the roll movements [Lohmeier et al., 2006]. When the two motors are actuated in the same direction, the mechanism produces a pitch only movement demonstrating good torque transmission characteristics. It has been shown in biomechanics studies that during the ankle pitch movement of human gait, a peak torque between 105 Nm and 120 Nm is required when flexion/extension angle is between $-6°$ and $-12°$[Zoss et al., 2006]. To reflect this in the ankle design, the foot attachment points of the two linear actuators may be displaced along the z-axis by 30 mm. Utilizing a common universal joint at the offset points, as in the case of 2SPU+1U mechanism, reduces the workspace of the roll movement. Instead, two skew revolute joints, with axes parallel to the axes of universal joint on constraint generator leg, connected by an intermediate offset link are used to provide the desired torque characteristics in the pitch movement with minimal influences on the motion range of roll movement.

4.2 Architecture and Constraint Equations

The mobility of a mechanism ($\mathcal{M}$) can be calculated with the help of Kutzbach-Grübler criteria as follows: $d_s(\mathcal{M}) = s(n-m-1)+f = s(-c)+f$, where n is number of links in the mechanism $3+3+2 = 8$, m is total number of joints $4+4+1 = 9$, f is total dof of joints $2+6+6 = 14$, and s is the motion parameter. Since, it is a spatial mechanism, $s = 6$. Hence, the mobility can be calculated as $d_s(\mathcal{M}) = 6(8-9-1)+14 = 2$.

The manipulator architecture and geometry is shown in Fig. 4.4. Let us define a set of three points: shank point ($\boldsymbol{s}_i$), foot attachment point ($\boldsymbol{f}_i$) and the offset point ($\boldsymbol{k}_i$) on the two auxiliary actuation legs of the mechanism. The base frame O is attached to the shank link and is coincident with the end effector (EE) frame E attached to the foot link in zero configuration. The intermediate offset link $\boldsymbol{f}_i\boldsymbol{k}_i$ rotates about the x-axis (denoted as $\hat{\boldsymbol{n}}$) of the frame defined at $\boldsymbol{f}_i$, thus point $\boldsymbol{k}_i$ moves on a circle of radius r equal to the length of the link, $\|\boldsymbol{f}_i - \boldsymbol{k}_i\|$. The length of the linear actuators (d_i) is the norm of the vector ($\boldsymbol{k}_i - \boldsymbol{s}_i$). We also define a vector $\boldsymbol{\delta}_i := (\boldsymbol{s}_i - \boldsymbol{f}_i)$.

The constraint equations of the manipulator are the following:

$$d_i^2 = \|\boldsymbol{k}_i - \boldsymbol{s}_i\|^2 = \|\boldsymbol{p}_i - \boldsymbol{s}_i\|^2 + \|\boldsymbol{k}_i - \boldsymbol{p}_i\|^2 \ \ , i \in \{1,2\} \tag{4.1}$$

We can rewrite Equation 4.1 purely as a function of $(\boldsymbol{s}_i, \hat{\boldsymbol{n}}, \boldsymbol{f}_i)$.

$$\begin{aligned} d_i^2 &= \|\hat{\boldsymbol{n}} \cdot \boldsymbol{\delta}_i\|^2 + (\|\boldsymbol{f}_i - \boldsymbol{p}_i\| - r)^2 \\ d_i^2 &= \|\hat{\boldsymbol{n}} \cdot \boldsymbol{\delta}_i\|^2 + (\sqrt{\|\boldsymbol{\delta}_i\|^2 - \|\hat{\boldsymbol{n}} \cdot \boldsymbol{\delta}_i\|^2} - r)^2 \\ d_i^2 &= \|\hat{\boldsymbol{n}} \cdot \boldsymbol{\delta}_i\|^2 + (\|\hat{\boldsymbol{n}} \times \boldsymbol{\delta}_i\| - r)^2 \end{aligned} \tag{4.2}$$

For the purpose of visualization or computing passive joint angles, it is necessary to compute the point $\boldsymbol{k}_i$ which is given by Equation 4.3.

$$\boldsymbol{k}_i = \boldsymbol{f}_i + r\frac{\boldsymbol{\delta}_i - (\hat{\boldsymbol{n}} \cdot \boldsymbol{\delta}_i)\hat{\boldsymbol{n}}}{\|\boldsymbol{\delta}_i - (\hat{\boldsymbol{n}} \cdot \boldsymbol{\delta}_i)\hat{\boldsymbol{n}}\|} \tag{4.3}$$

The orientation of the moving platform is parameterized by roll (θ, around X axis) and pitch (ϕ, around Y axis) angles such that ${}^{O}\boldsymbol{R}_E = \mathrm{Rot}(X,\theta) \cdot \mathrm{Rot}(Y,\phi)$. The revolute joint axis vector ($\hat{\boldsymbol{n}}$) and the foot attachment point ($\boldsymbol{f}_i$) are expressed in global coordinate frame using $\hat{\boldsymbol{n}} = {}^{O}\boldsymbol{R}_E \cdot \hat{\boldsymbol{n}}_E$ and $\boldsymbol{f}_i = {}^{O}\boldsymbol{R}_E \cdot \boldsymbol{f}_i^E$ respectively where $\hat{\boldsymbol{n}}_E$ and $\boldsymbol{f}_i^E$ denote the revolute joint axis and foot attachment point vector in EE frame.

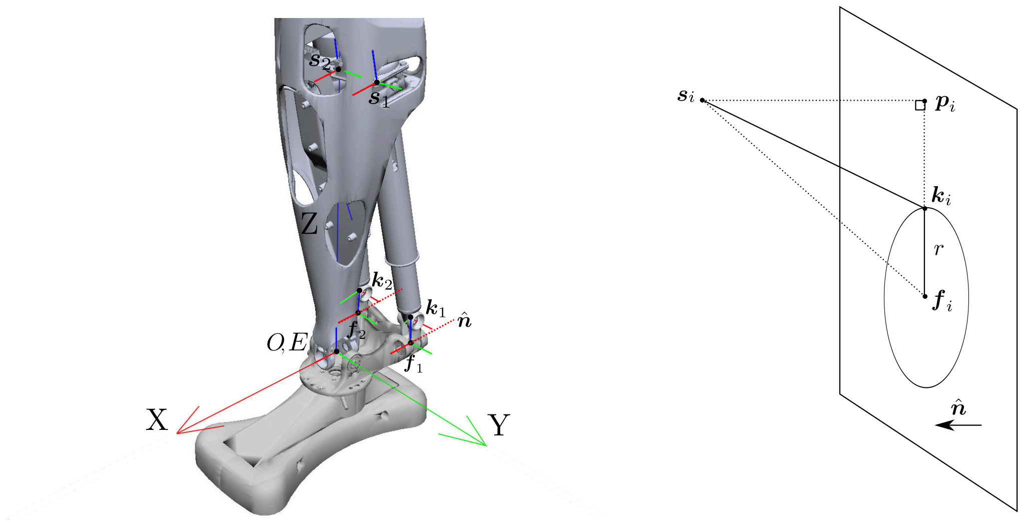


Figure 4.4: Geometry of the 2 dof ankle mechanism (RGB colours denote XYZ axes)

4.3 Solving Forward and Inverse Kinematics

Algebraic geometry techniques have proven to be useful in solving the forward kinematics of parallel manipulators but they require the constraint equations to be algebraic. Tangent half angle substitutions might leave the constraint equations undefined for π orientations. Hence, in order to have an algebraic description of the mechanism's constraint equations, cosines and sines are replaced by $\cos(\theta) = x$, $\sin(\theta) = y$, $\cos(\phi) = u$ and $\sin(\phi) = v$ in ${}^{O}\boldsymbol{R}_{E}$ though it comes at a cost of adding two more equations to the ideal set. To this end, rearranging Eq. (4.2) and squaring to avoid the square root term $\|\hat{\boldsymbol{n}} \times \boldsymbol{\delta}_i\|$ leads to four algebraic constraint equations:

$$g_1 := (d_1^2 - \|\hat{\boldsymbol{n}} \cdot \boldsymbol{\delta}_1\|^2 - \|\hat{\boldsymbol{n}} \times \boldsymbol{\delta}_1\|^2 - r^2)^2 - 4\,\|\hat{\boldsymbol{n}} \times \boldsymbol{\delta}_1\|^2\, r^2 = 0 \tag{4.4a}$$

$$g_2 := (d_2^2 - \|\hat{\boldsymbol{n}} \cdot \boldsymbol{\delta}_2\|^2 - \|\hat{\boldsymbol{n}} \times \boldsymbol{\delta}_2\|^2 - r^2)^2 - 4\,\|\hat{\boldsymbol{n}} \times \boldsymbol{\delta}_2\|^2\, r^2 = 0 \tag{4.4b}$$

$$g_3 := x^2 + y^2 - 1 = 0 \tag{4.4c}$$

$$g_4 := u^2 + v^2 - 1 = 0 \tag{4.4d}$$

After substituting the geometric dimensions provided in Table 4.1, the constraint equations are only a function of variables x, y, u, v, d_1 and d_2. g_1 and g_2 are 16 degree polynomials and are quite long to be shown here.

The solution to inverse kinematics problem (IKP) of the manipulator is straightforward and unique for a given orientation of the moving platform as the joint variables d_i can be easily calculated from Eq. (4.2). It is noteworthy that when the roll angle is zero, Eq. (4.2) yields $d_1 = d_2$.

The direct kinematics problem (DKP) aims to find the variables x, y, u and v when

i	$\boldsymbol{s}_i$	$\boldsymbol{f}_i^E$	$\hat{\boldsymbol{n}}_E$	$\Vert\boldsymbol{f}_i\boldsymbol{k}_i\Vert$
1	$(-22.30, 25, 291.27)^T$	$(-70, 40, 0)^T$	$(1, 0, 0)^T$	30
2	$(-22.30, -25, 291.27)^T$	$(-70, -40, 0)^T$	$(1, 0, 0)^T$	30

Table 4.1: Geometric dimensions (in mm) of the ankle mechanism

Number of solutions	0	2	4	6	8
Number of poses (/2601)	124	268	2146	63	0
Percentage	4.77	10.30	82.51	2.42	0

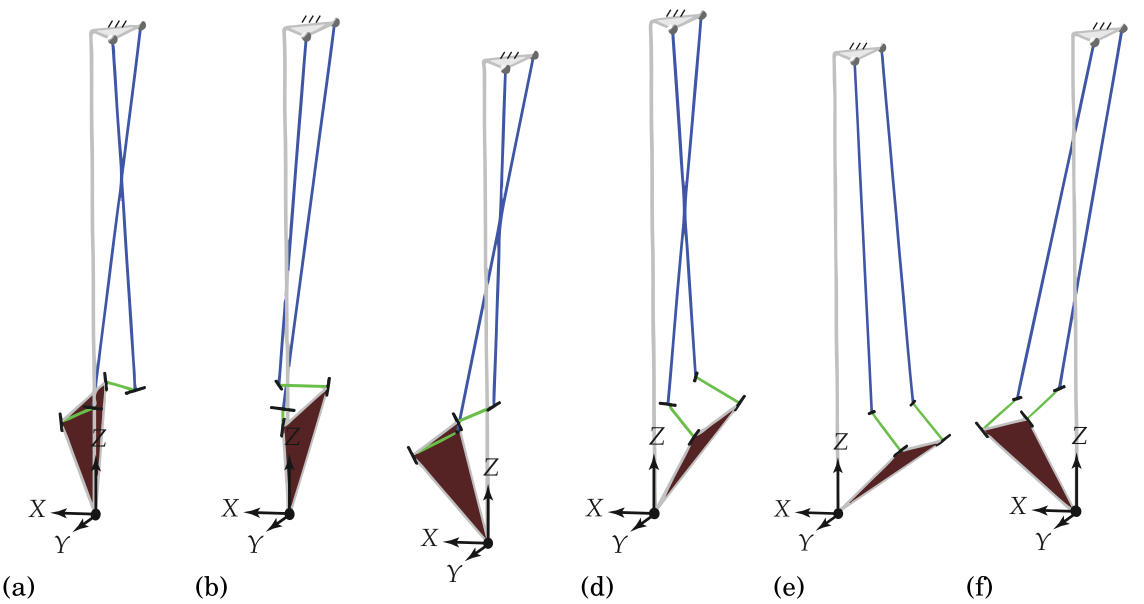
Table 4.2: Percentage of real solutions to direct kinematics

the prismatic joint lengths are specified. In search of maximum number of solutions to DKP (assembly modes), an ideal of the constraint polynomials g_i is defined: $\mathcal{I} = \langle g_1, g_2, g_3, g_4\rangle \,|\, \mathcal{I} \subseteq k[u, v, x, y]$. Finding the Groebner basis with a pure lexicographic ordering of the orientation parameters in any order leads to a univariate polynomial of degree 32. Since squaring two of the four constraint equations quadruples the number of solutions, the number of solutions must be quartered. Hence, the upper limit to DKP solutions of the manipulator under study is eight. To investigate the number of real solutions, **RootFinding[Isolate]** function of Maple$^{\text{TM}}$is used. The algorithm behind this function finds out the rational univariate representation of the set of polynomials and isolates the real roots of these univariate polynomials based on Descartes' rule of sign and the bisection strategy in a unified framework. The variables d_1 and d_2 are varied from 221 mm to 331 mm (physical motion range of linear actuators) with an increment of 6 mm and the percentage of the number of real DKP solutions is listed in Table 4.2. It is evident that the maximum number of real solutions for the considered set of prismatic joint lengths is six. Figure 4.5 shows six such assembly modes when $d_1 = 221\,\text{mm}$ and $d_2 = 228.3\,\text{mm}$. It is speculated that a different choice of design parameters might lead to eight real solutions to DKP. In the physical construction of the ankle joint, passive joint limits are chosen such that there exists a unique solution to forward kinematics for a given input of actuator lengths in their feasible motion range (for instance Fig. 4.5e).

4.4 Workspace, Singularity and Performance Analysis

4.4.1 Configuration Space and Workspace

To demonstrate the suitability of the novel 2-SPRR+1U mechanism as a humanoid ankle joint, it is important to compute and characterize its feasible workspace in orientation and configuration domains. The feasible configuration space is calcu-



(a) $\theta = 172.32°$ $\phi = -92.94°$

(b) $\theta = 172.33°$ $\phi = -78.30°$

(c) $\theta = 175.32°$ $\phi = -116.30°$

(d) $\theta = 175.30°$ $\phi = -54.91°$

(e) $\theta = 5.03°$ $\phi = 39.28°$

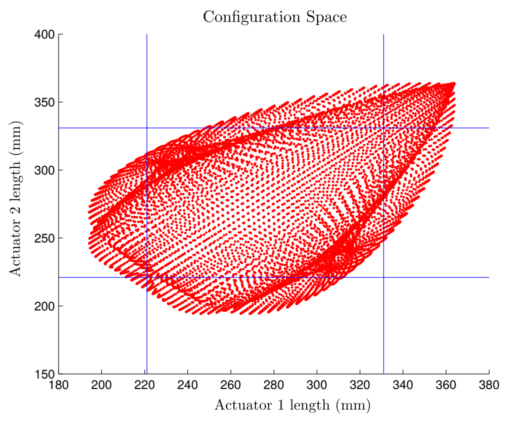
(f) $\theta = 5.09°$ $\phi = 131.92°$

Figure 4.5: Assembly modes for $d_1 = 221\,\text{mm}$ and $d_2 = 228.3\,\text{mm}$

lated by varying the orientation variables describing foot rotation, roll (θ) and pitch (ϕ) angles, in the range $[-\pi, \pi]$. Then the physical limits of the linear actuators ($d_i \in [221, 331]\text{mm}$) are imposed to compute the workspace of the mechanism under actuator constraints. The resulting configuration space and orientation workspace are shown in Fig. 4.6. It is possible to take into account physical limits of passive joints in the mechanism to further compute the physically realizable workspace which is indicated with a closed curve in the figure. The final range of motion (ROM) of the proposed ankle mechanism is more than that of an average human and is presented in Table 4.3 (compare with [Zoss et al., 2006]). Hence, the available range of motion (ROM) in the humanoid ankle is between $-57°$ and $57°$ for the roll angle (ϕ) and between $-51.5°$ and $45°$ for the pitch angle (θ).

4.4.2 Singularity Analysis

The ankle mechanism under study does not have any limb singularities since the auxiliary actuation legs do not generate any constraints on the moving platform. Nonetheless, the actuation scheme results in the so called actuation singularities that can be determined through the kinematic Jacobian matrix of the manipulator obtained by the partial differentiation of the constraint polynomials in Eq. (4.4) with



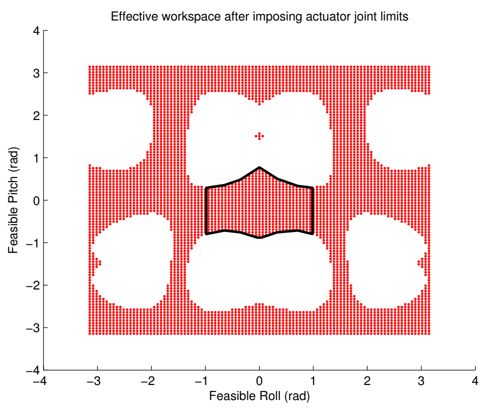


Figure 4.6: Configuration space and orientation workspace under actuator physical limits

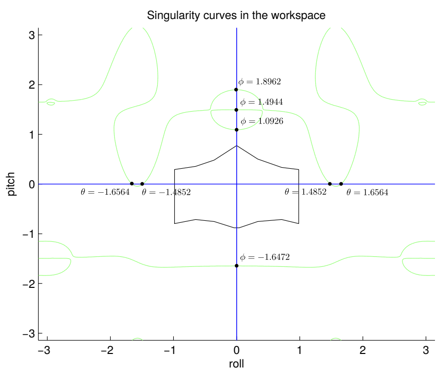


Figure 4.7: Singularity curve

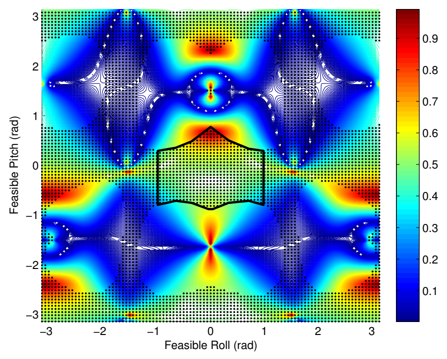
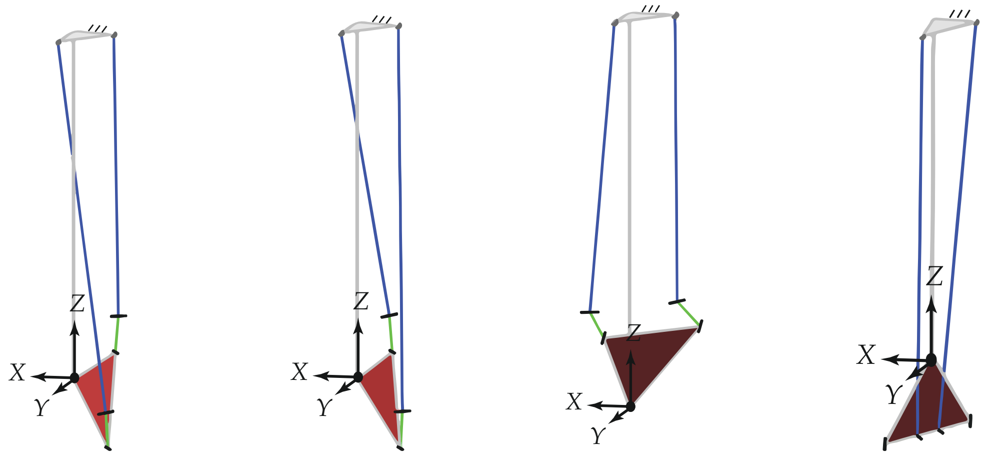


Figure 4.8: Inverse of condition number

respect to the orientation parameters:

$$\boldsymbol{J} = \begin{bmatrix} \dfrac{\partial g_1}{\partial \theta} & \dfrac{\partial g_1}{\partial \phi} \\ \dfrac{\partial g_2}{\partial \theta} & \dfrac{\partial g_2}{\partial \phi} \end{bmatrix} \tag{4.5}$$

The configurations for which the determinant of the Jacobian matrix $\boldsymbol{J}$ vanishes are called actuation singularities. The determinant of $\boldsymbol{J}$ depends only on θ and ϕ. An implicit plot of the equation $\det(\boldsymbol{J}) = 0$ in terms of the orientation variables θ and ϕ is shown in Fig. 4.7 which shows the singularity curves in the mechanism's workspace. Also, it can be observed that there exist four singularities each for pure roll ($\phi = 0$) and pure pitch ($\theta = 0$) movements. Fig. 4.9 shows the singular poses for the pure roll and pure pitch movements which are closest to the zero configuration of the mechanism.



(a)
$\theta = -1.49$ rad
$d_1 = 306$ mm
$d_2 = 228$ mm

(b)
$\theta = 1.49$ rad
$d_1 = 228$ mm
$d_2 = 306$ mm

(c)
$\phi = 1.09$ rad
$d_1 = 216$ mm
$d_2 = 216$ mm

(d)
$\phi = -1.65$ rad
$d_1 = 362$ mm
$d_2 = 362$ mm

Figure 4.9: Singularity configurations for pure roll (θ) and pure pitch (ϕ) movements

Range (min. to max.)	Position	Max. abs. force	Max. abs. velocity
Ankle pitch	$-51.5°$ to $45°$	43.8 N m to 110.1 N m	$61\ °\,s^{-1}$ to $154°\,s^{-1}$
Ankle roll	$-57°$ to $57°$	30.6 N m to 57 N m	$118\ °\,s^{-1}$ to $222°\,s^{-1}$
Linear actuator	221 mm to 331 mm	754 N	$81\text{mm}\,s^{-1}$

Table 4.3: Ankle joint specification (total weight of lower leg = 3.2 kg, weight of one actuator = 0.44 kg)

4.4.3 Performance Analysis

The quality of velocity or force transmission of a parallel manipulator can be measured by plotting the inverse of condition number of the kinematic Jacobian matrix($\boldsymbol{J}$) over the manipulator's workspace. The inverse of condition number of the Jacobian is calculated with $c(\boldsymbol{J}) = \frac{1}{\|\boldsymbol{J}\|\|\boldsymbol{J}^{-1}\|}$ where $\|\cdot\|$ represents the Euclidean norm of the matrix. The inverse of the condition number is plotted over the feasible orientation workspace of the ankle as shown in Fig. 4.8. From Fig. 4.8, it is evident that the kinematic Jacobian matrix is well-conditioned in the feasible orientation workspace of the ankle mechanism.

4.4.4 Maximum Velocity and Force Transmission

For practical purposes, it is crucial to calculate the maximum absolute velocity and torque available at the EE, from the maximum force and velocity that can be delivered by the actuators. These are computed with the help of kinematic Jacobian matrix and actuator specification (see Table 4.3).

4.4.4.1 Velocity Transmission Analysis

The velocity transmission of a parallel mechanism is given by:

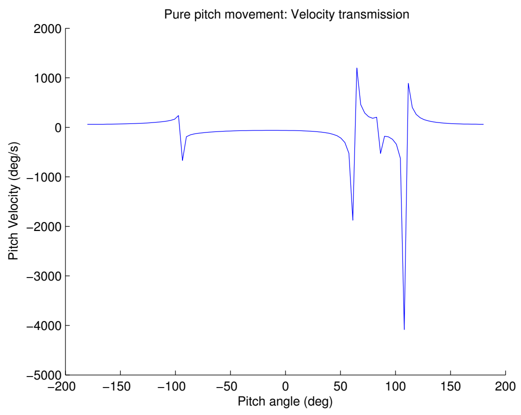
$$\dot{\boldsymbol{x}} = \boldsymbol{J}\dot{\boldsymbol{q}} \tag{4.6}$$

where $\dot{\boldsymbol{q}}$ is the vector containing joint velocities ($\dot{\boldsymbol{q}} = \begin{bmatrix} \dot{d}_1 & \dot{d}_2 \end{bmatrix}$) and $\dot{\boldsymbol{x}}$ is the vector containing task space velocities ($\dot{\boldsymbol{x}} = \begin{bmatrix} \dot{\theta} & \dot{\phi} \end{bmatrix}$). This expression can be used to compute maximum velocity in task space from maximum input velocity in actuation space.

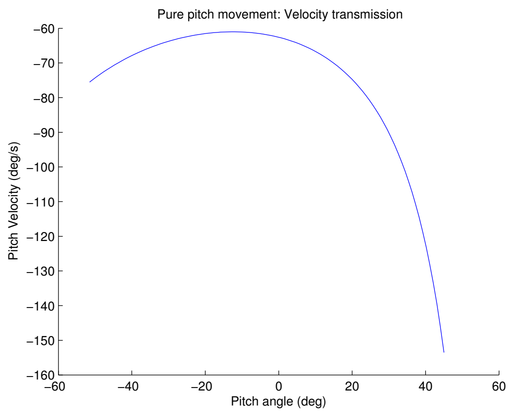
Pure Pitch Movement It is trivial to compute the pure pitch velocity, as we know that for pure pitch movement ($\theta = 0$ and $\dot{\theta} = 0$), the movement required in the linear actuators is identical i.e. $d_1 = d_2$ and $\dot{d}_1 = \dot{d}_2$. To compute the maximum pitch velocity, we vary the pitch angle ϕ in $[-\pi, \pi]$ in the following formula:

$$\dot{\boldsymbol{x}}_{max} = \boldsymbol{J}(0, \phi)\dot{\boldsymbol{q}}_{max} \tag{4.7}$$

where $\dot{\boldsymbol{q}}_{max} = \begin{bmatrix} \dot{d}_{max} & \dot{d}_{max} \end{bmatrix}$. The maximum speed of the linear actuators i.e. $\dot{d}_{max}$ can be found in Table 4.3. Fig. 4.10 shows the pitch velocity component of $\dot{\boldsymbol{x}}_{max}$ in complete and feasible working range of the mechanism. The discontinuities in the curve in Fig. 4.10a show the singular points. Obviously, the roll velocity component is zero and hence not shown in the plots. Fig. 4.10b shows the singularity free pitch velocity transmission curve in physically realizable workspace of the mechanism.



(a) Full working range



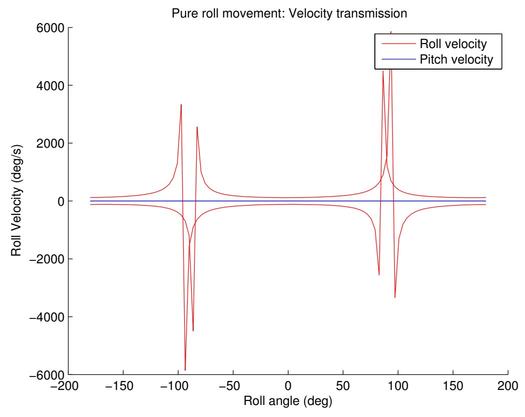
(b) Feasible working range

Figure 4.10: Velocity transmission in pure pitch movement

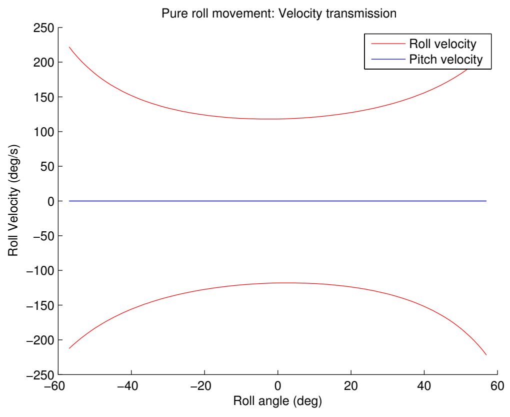
Pure Roll Movement For pure roll movement ($\phi = 0$ and $\dot{\phi} = 0$), such an analysis is not so straightforward because an explicit relation between d_1 and d_2 is not known. However, the ratio of $\dot{d}_1$ and $\dot{d}_2$ can be computed with the help of inverse kinematic Jacobian matrix using the fact that $\phi = 0$ for the pure roll movement:

$$\begin{gathered}\dot{\boldsymbol{q}} = \boldsymbol{J}^{-1}(\theta, 0)\dot{\boldsymbol{x}} \\ \begin{bmatrix} \dot{d}_1 \\ \dot{d}_2 \end{bmatrix} = \begin{bmatrix} a_{11} & a_{12} \\ a_{21} & a_{22} \end{bmatrix} \begin{bmatrix} \dot{\theta} \\ 0 \end{bmatrix} \\ \frac{\dot{d}_1}{\dot{d}_2} = \frac{a_{11}}{a_{21}}\end{gathered} \tag{4.8}$$

where $a_{11}, a_{12}, a_{21}, a_{22}$ are elements of the inverse kinematic Jacobian and are functions of output variables (θ, ϕ). In Equation 4.8, one has the freedom to provide maximum velocity to one of the actuator and calculate the maximum possible speed in the other actuator. For example, if the second actuator is chosen to work at maximum speed i.e. $\dot{d}_2 = \dot{d}_{max}$, then the first actuator speed can be calculated as $\dot{d}_1 = \frac{a_{11}}{a_{21}}\dot{d}_{max}$. Once the two actuator speeds are known, one can substitute them in Equation 4.7 to compute the maximum roll velocity. Fig. 4.11 shows the roll and pitch velocity components of $\dot{\boldsymbol{x}}_{max}$ in complete and feasible working range of the mechanism. The two set of curves in Fig. 4.11a show which actuator was chosen to operate at the maximum speed. The discontinuities in the roll curves show the four singular points. As one can see, the pitch velocity component is zero for a pure roll movement. Fig. 4.11b shows the maximum roll velocity in the feasible working range of the mechanism.



(a) Full working range



(b) Feasible working range

Figure 4.11: Velocity transmission in pure roll movement

In both cases, it can be noticed that the maximum ouput velocity is not constant and depends on the configuration of the mechanism. The range of the maximum output velocity has been documented in Table 4.3.

4.4.4.2 Force Transmission Analysis

The force transmission of a parallel mechanism is given by:

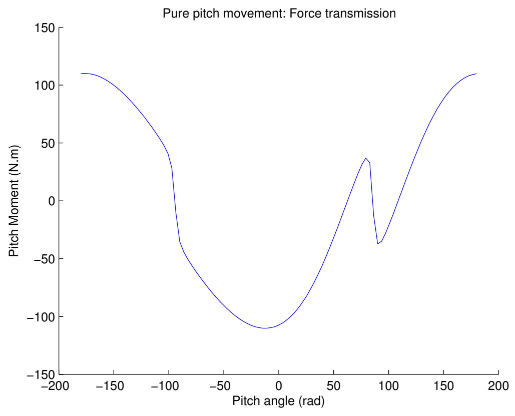
$$\boldsymbol{f} = \boldsymbol{J}^{-T}\boldsymbol{\tau} \tag{4.9}$$

where $\boldsymbol{\tau}$ is the vector containing joint forces ($\boldsymbol{\tau} = \begin{bmatrix} f_1 & f_2 \end{bmatrix}$) and $\boldsymbol{f}$ is the vector containing task space moments ($\boldsymbol{f} = \begin{bmatrix} \tau_{roll} & \tau_{pitch} \end{bmatrix}$). This expression can be used to compute maximum torques in task space from maximum input forces in actuation space.

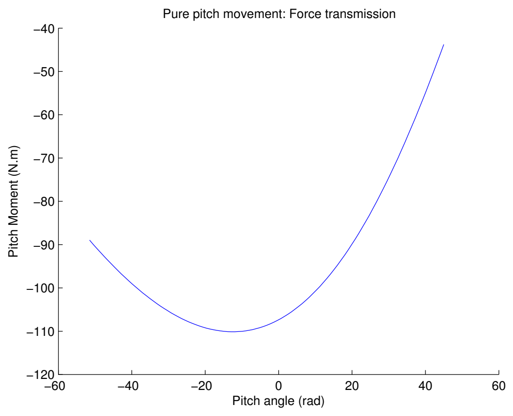
Pure Pitch Movement It is trivial to compute the pure pitch torque, as we know that for pure pitch movement ($\theta = 0$), the movement required in the linear actuators is identical i.e. $f_1 = f_2$. To compute the maximum pitch moment, we vary the pitch angle ϕ in $[-\pi, \pi]$ in the following formula:

$$\boldsymbol{f}_{max} = \boldsymbol{J}^{-T}\boldsymbol{\tau}_{max} \tag{4.10}$$

where $\vec{\tau}_{max} = \begin{bmatrix} f_{max} & f_{max} \end{bmatrix}$ Fig. 4.12a and Fig. 4.12b show the pure pitch velocity transmission in complete and feasible working range of the mechanism respectively. The points where the pure pitch moment becomes zero are the actuation singularities. It can also be noticed that the maximum pitch torque of 110 N m is available when the pitch angle is between $-6°$ and $-12°$ which is the main motivation during the design process.



(a) Full working range



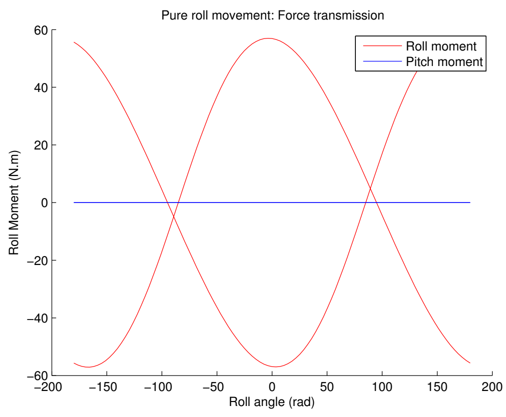
(b) Feasible working range

Figure 4.12: Force transmission in pure pitch movement

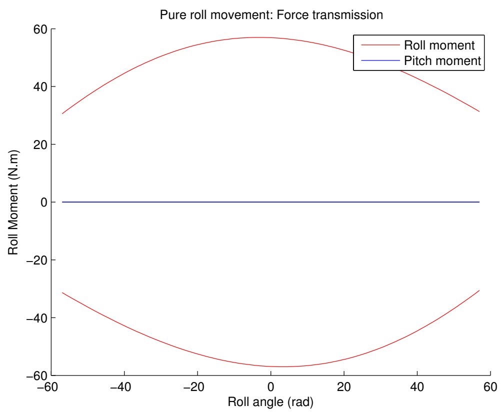
Pure Roll Movement For pure roll movement ($\phi = 0$), again such an analysis is not so straightforward. However, the ratio of f_1 and f_2 can be computed with the help of forward kinematic Jacobian matrix using the fact that $\tau_{pitch} = 0$ for the pure roll movement.

$$\begin{gathered} \boldsymbol{\tau} = \boldsymbol{J}^T(\theta, 0)\boldsymbol{f} \\ \begin{bmatrix} f_1 \\ f_2 \end{bmatrix} = \begin{bmatrix} b_{11} & b_{12} \\ b_{21} & b_{22} \end{bmatrix} \begin{bmatrix} \tau_{roll} \\ 0 \end{bmatrix} \\ \frac{f_1}{f_2} = \frac{b_{11}}{b_{21}} \end{gathered} \tag{4.11}$$

In Equation 4.11, one has the freedom to provide maximum force to one of the actuator and calculate the maximum possible force in the other actuator. For example, if the second actuator is chosen to work at maximum force i.e. $f_2 = f_{max}$, then the first actuator force can be calculated as $f_1 = \frac{b_{11}}{b_{21}} f_{max}$. Once the two actuator forces are known, one can substitute them in Equation 4.10 to compute the maximum roll moment. Fig. 4.13 shows the roll and pitch moment components of $\boldsymbol{\tau}_{max}$ in complete and feasible working range of the mechanism. The two set of curves in Fig. 4.13a and Fig. 4.13b show which actuator was chosen to operate at the maximum force. The points where roll moment drops to zero are the singular points. As one can see, the pitch moment component is zero for a pure roll movement. Fig. 4.13b shows the maximum roll moment in the feasible working range of the mechanism.



(a) Full working range



(b) Feasible working range

Figure 4.13: Force transmission in pure roll movement

Again, in both cases, it can be noticed that the maximum output force is not constant and depends on the configuration of the mechanism. The range of the maximum output force for pure pitch and pure roll movements has been documented in Table 4.3. The proposed ankle design provides good force and velocity transmission along pure pitch and roll movements which are highly desired in modern humanoids.

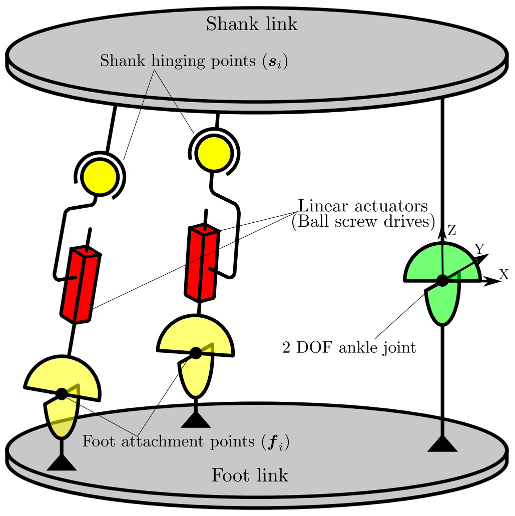

Shank link
Shank hinging points (s_i)
Linear actuators (Ball screw drives)
Z
Y
X
2 DOF ankle joint
Foot attachment points (f_i)
Foot link

Figure 4.14: Manipulator architecture of 2SPU+1U ankle mechanism

i	s_i	f_i^E
1	$\begin{pmatrix} -22.30 \\ 25 \\ 291.27 \end{pmatrix}$	$\begin{pmatrix} -70 \\ 40 \\ 0 \end{pmatrix}$
2	$\begin{pmatrix} -22.30 \\ -25 \\ 291.27 \end{pmatrix}$	$\begin{pmatrix} -70 \\ -40 \\ 0 \end{pmatrix}$

Table 4.4: Geometric dimensions (in mm) of the 2SPU+1U ankle mechanism

4.4.5 Comparison Between 2SPRR+1U and 2SPU+1U Designs

2SPU+1U mechanism is a special case of 2SPRR+1U mechanism with intersecting revolute joint axes. The manipulator architecture applied to an ankle design is shown in Fig. 4.14. The analysis presented in this chapter can be easily applied to 2SPU+1U mechanism by substituting $r = 0$. To make a comparison between the 2SPRR+1U and 2SPU+1U designs, we perform the kinematic analysis of 2SPU+1U mechanism based on the design parameters provided in Table 4.4. Same attachment points are used for this ankle design to compare against the ankle based on 2SPRR+1U architecture (compare with Table 4.1). For example, Fig. 4.15 shows the configuration space and orientation workspace of this mechanism. In comparison to the workspace of 2SPRR+1U mechanism as shown in Fig. 4.6, it can be noticed that 2SPU+1U mechanism has a poor orientation workspace especially for pure roll movements. Hence, the 2SPRR+1U architecture is the preferred solution for the ankle design since it provides the ideal force transmission without compromising on the ankle workspace. Fig. 4.16 shows the singularity curve and inverse of cond($\boldsymbol{J}$) over workspace for 2SPU+1U mechanism.

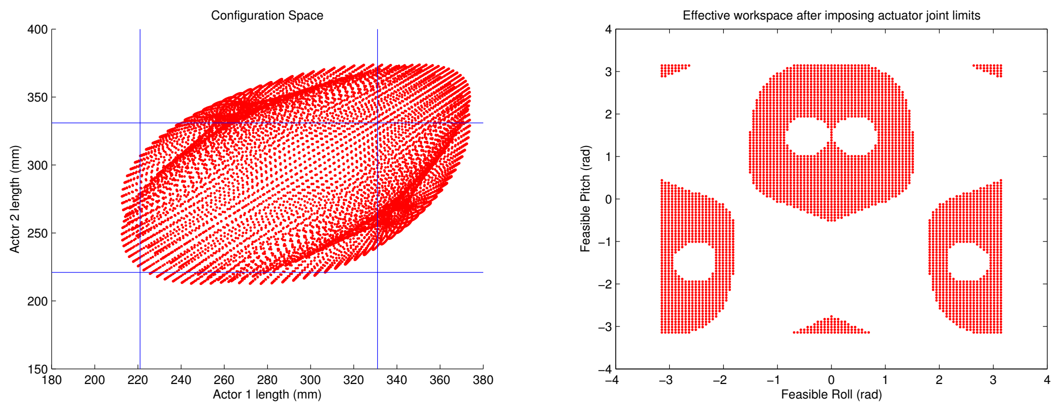


Figure 4.15: Configuration space and orientation workspace for ankle design based on 2SPU+1U mechanism

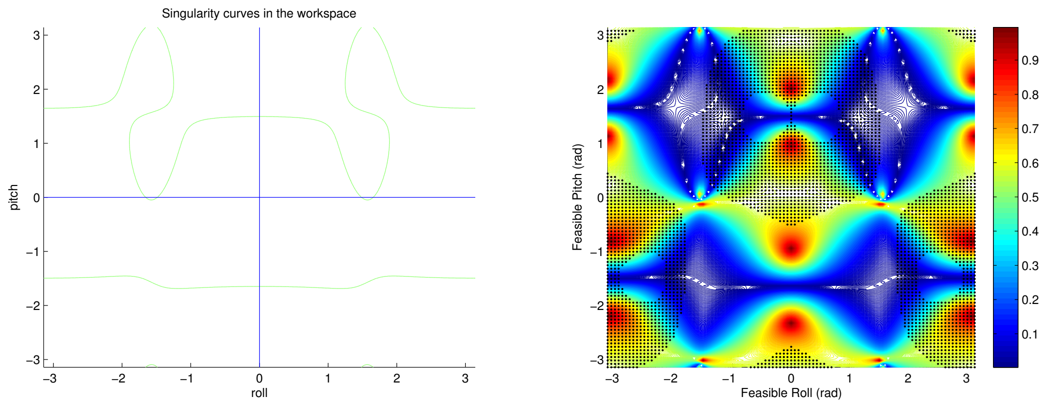


Figure 4.16: Singularity curve and inverse of condition number over workspace for ankle design based on 2SPU+1U mechanism

4.5 Application in Ankle, Wrist and Torso Design of RH5 Humanoid

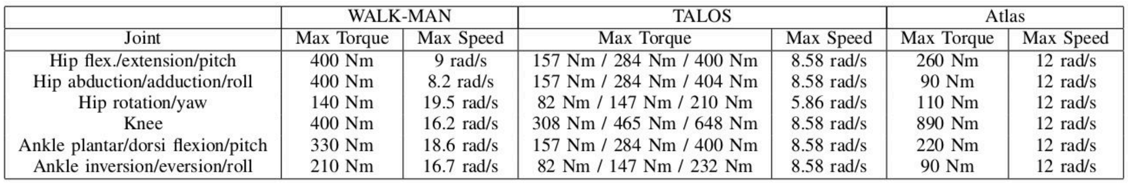
The 2SPRR+1U mechanism and its derivative 2SPU+1U mechanism has been used as a 2 DOF joint module in the design of RH5 humanoid (see Fig. 4.1). In particular, they have been used to construct the ankle, torso and wrist joints in the humanoid. In this section, the kinematic analysis performed above is applied to an optimized ankle, torso and wrist designs for the humanoid robot.

4.5.1 Ankle Design

Despite good kinematic performance of the 2SPRR+1U topology, it was realised that a poor choice of linear actuators led to insufficient velocity and moment available at the ankle joint. Hence, the design was upgraded with new faster and stronger linear

	Range (min)	Range (max)	Max. abs. torque	Max. abs. velocity
Ankle pitch	-45°	45°	130 N m	$176° \mathrm{s}^{-1}$
Ankle roll	-19.5°	19.5°	40 N m	$120° \mathrm{s}^{-1}$

Table 4.5: TORO ankle joint specification [Englsberger et al., 2014]



	WALK-MAN		TALOS		Atlas	
Joint	Max Torque	Max Speed	Max Torque	Max Speed	Max Torque	Max Speed
Hip flex./extension/pitch	400 Nm	9 rad/s	157 Nm / 284 Nm / 400 Nm	8.58 rad/s	260 Nm	12 rad/s
Hip abduction/adduction/roll	400 Nm	8.2 rad/s	157 Nm / 284 Nm / 404 Nm	8.58 rad/s	90 Nm	12 rad/s
Hip rotation/yaw	140 Nm	19.5 rad/s	82 Nm / 147 Nm / 210 Nm	5.86 rad/s	110 Nm	12 rad/s
Knee	400 Nm	16.2 rad/s	308 Nm / 465 Nm / 648 Nm	8.58 rad/s	890 Nm	12 rad/s
Ankle plantar/dorsi flexion/pitch	330 Nm	18.6 rad/s	157 Nm / 284 Nm / 400 Nm	8.58 rad/s	220 Nm	12 rad/s
Ankle inversion/eversion/roll	210 Nm	16.7 rad/s	82 Nm / 147 Nm / 232 Nm	8.58 rad/s	90 Nm	12 rad/s

Figure 4.17: Ankle joint specification from different robots [Stasse et al., 2017]

actuators. The attachment points and the geometric dimensions didn't change during the design upgrade. Table 4.6 presents the joint specification for the upgraded version of ankle joint in RH5 humanoid. Since, there is no change in geometric dimensions, the configuration space and workspace of the mechanism remains the same as Fig. 4.6. Moreover, the singularity curve and conditioning of the workspace is also not affected (same as Fig. 4.7 and Fig. 4.8).

Comparision with other humanoid ankle designs TORO is a torque controlled humanoid developed by DLR [Englsberger et al., 2014]. The pitch actuator (ILM85 weighing $2.062\mathrm{kg}$) is placed closer to the knee joint with the help of a parallel 4 bar mechanism. The roll actuator (ILM50 weighing $0.832\mathrm{kg}$) is then connected in series with the pitch guiding mechanism. The velocity and torque specification of the TORO ankle is listed in Table 4.5. In comparison to TORO ankle, the ankle joint presented in this paper provides better range of motion and force/velocity transmission capabilities while promising a light weight design and low leg inertia. The total weight of lower leg of RH5 humanoid is $3.6\mathrm{kg}$ including the foot unit, all the sensors, and electronics while the lower leg in TORO weighs $7.648\mathrm{kg}$. The actuator only weight of our mechanism is only $1.22\mathrm{kg}$ while the actuator only weight of TORO ankle is $2.894\mathrm{kg}$. The new specification outperforms TORO ankle in every aspect and the data is comparable to the most dynamic humanoid ATLAS from Boston Dynamics (Fig. 4.17).

4.5.2 Torso Design

A variant of 2SPRR+1U linkage with $r = 0$ has been used to construct the torso joint in RH5 humanoid. Its CAD rendering is shown in Fig. 4.18. The physical dimensions of the 2SPU+1U mechanism applied to a torso joint is provided in Table 4.7 based on

Range (min. to max.)	Position	Max. abs. force	Max. abs. velocity
Ankle pitch	$-51.5°$ to $45°$	$121\ \mathrm{N\,m}$ to $304\ \mathrm{N\,m}$	$200\ °\,\mathrm{s}^{-1}$ to $502°\,\mathrm{s}^{-1}$
Ankle roll	$-57°$ to $57°$	$84\ \mathrm{N\,m}$ to $158\ \mathrm{N\,m}$	$386\ °\,\mathrm{s}^{-1}$ to $726°\,\mathrm{s}^{-1}$
Linear actuator	$221\ \mathrm{mm}$ to $331\ \mathrm{mm}$	$2000\ \mathrm{N}$	$265\mathrm{mm\,s}^{-1}$

Table 4.6: RH5 Ankle v2.0 joint specification (total weight of lower leg = $3.6\ \mathrm{kg}$, weight of one actuator = $0.610\ \mathrm{kg}$)

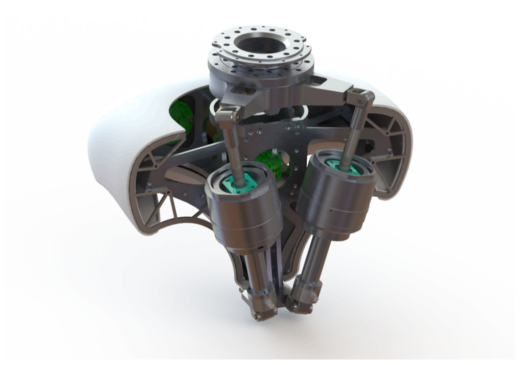


Figure 4.18: CAD prototype of Torso joint
(Credits: Heiner Peters, DFKI GmbH)

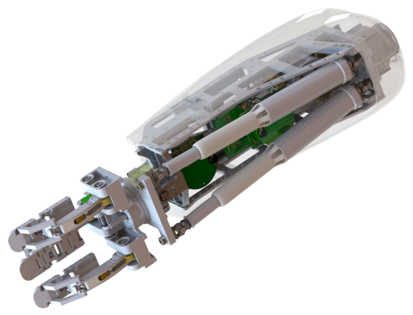


Figure 4.19: CAD prototype of Wrist joint
(Credits: Heiner Peters, DFKI GmbH)

i	$\boldsymbol{s}_i$	$\boldsymbol{f}_i^E$
1	$(-80.49, 20, -210)^T$	$(-91.22, 76.81, 23)^T$
2	$(-80.49, -20, -210)^T$	$(-91.22, -76.81, 23)^T$

Table 4.7: Geometric dimensions (in mm) of the torso mechanism

Range (min. to max.)	Position	Max. abs. force	Max. abs. velocity
Torso pitch	$-25°$ to $29°$	$380\ \mathrm{N\,m}$ to $493\ \mathrm{N\,m}$	$184\ °\,\mathrm{s}^{-1}$ to $238°\,\mathrm{s}^{-1}$
Torso roll	$-36°$ to $36°$	$285\ \mathrm{N\,m}$ to $386\ \mathrm{N\,m}$	$208\ °\,\mathrm{s}^{-1}$ to $400°\,\mathrm{s}^{-1}$
Linear actuator	$195\ \mathrm{mm}$ to $284\ \mathrm{mm}$	$2716\ \mathrm{N}$	$291\mathrm{mm\,s}^{-1}$

Table 4.8: RH5 torso joint specification

Fig. 4.20. Fig. 4.21 shows the configuration space and workspace of the torso taking into account the physical limits of the actuators. Fig. 4.22 shows the singularity curve and inverse of cond($\boldsymbol{J}$) over workspace for 2SPU+1U mechanism. Table 4.8 presents the overall joint specification of the torso joint in RH5 humanoid.

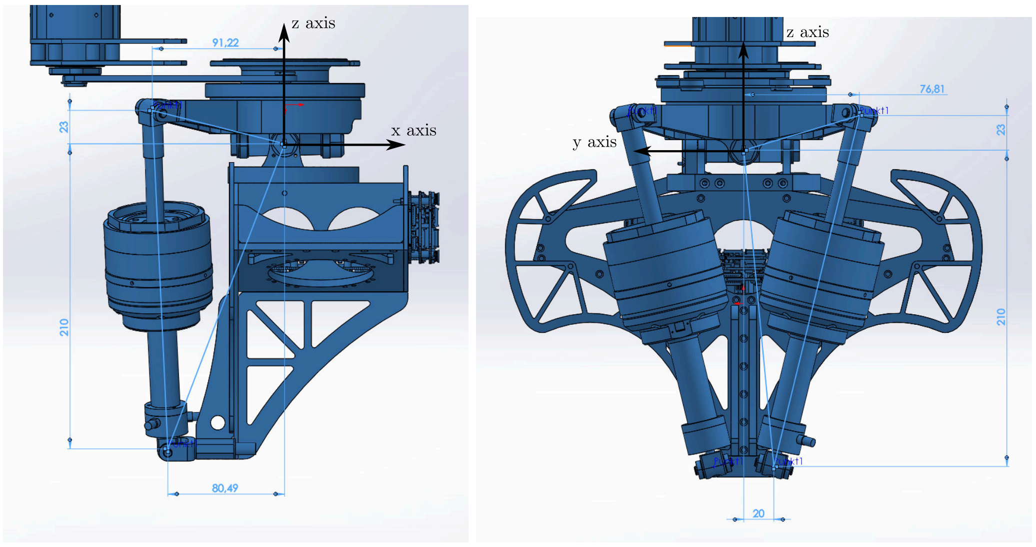


Figure 4.20: Geometric dimensions of RH5 torso joint

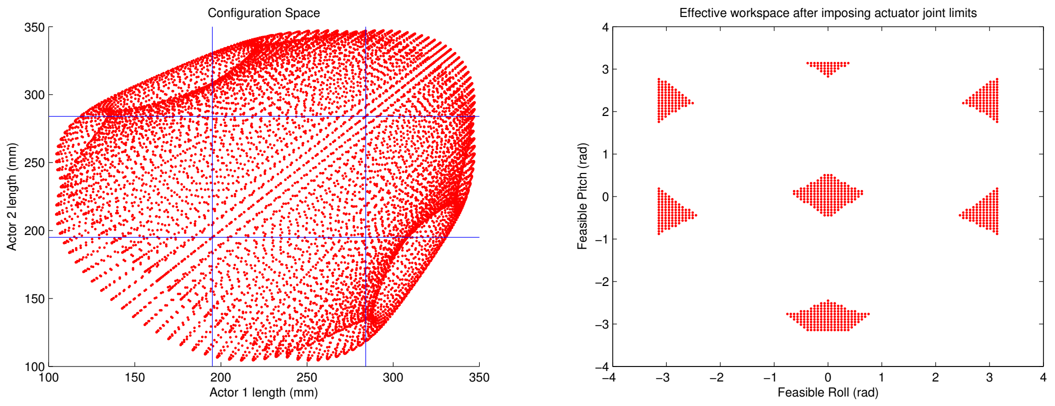


Figure 4.21: Configuration space and orientation workspace for 2SPU+1U mechanism applied to RH5 torso

4.5.3 Wrist Design

The same variant has also been used to construct the wrist joint in RH5 humanoid (see Fig. 4.19 for its CAD rendering). The physical dimensions of the 2SPU+1U mechanism applied to a wrist joint is provided in Table 4.9 based on Fig. 4.23. Fig. 4.24 shows the configuration space and workspace of the torso taking into account the physical limits of the actuators. Fig. 4.25 shows the singularity curve and inverse of cond($\boldsymbol{J}$) over workspace for 2SPU+1U mechanism. Table 4.10 presents the overall joint specification of the wrist joint in RH5 humanoid.

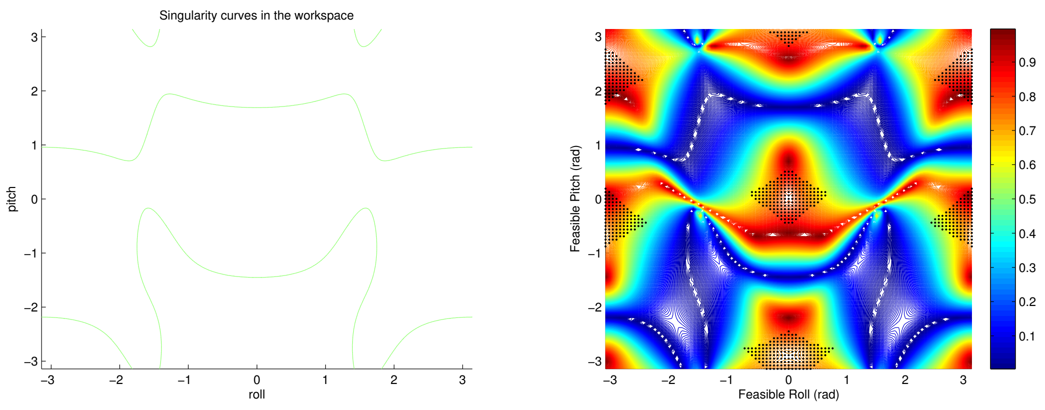


Figure 4.22: Singularity curve and inverse of condition number over workspace for 2SPU+1U mechanism applied to RH5 torso joint

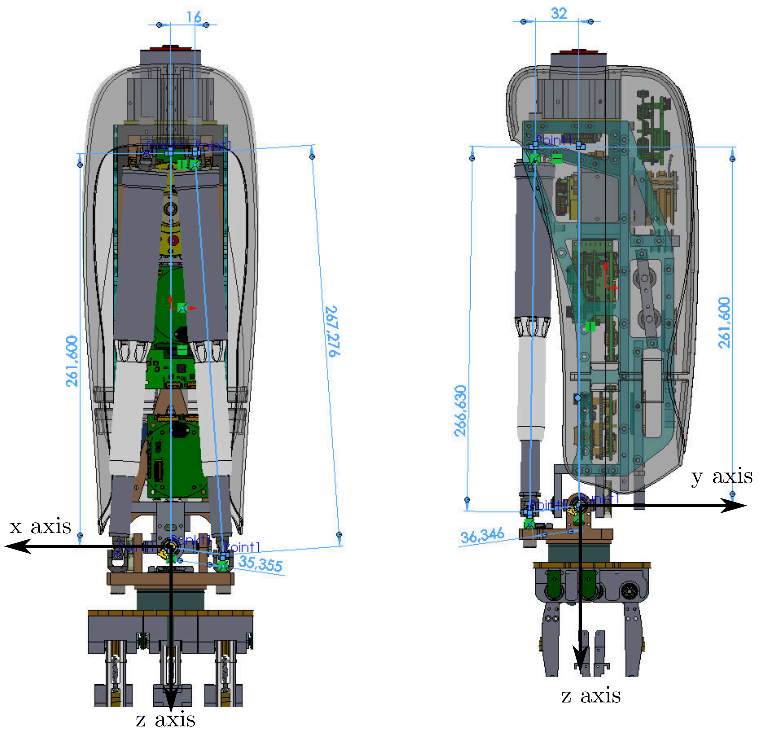
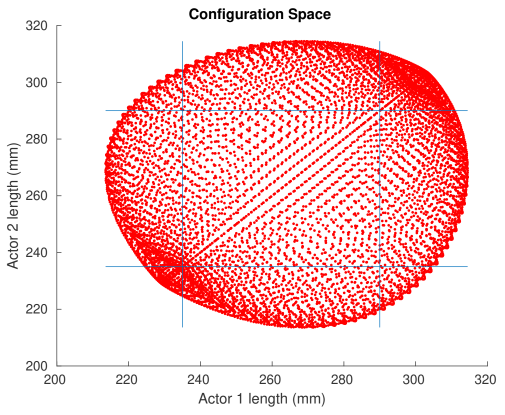


Figure 4.23: Geometric dimensions of RH5 wrist joint

i	$\boldsymbol{s}_i$	$\boldsymbol{f}_i^E$
1	$(16, -32, -261.60)^T$	$(36, -35, 5)^T$
2	$(-16, -32, -261.60)^T$	$(-36, -35, 5)^T$

Table 4.9: Geometric dimensions of the wrist mechanism



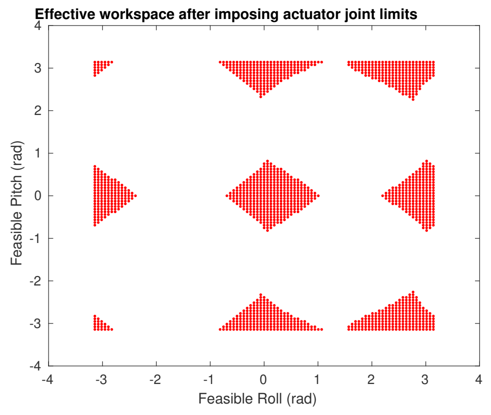
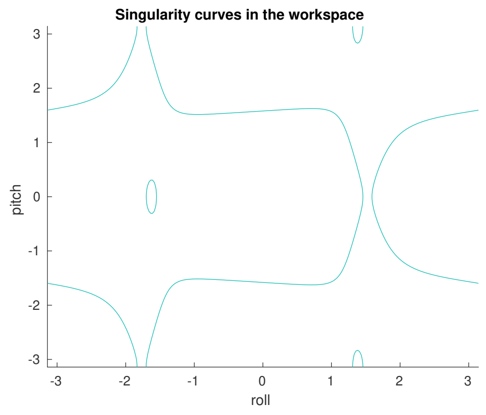


Figure 4.24: Configuration space and orientation workspace for 2SPU+1U mechanism applied to RH5 wrist



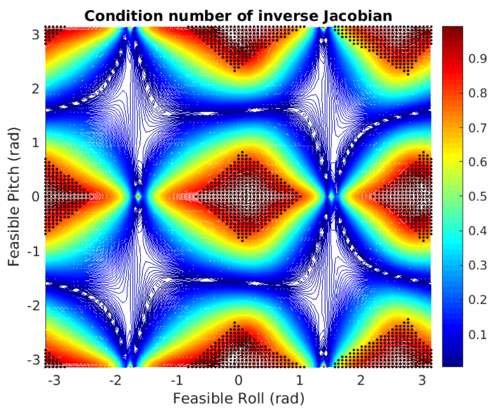


Figure 4.25: Singularity curve and inverse of condition number over workspace for 2SPU+1U mechanism applied to RH5 wrist joint

Range (min. to max.)	Position	Max. abs. force	Max. abs. velocity
Wrist pitch	$-46.8°$ to $46.8°$	24 N m to 35 N m	$7\,°\,s^{-1}$ to $14°\,s^{-1}$
Wrist roll	$-39.6°$ to $57.6°$	22 N m to 35 N m	$8\,°\,s^{-1}$ to $13°\,s^{-1}$
Linear actuator	235 mm to 290 mm	495 N	$5\mathrm{mm\,s}^{-1}$

Table 4.10: RH5 wrist joint specification

4.6 Conclusion

This chapter presents a comprehensive kinematic analysis of a novel 2SPRR+1U parallel mechanism for application as an abstraction to universal joint with two degrees of freedom. Using tools from computational algebraic geometry, an upper bound to the number of solutions to the direct kinematics problem and the real assembly modes have been studied. Inverse kinematics is used to study the mechanism's workspace, compute the singularity curves and quality of velocity and force transmission. The analysis presented in this work can be easily applied to 2SPU+1U mechanism by substituting $r = 0$. From the study, it is clear that these architectures are highly suitable to be used as two DOF universal joint modules for constructing ankle, torso and wrist subsystems of a humanoid robot. The work presented in this chapter can be used to derive the loop closure function (which will be introduced in Section 7.1.3 of Chapter 7) of the mechanism which will be integrated in the HyRoDyn software framework (described in Section 8.4.1 of Chapter 8) as a standalone submechanism library.

Chapter 5

Study of 3R-[2SS] Device for Abstraction of Spherical Joint

This chapter presents the study of the novel 3R-[2SS] mechanism also called as Active Ankle which has been used for the abstraction of a spherical joint in the design of Recupera Reha exoskeleton (see Fig. 5.1). The chapter is organized as follows: In Section 5.2, the design and the construction of the Active Ankle are reflected in comparison to the state-of-the-art and its general mobility is determined. In Section 5.4, the inverse kinematic problems and solution methods suitable for its kinematic control are presented. In Section 5.5, a forward kinematic analysis based on the tools from computational algebraic geometry is presented which provides some global insights into mechanism's geometry. The Section 5.6 presents the control of the Active Ankle in comparison to the range of motion of the human ankle. Finally, conclusions are drawn in Section 5.7. The content of this chapter is based on [Kumar et al., 2018a] and [Kumar et al., 2018b].

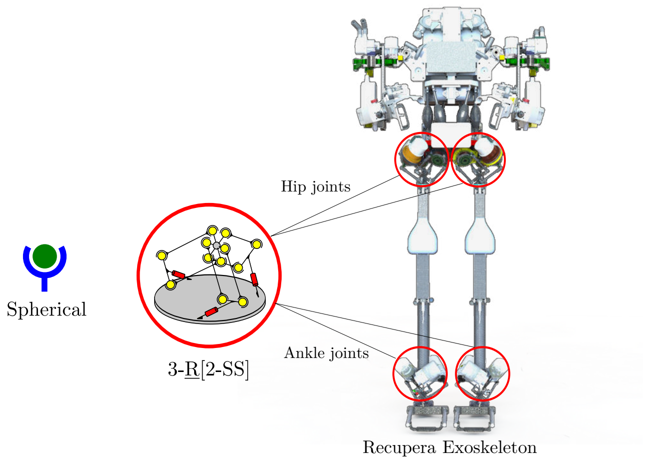


Figure 5.1: Spherical joint abstraction in Recupera exoskeleton at DFKI-RIC

5.1 Introduction

If the location of the end-effector of a PM remains constant, the device is called a spherical parallel manipulator (SPM). The AGILE EYE [Gosselin et al., 1996] and its improved variant AGILE WRIST [Niyetkaliyev and Shintemirov, 2014] are prominent examples of SPMs with three degrees of freedom (DOF). The joint axes of this class of spherical manipulators are required to intersect in a single point. However, due to machining and assembling errors, it is difficult to achieve an accurate intersection of all joint axes [Gallardo-Alvarado, 2016]. Misalignments may lead to undesirable reaction forces in the structure, and hence to a reduced service life of the mechanism or sometimes makes the complete system difficult to assemble [Vischer and Clavel, 2000]. Moreover, the use of C-shaped links in the system prevents the design from being used in high payload applications. Due to the kinematic layout that requires an exact intersection of all rotation axes, a high-precision manufacturing is indispensable for these SPMs [Al-Widyan et al., 2011]. The ARGOS mechanism, an SPM with three DOF, was developed by Vischer and Clavel [Vischer and Clavel, 2000] to overcome these shortcomings. Their $3[\,R\,[RR/SS]\,S\,]$-design consists of three identical legs containing a revolute joint at the base whose axis is pointing to a virtual rotation center.

A novel, almost-spherical parallel manipulator (ASPM) ACTIVE ANKLE (Fig. 1.3a) has recently been introduced in [Simnofske, 2015] and [Simnofske et al., 2016]. Due to its unique, simple and compact $3[R\,2\,[SS]]$ design, the constraint of moving the end-effector about an exact center (of rotation) in case of spherical parallel manipulators (SPM) is relaxed to almost spherical motions that includes a shift of the end effector about a tolerated, very small domain. Due to the presence of a closed loop in each leg, the mechanism offers high stiffness and orientation accuracy. The mechanism features a low link diversity and its simple, robust and modular design makes it highly suitable for many applications. While the primary application of the ACTIVE ANKLE is an active ankle joint in an exoskeleton or a humanoid, it could also be integrated as a submechanism into a regional manipulator for obtaining precise six DOF motions if the constrained translations of the ASPM are compensated by the previous joints of the overall device.

5.2 Mechanism's Design Description

5.2.1 Type Synthesis

The geometric type of the spatial almost-spherical parallel mechanism ACTIVE ANKLE are set into context in Table 5.1 and Table 5.2. The various possible leg config-

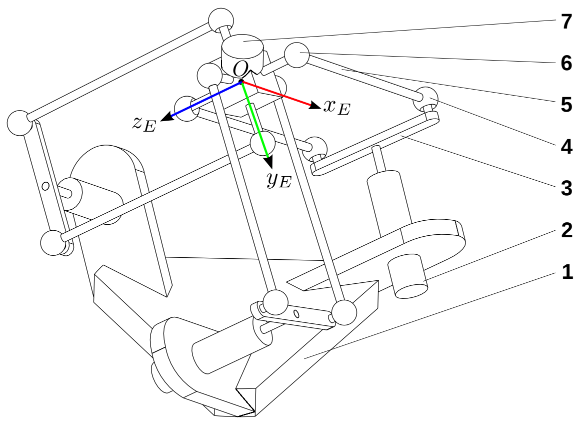


Figure 5.2: Sketch of the ACTIVE ANKLE [Simnofske, 2015] including (1) base, (2) rotative actuator, (3) crank, (4 & 6) ball and socket joints, (5) rod, (7) end-effector.

urations can be derived using the Kutzbach-Gruebler formula. Table 5.1 describes the possibilities by using the relation between the desired degree of freedom of the parallel manipulator d, the numberof kinematic chains k, and the sum of the joint DOF ofeach chain f. Each kinematic leg can be realized by a serial arrangement of links and joints or with closed loops. The latter comes with an inherent advantage of increased stiffness. For example, in the famous DELTA robot which has 3 DOF, each of its three legs is realized by a closed parallelogram (4S) mechanism which makes it a stiff positioning system. This is an inspiration for finding a novel parallel manipulator which can produce spherical movements while still keeping the topological arrangement of DELTA robot. With a homogeneous distribution of five DOF to all three legs (Table 5.1), the type of the ACTIVE ANKLE matches those of the DELTA robot. The topological setup of both mechanisms also equals on the level of each of the three identical legs. Both consist of one rotative actuator in series with one closed loop with four spherical joints (Fig. 5.3). For these reasons, the ACTIVE ANKLE can be classified as the (almost) rotative counterpart of the DELTA robot. In comparison to the DELTA robot, which provides a stiff positioning functionality, the ACTIVE ANKLE provides a stiff orientating feature, due to the employment of parallel structures within the three kinematic chains.

5.2.2 Design and Construction

The mechanical layout of ACTIVE ANKLE is modular and depicted in Fig. 5.2: the device features three rotative actuators fixed to the base. Each of the three motors drives a spatial quadrilateral consisting of a symmetric crank, two rods, and a line segment on the mobile platform. The three line segments mutually intersect orthogonally and together form a spatial cross on the end-effector link. The total weight of

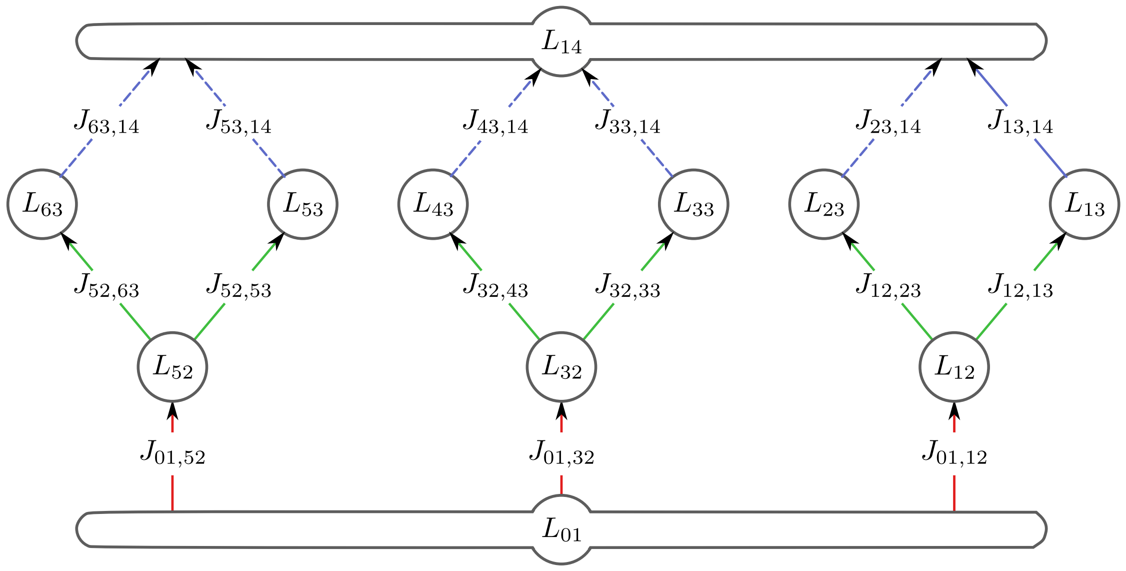


Figure 5.3: Link graph of the parallel manipulator ACTIVE ANKLE, including $n = 11$ links and $m = 15$ joints.

the mechanism including the three actuators is 1.8 Kg. With regard to the electronics, the device features three actuator modules which include a brushless DC motor coupled with harmonic gear drives (nominal torque 28 Nm, weight 0.392 Kg), FPGA based control, and power electronics. Each actuator module is capable of a cascaded position, velocity, and current based torque control [Simnofske et al., 2016]. The presented prototype of this mechanism is designed to carry static loads upto 30 Kg in the zero configuration.

5.2.3 Topology and General Mobility

The topology of the mechanism is equivalent to DELTA robot as depicted in Fig. 5.3. The $n = 11$ links L_i are enumerated as L_{01}, L_{12}, L_{13}, L_{14}, L_{23}, L_{32}, L_{33}, L_{43}, L_{52}, L_{53}, and L_{63}. The $m = 15$ joints $J_{i,j}$ are distinguished using double indices, as indicated in Fig. 5.3. The number of independent loops of the ACTIVE ANKLE is computed with $c = m - n + 1 = 15 - 11 + 1 = 5$. The general mobility of the mechanism can be estimated by means of the Kutzbach-Grübler formula: $d_s(\mathcal{M}) = s(n - m - 1) + f = s(-c) + f$, where the total number of freedoms $f = \sum_{ij} f_{ij}$ is determined by considering three rotative joints, six spherical joints, and six universal joints, which results in $f = 3 \cdot 1 + 6 \cdot 3 + 6 \cdot 2 = 3 + 18 + 12 = 33$. Since the device is *almost* spherical, the motion parameter $s = 6$ (spatial) and $s = 3$ (spherical). Hence, the mobility of the device can be computed as: $d_s(\mathcal{M}) = 6 \cdot (11 - 15 - 1) + 33 = 3$.

Table 5.1: Overview of spatial parallel manipulators with general mobility d with distributions of degrees of freedom to k kinematic chains (legs), in accordance to [Frindt, 2001].

	$d = 2$	$d = 3$	$d = 4$	$d = 5$	$d = 6$
$k = 2$	②⑥ ③⑤ ④④	③⑥ ④⑤	④⑥ ⑤⑤	⑤⑥	⑥⑥
$k = 3$	–	③⑥⑥ ④⑤⑥ ⑤⑤⑤	④⑥⑥ ⑤⑤⑥	⑤⑥⑥	⑥⑥⑥
$k = 4$	–	–	④⑥⑥⑥ ⑤⑤⑥⑥	⑤⑥⑥⑥	⑥⑥⑥⑥
$k = 5$	–	–	–	⑤⑥⑥⑥⑥	⑥⑥⑥⑥⑥
$k = 6$	–	–	–	–	⑥⑥⑥⑥⑥⑥

Table 5.2: Examples of mechanisms with respect to type and mobility. *Watt's and Chebyshev's linkages are almost prismatic [Kempe, 1877].

Mechanism type			General mobility d		
Motion	Group	Dim.	1	3	6
Position	P^2	2	Peaucellier–Lipkin*	–	–
Flat	P^2R	3	Planar 4R	Planar Stewart	–
Spherical	R^3	3	Spherical 4R	Agile Eye, Argos	–
Position	P^3	3	Sarrus	Delta robot	–
Spatial	P^3R^3	6	Bennett 4R	**Active Ankle**	Stewart

5.2.4 Design Features

The mechanism's homogeneous and simple design leads to a low link diversity, permits a low-cost construction, and provides robustness against production inaccuracies. A crucial feature of the mechanism's design is the stress distribution among the structure. The six rods that transmit the forces from the cranks to the platform are only loaded with forces along their axes due to the spherical joints attached to them. Moreover, any force applied along the direction of its platform's torsional axis can be supported without an active torque in the motors.

A multibody dynamics simulation analysis and a subsequent FEM analysis have been performed to check the deformation of the critical parts as rods and cranks under desired loads (Fig. 5.4). A force corresponding to the weight of the exoskeleton is applied to the end effector and the forces in the spherical joints are measured. In the

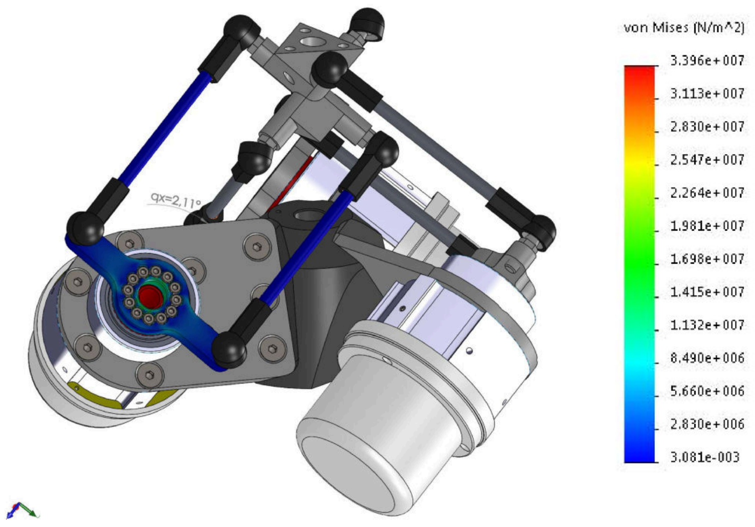


Figure 5.4: FEM analysis of the ACTIVE ANKLE.

zero configuration, this force – equivalent to 350 N perpendicular to the end effector's top plate – leads to a reaction force of approximately 100N in each spherical joint. The selected ball and socket joints are designed for a maximum axial tensile force of 600N in housing axis and a pivot angle of maximum of $\pm 25°$. The same magnitude of force occurs in the rods and this force has been found to be less than the buckling force of the rods (i.e. 2120N). Thus, it is ensured that the mechanism resists from buckling in all possible configurations [Simnofske et al., 2016].

5.2.5 Design Comparison

In this section, the design of the almost-spherical mechanism Active Ankle is analyzed from a principal and from an application-motivated point of view: First, its design is compared to that of spherical mechanisms, and second, its design is set into contrast with devices intended to interoperate with the human ankle.

Spherical Mechanisms. In Table 5.3, the almost-spherical ACTIVE ANKLE is briefly compared to a set of (purely) spherical devices.[1] The RRR chain and the Cardan mechanism [Temple, 1988] with three intersecting axes represent the most simple spherical devices: due to their serial construction, they lack the stiffness that is offered by their parallel counterparts. AGILE EYE and its variants are Spherical Parallel Manipulators (SPM) which offer high speeds for low payloads. Due to their design, they require high manufacturing and assembly accuracies. The design of the Asymmetrical Spherical Parallel Manipulator ASYSPM [Wu et al., 2015]

[1]The presented comparison is an outline of a more detailed argumentation [Simnofske et al., 2016].

Mechanism	Ref.	Links n	Joints m	Loops c
RRR / Cardan	[Temple, 1988]	4	3 (6)	0 (3)
Agile Eye / Wrist	[Gosselin et al., 1996]	8	9	2
AsySPM	[Wu et al., 2015]	11	13	3
CamSPM3	[Villgrattner et al., 2011]	8	10	4
Hexasphere	[Valasek et al., 2010]	14	19	6
Active Ankle	[Simnofske, 2015]	11	15	5

Table 5.3: A comparison of mechanisms, in terms of their members, links n, joints m, and number of independent loops $c = m - n + 1$; quoted from [Simnofske et al., 2016].

involves the use of large number of different parts due to its asymmetrical leg configuration. In comparison to the Active Ankle, the 3-SPS manipulator (CamSPM3 in Table 5.3) [Villgrattner et al., 2011] follows a complementary actuation approach: prismatic, instead of revolute joints are employed to actuate the platform. The Hexasphere [Valasek et al., 2010] is a redundant SPM that features six motors to achieve the three rotative degrees of freedom of the platform.

Ankle Exoskeletons The Active Ankle is priorly designed to work as an active interface to three DOF human joints. Its application at the hip and the ankle joints within the novel full body Recupera exoskeleton [Kirchner et al., 2016, Kumar et al., 2019b]. While the exoskeleton is primarily designed for upper body rehabilitation [2] , the main purpose of the legs is to transfer the load of the upper body exoskeleton system to the ground and provide some mobility features (e.g. sitting, standing, walking etc.) to the human subject: the Recupera legs and the integrated Active Ankle instances are considered as *load transfer* devices according to the classification by Herr.[3] In contrast to the similar load carrying exoskeleton Bleex [Zoss et al., 2006] which only features four active DOF per leg, the Recupera exoskeleton provides seven active DOF in each leg, due to the role of Active Ankle as a modular spherical unit. The hydraulically-damped ankle-foot orthosis by Yamamoto et al. [Yamamoto et al., 2005] and the knee-ankle-foot exoskeleton Kafo driven by artificial pneumatic muscles [Sawicki and Ferris, 2009] both only provide a single DOF at the ankle joint of the human.

[2]The exoskeleton designs for upper body rehabilitation are usually attached to a fixed base (e.g. ARMIN [Nef et al., 2009], Recupera wheelchair system [Kumar et al., 2017b]) or to the patient's torso (e.g. RUPERT[Huang et al., 2016]) which either reduces the mobility of patients or forces the patient to carry the weight of the exoskeleton which might be difficult for weaker stroke patients. A more detailed survey of exoskeletons for upper body rehabilitation can be found in [Gopura et al., 2011]

[3]Herr [Herr, 2009] distinguishes parallel-limb exoskeletons according to their function, 'load transfer to the ground', 'torque and work augmentation', and 'increase human endurance'. Active devices are named 'exoskeletons', passive devices are named 'orthoses'.

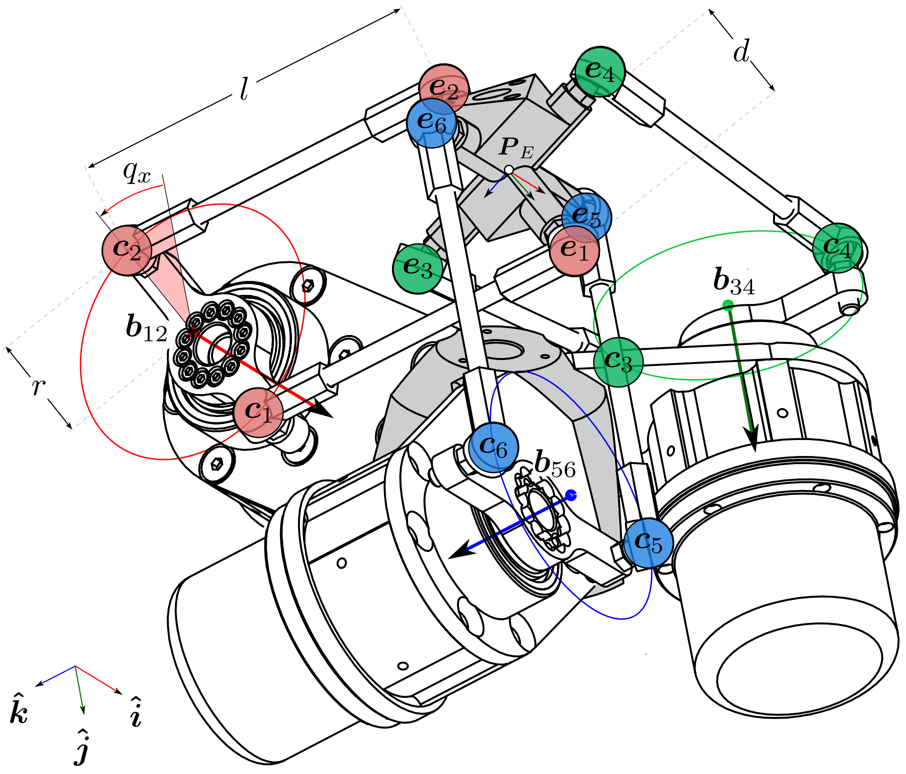


Figure 5.5: A posture of the ACTIVE ANKLE corresponding to the configuration $\boldsymbol{q} = (q_x, q_y, q_z) \approx (-25°, 0°, 0°)$. The design parameters are $d = r = 35\,\text{mm}$ and $l = 100\,\text{mm}$.

5.3 Mechanism Architecture and Constraint Equations

In this section, the parameterizations of the end effector and crank points are presented and the constraint equations of the mechanism are derived. The six points $(\boldsymbol{e}_1, \ldots, \boldsymbol{e}_6)$ on the end effector lie on a sphere with radius d. The points $\boldsymbol{c}_i$ and $\boldsymbol{c}_j$ rotate around $\boldsymbol{b}_{ij}$ in circles of radius r for $ij \in \{12, 34, 56\}$. The length of the six rods is denoted by l. The global frame O is coincident with the end effector position $\boldsymbol{e}$ when the mechanism is in its zero-configuration (Fig. 5.2). The unit vectors $\hat{\boldsymbol{s}}$, $\hat{\boldsymbol{n}}$ and $\hat{\boldsymbol{a}}$ are vectors along the x_E, y_E and z_E axes respectively expressed in the global frame O.

End effector points The points $\boldsymbol{e}_i$, $i \in \{1, .., 6\}$ are rigidly attached to the end-effector. Fig. 5.5 shows that the pair of points $(\boldsymbol{e}_1, \boldsymbol{e}_2)$ lies on a line $\mathcal{L}_{12} = (\boldsymbol{e}, \hat{\boldsymbol{n}})$ along unit vector $\hat{\boldsymbol{n}}$ passing through point $\boldsymbol{e}$. Similarly, the pairs $(\boldsymbol{e}_3, \boldsymbol{e}_4)$ and $(\boldsymbol{e}_5, \boldsymbol{e}_6)$ lie on lines $\mathcal{L}_{34} = (\boldsymbol{e}, \hat{\boldsymbol{a}})$ and $\mathcal{L}_{56} = (\boldsymbol{e}, \hat{\boldsymbol{s}})$ respectively. The coordinates of these points in terms of end effector position $(\boldsymbol{e})$ and orientation $(\hat{\boldsymbol{s}}, \hat{\boldsymbol{n}}, \hat{\boldsymbol{a}})$ are expressed as:

$$\begin{aligned} \boldsymbol{e}_1 &= \boldsymbol{e} + d\hat{\boldsymbol{n}} & \boldsymbol{e}_2 &= \boldsymbol{e} - d\hat{\boldsymbol{n}} \\ \boldsymbol{e}_3 &= \boldsymbol{e} + d\hat{\boldsymbol{a}} & \boldsymbol{e}_4 &= \boldsymbol{e} - d\hat{\boldsymbol{a}} \\ \boldsymbol{e}_5 &= \boldsymbol{e} + d\hat{\boldsymbol{s}} & \boldsymbol{e}_6 &= \boldsymbol{e} - d\hat{\boldsymbol{s}} \end{aligned} \tag{5.1}$$

The position vectors of six end effector points are stored column-wise in matrix $\boldsymbol{E} = (\boldsymbol{e}_1 \ldots \boldsymbol{e}_6)$. The parameterization of six end effector points using the end effector pose is implemented in the method Calculate End-effector Points (CEP) (Algorithm 5).

Crank points The crank points $\boldsymbol{c}_i$, $i \in \{1, .., 6\}$ are allowed to move on the circles defined by the motion of three actuators. The pair of points $(\boldsymbol{c}_i, \boldsymbol{c}_j)$ lie diametrically opposite to each other on a circle of radius r with center $\boldsymbol{b}_{ij}$, $ij \in \{12, 34, 56\}$. The position vector of six crank points are parameterized using input joint angles (q_x, q_y, q_z) with the set of equations

$$\begin{aligned} \boldsymbol{c}_1(q_x) &= \boldsymbol{b}_{12} + \boldsymbol{c}_{12}(q_x) & \boldsymbol{c}_2(q_x) &= \boldsymbol{b}_{12} - \boldsymbol{c}_{12}(q_x) \\ \boldsymbol{c}_3(q_y) &= \boldsymbol{b}_{34} + \boldsymbol{c}_{34}(q_y) & \boldsymbol{c}_4(q_y) &= \boldsymbol{b}_{34} - \boldsymbol{c}_{34}(q_y) \\ \boldsymbol{c}_5(q_z) &= \boldsymbol{b}_{56} + \boldsymbol{c}_{56}(q_z) & \boldsymbol{c}_6(q_z) &= \boldsymbol{b}_{56} - \boldsymbol{c}_{56}(q_z) \ . \end{aligned} \tag{5.2}$$

In Equation 5.2, centers $(\boldsymbol{b}_{12}, \boldsymbol{b}_{34}, \boldsymbol{b}_{56})$ lie on (yz, zx, xy) planes at a distance of l units along (z, x, y) axes respectively. The general points $(\boldsymbol{c}_{12}, \boldsymbol{c}_{34}, \boldsymbol{c}_{56})$ on these circles are described as

$$\begin{aligned} \boldsymbol{b}_{12} &= l\hat{\boldsymbol{k}} \ , & \boldsymbol{c}_{12}(q_x) &= r\cos(q_x)\hat{\boldsymbol{\jmath}} + r\sin(q_x)\hat{\boldsymbol{k}} \\ \boldsymbol{b}_{34} &= l\hat{\boldsymbol{\imath}} \ , & \boldsymbol{c}_{34}(q_y) &= r\cos(q_y)\hat{\boldsymbol{k}} + r\sin(q_y)\hat{\boldsymbol{\imath}} \\ \boldsymbol{b}_{56} &= l\hat{\boldsymbol{\jmath}} \ , & \boldsymbol{c}_{56}(q_z) &= r\cos(q_z)\hat{\boldsymbol{\imath}} + r\sin(q_z)\hat{\boldsymbol{\jmath}} \ . \end{aligned}$$

The position vectors of six crank points are stored column-wise in matrix $\boldsymbol{C} = (\boldsymbol{c}_1 \ldots \boldsymbol{c}_6)$. The parameterization of six crank points using the input joint angles is implemented in the method Calculate Crank Points (CCP) (Algorithm 6).

Kinematic constraint equations The length of the line segment joining the crank points $(\boldsymbol{c}_i)$ to the end effector points $(\boldsymbol{e}_i)$ equals the rod length l.

$$\|\boldsymbol{e}_i - \boldsymbol{c}_i\| = l \ , \qquad i \in \{1, .., 6\} \tag{5.3}$$

Type	Well-determined				Over-determined			
Direction	Name	In	Eqs	Out	Name	In	Eqs	Out
Inverse	RIKP	$\boldsymbol{R}_E$	$\overset{6}{\longmapsto}$	$(\boldsymbol{q}, \boldsymbol{e})$	IKP	$(\boldsymbol{R}_E, \boldsymbol{e})$	$\overset{6}{\longmapsto}$	$\boldsymbol{q}$
	IKP*	$(\boldsymbol{R}_E, \boldsymbol{e})$	$\overset{3}{\longmapsto}$	$\boldsymbol{q}$				
Forward	FKP	$\boldsymbol{q}$	$\overset{6}{\longmapsto}$	$(\boldsymbol{R}_E, \boldsymbol{e})$	TFKP	$(\boldsymbol{q}, \boldsymbol{R}_E)$	$\overset{6}{\longmapsto}$	$\boldsymbol{e}$
	TFKP*	$(\boldsymbol{q}, \boldsymbol{R}_E)$	$\overset{3}{\longmapsto}$	$\boldsymbol{e}$				

Table 5.4: Overview of problem characteristics, $\dim(\boldsymbol{R}_E) = 3$, $\dim(\boldsymbol{e}) = 3$, $\dim(\boldsymbol{q}) = 3$. * denotes the relaxed problem.

Substituting Equation 5.1 and Equation 5.2 in Equation 5.3, and squaring both sides, the six distance constraints equations are derived:

$$(e_x + dn_x)^2 + (e_y + dn_y - r\cos(q_x))^2 + \ (e_z + dn_z - l - r\sin(q_x))^2 = l^2 \tag{5.4}$$

$$(e_x - dn_x)^2 + (e_y - dn_y + r\cos(q_x))^2 + \ (e_z - dn_z - l + r\sin(q_x))^2 = l^2 \tag{5.5}$$

$$(e_x + da_x - l - r\sin(q_y))^2 + (e_y + da_y)^2 + \ (e_z + da_z - r\cos(q_y))^2 = l^2 \tag{5.6}$$

$$(e_x - da_x - l + r\sin(q_y))^2 + (e_y - da_y)^2 + \ (e_z - da_z + r\cos(q_y))^2 = l^2 \tag{5.7}$$

$$(e_x + ds_x - r\cos(q_z))^2 + (e_y + ds_y - l - \ r\sin(q_z))^2 + (e_z + ds_z)^2 = l^2 \tag{5.8}$$

$$(e_x - ds_x + r\cos(q_z))^2 + (e_y - ds_y - l + \ r\sin(q_z))^2 + (e_z - ds_z)^2 = l^2 \tag{5.9}$$

Problem Overview In Table 5.4, an overview of the nature of kinematics problems is presented based on the dimensionality of the input and output variables and the number of constraint equations. With regard to the dimensionality of the unknown variables and number of equations, it can be noticed that the inverse kinematics problem (IKP) for this mechanism is over-determined while the forward kinematics problem (FKP) is well determined. From the point of view of kinematic control, inverse kinematics problem in its original form is not relevant due to almost spherical nature of this mechanism. It is intended to be used as a spherical device, and hence this demands the solution to a rotative inverse kinematics problem (RIKP) which in-

Algorithm 1 Inverse kinematic model (IKM)

```
(in) Target pose $\boldsymbol{P}_e$
(out) Joint configuration $(q_x, q_y, q_z)$
 1: function IKM($\boldsymbol{P}_e$)
 2:     $(\boldsymbol{e}_1, \dots, \boldsymbol{e}_6) \leftarrow \text{CEP}(\boldsymbol{P}_e)$    ▷ Platform coords
 3:     for $ij \in \{12, 34, 56\}$ do
 4:         $\boldsymbol{p}_{i+}, \boldsymbol{p}_{i-} \leftarrow \mathcal{S}(\boldsymbol{e}_i, d_i) \cap \mathcal{C}(\boldsymbol{b}_{ij}, \frac{c}{2}, \hat{\boldsymbol{z}}_{ij})$ ▷ Sphere-circle intersections for $i$
 5:         $\boldsymbol{p}_{j+}, \boldsymbol{p}_{j-} \leftarrow \mathcal{S}(\boldsymbol{e}_j, d_j) \cap \mathcal{C}(\boldsymbol{b}_{ij}, \frac{c}{2}, \hat{\boldsymbol{z}}_{ij})$ ▷ Sphere-circle intersections for $j$
 6:         $\mathcal{I} \leftarrow \{\boldsymbol{p}_{i+}, \boldsymbol{p}_{i-}\}$, $\mathcal{J} \leftarrow \{\boldsymbol{p}_{j+}, \boldsymbol{p}_{j-}\}$
 7:         $\boldsymbol{p}_+, \boldsymbol{p}_- \leftarrow \underset{\boldsymbol{p}_i \in \mathcal{I}, \boldsymbol{p}_j \in \mathcal{J}}{\operatorname{argmax}} \left((\boldsymbol{p}_i - \boldsymbol{b}_{ij})^{\circleddash} \cdot (\boldsymbol{b}_{ij} - \boldsymbol{p}_j)^{\circleddash}\right)$ ▷ Operator $^{\circleddash} : \boldsymbol{a} \mapsto \hat{\boldsymbol{a}} = \frac{\boldsymbol{a}}{\|\boldsymbol{a}\|}$
 8:         $\boldsymbol{r}_{ij} \leftarrow \boldsymbol{p}_+ - \boldsymbol{p}_-$    ▷ Antipodes
 9:         $\boldsymbol{d}_{ij} \leftarrow \boldsymbol{c}_i^{(0)} - \boldsymbol{c}_j^{(0)}$    ▷ Zero Posture
10:     $q_x, q_y, q_z \leftarrow \angle(\boldsymbol{d}_{12}, \boldsymbol{r}_{12}), \angle(\boldsymbol{d}_{34}, \boldsymbol{r}_{34}), \angle(\boldsymbol{d}_{56}, \boldsymbol{r}_{56})$
11:     return $(q_x, q_y, q_z)$
```

volves finding a joint configuration from a given platform orientation in $SO(3)$ instead of a given platform pose in $SE(3)$. This problem is again well-determined.

5.4 Inverse Kinematics

In this section, the inverse and rotational inverse kinematics problems are presented along with their solution methods.

5.4.1 Inverse Kinematics

Problem 1 (Inverse Kinematics) *The Inverse Kinematics Problem (*IKP*) is defined as the problem of finding the input joint angles needed to achieve a specific pose of the end effector [Khalil and Dombre, 2002a], formally,*

$$[q_x, q_y, q_z] = \text{IKP}(\boldsymbol{P}_E), \quad \boldsymbol{P}_E \in SE(3) ,$$

where $\boldsymbol{P}_E$ is the homogeneous transformation matrix of the end effector E with respect to the global frame O and $[q_x, q_y, q_z]$ are the active revloute joint angles.

$$\boldsymbol{P}_E = \begin{bmatrix} s_x & n_x & a_x & e_x \\ s_y & n_y & a_y & e_y \\ s_z & n_z & a_z & e_z \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

Algorithm 2 Matrix minor (MINOR)

```
(in) Matrix $A \in \mathbb{R}^{m\times n}$, row indices $R = (r_1, r_2, \dots, r_p)$, column indices $C = (c_1, c_2, \dots, c_q)$, $x \in \{0, 1\}$
(out) Determinant of the submatrix $A_{[R][C]} \in \mathbb{R}^{p\times q}$
1: function MINOR($A$, $R$, $C$, $x$)
2:     if $x = 0$ then                          ▷ Index handling
3:         do $r_i \leftarrow r_i + 1$ for $r_i \in R$
4:         do $c_j \leftarrow c_j + 1$ for $c_j \in C$
5:     $A_{[R][C]} \leftarrow$ EXTRACT($A, R, C$)        ▷ Submatrix
6:     $m \leftarrow \det(A_{[R][C]})$                 ▷ Minor
7:     return $m$
```

Algorithm 3 Submatrix extraction (EXTRACT)

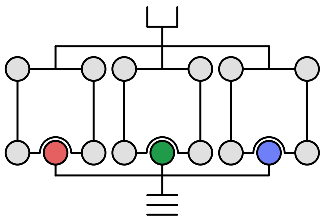
```
(in) Matrix $A \in \mathbb{R}^{m\times n}$, row indices $R = (r_1, r_2, \dots, r_p)$ with $1 \le r_i \le m$ for $1 \le i \le p$, and column indices $C = (c_1, c_2, \dots, c_q)$ with $1 \le r_j \le n$ for $1 \le j \le q$
(out) Submatrix $A_{[R][C]} \in \mathbb{R}^{p\times q}$ extracted by $R$ and $C$
1: function EXTRACT($A$, $R$, $C$)
2:     $R \leftarrow ({}^{m}\hat{e}_{r_1}\ {}^{m}\hat{e}_{r_2}\ \dots\ {}^{m}\hat{e}_{r_p})^T$      ▷ $R \in \mathbb{R}^{p\times m}$
3:     $C \leftarrow ({}^{n}\hat{e}_{c_1}\ {}^{n}\hat{e}_{c_2}\ \dots\ {}^{n}\hat{e}_{c_q})$      ▷ $C \in \mathbb{R}^{n\times q}$
4:     $A_{[R][C]} \leftarrow RAC$
5:     return $A_{[R][C]}$
```

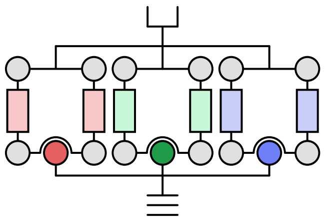
As noted in Table 5.4, in context of the IKP, the system of non-linear equations is overdetermined as the number of unknowns (three) is less than the number of equations (six).

The method IKM in Algorithm 1 provides an analytical solution method to IKP. The computation of the intersections of sphere and circle in Line 4 and Line 5 of Algorithm 1 can be conducted by means of the intersection method SPHINT (Algorithm 4) for three spheres.[4] In Line 7, a pair of antipodal points is selected from the set of four intersection points by maximizing the cosine similarity between two normalized difference vectors. The line segment between the selected two points represents the current alignment of the rod. The angle between the current alignment and the zero reference alignment (Line 9) of one rod yields the angle of one input joint, determined in Line 10 of Algorithm 1.

Since the IKM solution depends on the knowledge of the end effector shift (e_x, e_y, e_z), it is not sufficient for achieving a kinematic control of the mechanism in spherical task space $SO(3)$. Therefore it is required to calculate the input joint angles only from the desired orientation of the end effector.

[4]For a given a circle $\mathcal{C}(\boldsymbol{m}_C, r_C, \hat{\boldsymbol{n}}_C)$ with midpoint $\boldsymbol{m}_C$, radius r_C, and unit normal $\hat{\boldsymbol{n}}_C$, two substituting spheres $\mathcal{S}_A(\boldsymbol{m}_A, r_A)$ and $\mathcal{S}_B(\boldsymbol{m}_B, r_B)$ are given by the midpoints $\boldsymbol{m}_A, \boldsymbol{m}_B = \boldsymbol{m}_C \pm \frac{4}{3} r_C \hat{\boldsymbol{n}}_C$ and the radii $r_A = r_B = \frac{5}{3} r_C$.


(a) Physical setup of the Active Ankle.


(b) Relaxed structure for IKM* (Algorithm 7).

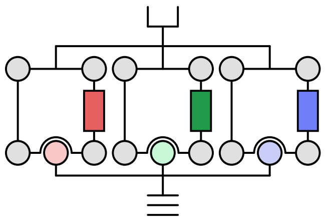

(c) Relaxed structure for TFKM* (Algorithm 8).

Figure 5.6: The mechanism Active Ankle and relaxed variants featuring additional freedoms (virtual prismatic joints).

Algorithm 4 Intersection of three spheres (SphInt)

(in) Spheres; midpoints $\boldsymbol{m}_1$, $\boldsymbol{m}_2$, $\boldsymbol{m}_3$ and radii r_1, r_2, r_3

(out) Intersection; points $\boldsymbol{p}_+$ and $\boldsymbol{p}_-$, with $\|\boldsymbol{p}_+ - \boldsymbol{s}_i\| = r_i$ and $\|\boldsymbol{p}_- - \boldsymbol{s}_i\| = r_i$ for $i \in \{1, 2, 3\}$, or empty set

```
1:  function SphInt($\boldsymbol{m}_1, \boldsymbol{m}_2, \boldsymbol{m}_3, r_1, r_2, r_3$)
2:      $R_1 \leftarrow r_1^2,\ R_2 \leftarrow r_2^2,\ R_3 \leftarrow r_3^2$
3:      $\boldsymbol{Q} \leftarrow \begin{pmatrix} 0 & 1 & 1 & 1 & 1 \\ 1 & 0 & Q(\boldsymbol{m}_1,\boldsymbol{m}_2) & Q(\boldsymbol{m}_1,\boldsymbol{m}_3) & R_1 \\ 1 & Q(\boldsymbol{m}_2,\boldsymbol{m}_1) & 0 & Q(\boldsymbol{m}_2,\boldsymbol{m}_3) & R_2 \\ 1 & Q(\boldsymbol{m}_3,\boldsymbol{m}_1) & Q(\boldsymbol{m}_3,\boldsymbol{m}_2) & 0 & R_3 \\ 1 & R_1 & R_2 & R_3 & 0 \end{pmatrix}$
4:      $D_{(1234)} \leftarrow \frac{1}{8}\det(\boldsymbol{Q})$                    ▷ CM determinant
5:      if $D_{(1234)} < 0$ then                                ▷ Empty intersection
6:          return $\emptyset$
7:      $D_{(123)} \leftarrow -\frac{1}{4}$Minor$(\boldsymbol{Q}, (0,1,2,3), (0,1,2,3), 0)$
8:      $D_{(123;124)} \leftarrow -\frac{1}{4}$Minor$(\boldsymbol{Q}, (0,1,2,3), (0,1,2,4), 0)$
9:      $D_{(123;134)} \leftarrow -\frac{1}{4}$Minor$(\boldsymbol{Q}, (0,1,2,3), (0,1,3,4), 0)$
10:     $\boldsymbol{v}_1 \leftarrow \boldsymbol{m}_2 - \boldsymbol{m}_1$
11:     $\boldsymbol{v}_2 \leftarrow \boldsymbol{m}_3 - \boldsymbol{m}_1$
12:     $\boldsymbol{v}_0 \leftarrow -D_{(123;134)}\boldsymbol{v}_1 + D_{(123;124)}\boldsymbol{v}_2$
13:     $\boldsymbol{v}_\triangle \leftarrow \sqrt{D_{(1234)}}(\boldsymbol{v}_1 \times \boldsymbol{v}_2)$
14:     $\boldsymbol{p}_+ \leftarrow \boldsymbol{m}_1 + \frac{1}{D_{(123)}}(\boldsymbol{v}_0 + \boldsymbol{v}_\triangle)$
15:     $\boldsymbol{p}_- \leftarrow \boldsymbol{m}_1 + \frac{1}{D_{(123)}}(\boldsymbol{v}_0 - \boldsymbol{v}_\triangle)$
16:     return $\boldsymbol{p}_+, \boldsymbol{p}_-$
```

Algorithm 5 Calculation of effector points (CEP)

```
(in) Homogeneous transformation of end effector $\boldsymbol{P}_E$
(out) End effector point matrix $\boldsymbol{E}$
1: function CEP($\boldsymbol{P}_E$)
2:     $\begin{bmatrix} \hat{\boldsymbol{s}} & \hat{\boldsymbol{n}} & \hat{\boldsymbol{a}} & \boldsymbol{e} \\ 0 & 0 & 0 & 1 \end{bmatrix} \leftarrow \boldsymbol{P}_E$    ▷ Extraction
3:     $\boldsymbol{e}_1 \leftarrow \boldsymbol{e} + d\hat{\boldsymbol{n}}, \quad \boldsymbol{e}_2 \leftarrow \boldsymbol{e} - d\hat{\boldsymbol{n}}$
4:     $\boldsymbol{e}_3 \leftarrow \boldsymbol{e} + d\hat{\boldsymbol{a}}, \quad \boldsymbol{e}_4 \leftarrow \boldsymbol{e} - d\hat{\boldsymbol{a}}$
5:     $\boldsymbol{e}_5 \leftarrow \boldsymbol{e} + d\hat{\boldsymbol{s}}, \quad \boldsymbol{e}_6 \leftarrow \boldsymbol{e} - d\hat{\boldsymbol{s}}$
6:     $\boldsymbol{E} \leftarrow (\boldsymbol{e}_i : 1 \le i \le 6)$
7:     return $\boldsymbol{E}$
```

Algorithm 6 Calculation of crank points (CCP)

```
(in) Input joint angles $[q_x, q_y, q_z]$
(out) Crank point matrix $\boldsymbol{C}$
1: function CCP($q_x, q_y, q_z$)
2:     $\boldsymbol{c}_1 \leftarrow (0\ , r\cos q_x\ , l + r\sin q_x)^T$
3:     $\boldsymbol{c}_2 \leftarrow (0\ , -r\cos q_x\ , l - r\sin q_x)^T$
4:     $\boldsymbol{c}_3 \leftarrow (l + r\sin q_y\ , 0\ , r\cos q_y)^T$
5:     $\boldsymbol{c}_4 \leftarrow (l - r\sin q_y\ , 0\ , -r\cos q_y)^T$
6:     $\boldsymbol{c}_5 \leftarrow (r\cos q_z\ , l + r\sin q_z\ , 0)^T$
7:     $\boldsymbol{c}_6 \leftarrow (-r\cos q_z\ , l - r\sin q_z\ , 0)^T$
8:     $\boldsymbol{C} \leftarrow (\boldsymbol{c}_i : 1 \le i \le 6)$
9:     return $\boldsymbol{C}$
```

5.4.2 Rotative Inverse Kinematic Model

Problem 2 (Rotative Inverse Kinematics) *The Rotative Inverse Kinematic Problem (*RIKP*) is defined as the problem of finding the input joint angles needed to achieve a desired orientation of the end effector without having the knowledge of end effector position, formally,*

$$[q_x, q_y, q_z, e_x, e_y, e_z] = \text{RIKP}(\boldsymbol{R}_E), \quad \boldsymbol{R}_E \in SO(3)\ ,$$

where, $\boldsymbol{R}_E$ is the rotation matrix of the end effector w.r.t the global frame, $[q_x, q_y, q_z]$ and $[e_x, e_y, e_z]$ are the active revolute joint angles and end effector shift respectively.

In this case, the system of nonlinear equations Eqs. 5.4–5.9 is well determined as the number of unknowns is equal to the number of equations. To the best knowledge of the authors, it is not possible to derive a closed form solution to this problem due to coupled nature of the constraint equations. Instead of employing standard nonlinear solvers, a novel tailored and efficient algorithm is presented which is suitable for real time control of this mechanism. Its core idea is to decompose the overall equation system into two different equation sets and orthogonally iterate between their solutions to achieve the required overall solution with a desired accuracy. For concrete explanation, two subproblems related to the geometry of ACTIVE ANKLE are presented, namely, the Relaxed Inverse Kinematic Problem (IKP*) and the Relaxed Translative Forward Kinematic Problem (TFKP*). Based on their analytical solutions, the solution to the rotational inverse kinematic problem is presented.

5.4.2.1 Relaxed Inverse Kinematic Model

Since the nature of inverse kinematic problem is overdetermined (see Table 5.4), the two rod equations in each leg are subtracted to obtain a well determined system of leg equations. Problem 1 is relaxed in the sense that it ensures $l_{e_1c_1} = l_{e_2c_2}$, $l_{e_3c_3} = l_{e_4c_4}$, $l_{e_5c_5} = l_{e_6c_6}$ and not $l_{e_ic_i} = l$, $i \in \{1, .., 6\}$. A geometric interpretation of this relaxation is shown in Fig. 5.6b: the rods can be interpreted as virtual prismatic joints which change their lengths in pair in each leg.

Three leg equations Subtracting Equation 5.5 from Equation 5.4, Equation 5.7 from Equation 5.6, Equation 5.9 from Equation 5.8, the three leg equations are derived.

$$\begin{aligned} re_y \cos q_x + r(e_z - l)\sin q_x + d(ln_z - \boldsymbol{e}\cdot\boldsymbol{n}) &= 0 \\ re_z \cos q_y + r(e_x - l)\sin q_y + d(la_x - \boldsymbol{e}\cdot\boldsymbol{a}) &= 0 \\ re_x \cos q_z + r(e_y - l)\sin q_z + d(ls_y - \boldsymbol{e}\cdot\boldsymbol{s}) &= 0 \end{aligned} \tag{5.10}$$

The three leg equations, with the leg index $j \in \{1, 2, 3\}$, are of the form:

$$E_j \cos(q_j) + F_j \sin(q_j) + G_j = 0 \tag{5.11}$$

Relaxed IKP Solution Using the tangent half angle substitution,

$$t_j = \tan(\frac{q_j}{2}), \quad \cos(q_j) = \frac{1-t_j^2}{1+t_j^2}, \quad \sin(q_j) = \frac{2t_j}{1+t_j^2}\ ,$$

a quadratic equation in t is obtained

$$(G_j - E_j)t_j^2 + 2F_jt_j + (G_j + E_j) = 0\ . \tag{5.12}$$

The two solutions of the above quadratic equation is given by:

$$\begin{aligned} t_{j1,2} &= \frac{-F_j \pm \sqrt{E_j^2 + F_j^2 - G_j^2}}{G_j - E_j} \\ q_{j+}, q_{j-} &= 2\,\mathrm{atan2}(-F_j \pm H_j, G_j - E_j) \end{aligned} \tag{5.13}$$

where $H_j = \sqrt{E_j^2 + F_j^2 - G_j^2}$. The expressions for E_j, F_j and G_j for the three legs are given in Table 5.5.

The absolute minimum of the two solutions is chosen so that the mechanism stays close to the zero configuration and respects the physical constraints imposed by either link intersection or limits of passive spherical joints. The solution is implemented in

Table 5.5: Parameters for IKM* solution

Leg Index (j)	E_j	F_j	G_j
$j = 1$	re_y	$r(e_z - l)$	$d(ln_z - \boldsymbol{e} \cdot \boldsymbol{n})$
$j = 2$	re_z	$r(e_x - l)$	$d(la_x - \boldsymbol{e} \cdot \boldsymbol{a})$
$j = 3$	re_x	$r(e_y - l)$	$d(ls_y - \boldsymbol{e} \cdot \boldsymbol{s})$

Algorithm 7 Relaxed inverse kinematic model (IKM*)

```
(in) Homogeneous transformation of end effector $\boldsymbol{P}_E$
(out) Input joint angles $[q_x, q_y, q_z]$
1: function IKM$^\star$($\boldsymbol{P}_E$)
2:     for $j \in (1, 2, 3)$ do
3:         $H_j \leftarrow \sqrt{E_j^2 + F_j^2 - G_j^2}$                ▷ Table 5.5
4:         $q_{j_+}, q_{j_-} \leftarrow 2\,\mathrm{atan2}(-F_j \pm H_j, G_j - E_j)$
5:         $q_{j_+} \leftarrow \mathrm{atan2}(\sin q_{j_+}, \cos q_{j_+})$     ▷ Wrap to $\pm\pi$
6:         $q_{j_-} \leftarrow \mathrm{atan2}(\sin q_{j_-}, \cos q_{j_-})$     ▷ Wrap to $\pm\pi$
7:         $q_j \leftarrow \min(|q_{j_+}|, |q_{j_-}|)$
8:     $[q_x, q_y, q_z] \leftarrow (q_j : 1 \le j \le 3)$
9:     return $[q_x, q_y, q_z]$
```

the method IKM* in Algorithm 7.

5.4.2.2 Relaxed Translative Forward Kinematic Model

Translative Forward Kinematic Problem (TFKP) is defined as the problem of finding the end effector shift from the input joint configuration and desired orientation of the end effector, formally,

$$\boldsymbol{e} = \mathrm{TFKP}(q_x, q_y, q_z, \boldsymbol{R}_E) \ . \tag{5.14}$$

The solution to this problem provides the parasitic motion of the end effector. As noted in Table 5.4, this problem is an over-determined problem as the number of unknowns are three while the number of constraint equations equals to six. Each rod length constraint Eqs. 5.4–5.9 represents the equation of a sphere where the end effector point $[e_x, e_y, e_z]$ moves on its surface. They represent the system of equations of six spheres and the end effector of ACTIVE ANKLE must lie at their intersection point. However, while solving the IKM* it is already ensured that the two rod lengths forming a leg should be the same. So, end effector coordinates can be computed by solving the three rod equations including one from each leg which makes the problem well-determined. The problem is relaxed in the sense that it ensures either $l_{e_1c_1} = l_{e_3c_3} = l_{e_5c_5} = l$ or $l_{e_2c_2} = l_{e_4c_4} = l_{e_6c_6} = l$ and not $l_{e_ic_i} = l$, $i \in \{1, .., 6\}$.

Algorithm 8 Relaxed translative forward kinematic model (TFKM*)

```
(in) Input joint angles $[q_x, q_y, q_z]$ and rotation matrix $\boldsymbol{R}_E$
(out) End effector position $\boldsymbol{e}$
 1: function TFKM$^\star$($q_x, q_y, q_z, \boldsymbol{R}_E$)
 2:     $(r_1, r_2, r_3) \leftarrow l$
 3:     $\boldsymbol{s}_1 \leftarrow (-dn_x, r\cos q_x - dn_y, l - dn_z + r\sin q_x)^T$
 4:     $\boldsymbol{s}_2 \leftarrow (l - da_x + r\sin q_y, -da_y, r\cos q_y - da_z)^T$
 5:     $\boldsymbol{s}_3 \leftarrow (r\cos q_z - ds_x, l - ds_y + r\sin q_z, -ds_z)^T$
 6:     $\boldsymbol{e}_+, \boldsymbol{e}_- \leftarrow$ SPHINT($\boldsymbol{s}_1, \boldsymbol{s}_2, \boldsymbol{s}_3, r_1, r_2, r_3$)    ▷ Algorithm 4
 7:     if $\|e_+\| < d$ then
 8:         $\boldsymbol{e} \leftarrow \boldsymbol{e}_+$
 9:     else
10:         $\boldsymbol{e} \leftarrow \boldsymbol{e}_-$
11:     return $\boldsymbol{e}$
```

A geometric interpretation of this relaxation is shown in Fig. 5.6c: the unchosen rods can be interpreted as virtual prismatic joints which will adjust their lengths so that the chosen rod length becomes equal to l after solving the problem. Overall, the six sphere intersection problem reduces to a three sphere intersection problem.

Relaxed TFKP Solution. Three spheres intersect in maximally two points [Thomas and Ros, 2005]. Without the loss of generality, one can choose to solve for spheres represented by Equation 5.4, Equation 5.6 and Equation 5.8. This particular choice of sphere centers $(\boldsymbol{s}_i)$ and radii r_i, $i \in \{1, 2, 3\}$ is shown in the method TFKM* in Algorithm 8. The end-effector coordinates are estimated using

$$\boldsymbol{e}_+, \boldsymbol{e}_- = \text{SPHINT}(\boldsymbol{s}_1, \boldsymbol{s}_2, \boldsymbol{s}_3, r_1, r_2, r_3)\ . \tag{5.15}$$

The method SPHINT is specified in Algorithm 4. The solution with a norm less than equal to d is selected to avoid the mechanism to leave its assembly. This is implemented in the method TFKM* in Algorithm 8.

5.4.2.3 Solution Approach

The estimated end effector coordinates are defined as $\tilde{e} = [\tilde{e}_x, \tilde{e}_y, \tilde{e}_z]$. In the sequel, the approximate nature of a variable x is expressed by using a tilde $\tilde{x}$. The homogeneous transformation matrix of the end effector w.r.t. the global frame is given by

$$\tilde{\boldsymbol{P}}_E = \begin{bmatrix} \boldsymbol{R}_E & \tilde{e}_{3\times 1} \\ \boldsymbol{0}_{1\times 3} & 1 \end{bmatrix}\ . \tag{5.16}$$

Algorithm 9 Rotative inverse kinematic model (RIKM)

```
(in) Desired orientation of the end effector, $\boldsymbol{R}_E$
(out) Joint angles $[q_x, q_y, q_z]$ and end effector shift $[e_x, e_y, e_z]$
 1: function RIKM($\boldsymbol{R}_E, \epsilon$)
 2:     $\tilde{\boldsymbol{P}}_E \leftarrow \begin{bmatrix} \boldsymbol{R}_E & \mathbf{0}_{3\times1} \\ \mathbf{0}_{1\times3} & 1 \end{bmatrix}$          ▷ Initialization
 3:     while $E_{\text{rgd}} < \epsilon$ do
 4:         $(\tilde{\boldsymbol{e}}_1 \dots \tilde{\boldsymbol{e}}_6) \leftarrow \text{CEP}(\tilde{\boldsymbol{P}}_E)$          ▷ Algorithm 5
 5:         $[\tilde{q}_x, \tilde{q}_y, \tilde{q}_z] \leftarrow \text{IKM}^\star(\tilde{\boldsymbol{P}}_E)$          ▷ Algorithm 7
 6:         $(\tilde{\boldsymbol{c}}_1 \dots \tilde{\boldsymbol{c}}_6) \leftarrow \text{CCP}(\tilde{q}_x, \tilde{q}_y, \tilde{q}_z)$          ▷ Algorithm 6
 7:         $E_{\text{rgd}} \leftarrow \sum_i^6 (\|\tilde{\boldsymbol{e}}_i - \tilde{\boldsymbol{c}}_i\| - l)^2$          ▷ Rigidity error
 8:         $\tilde{\boldsymbol{e}} \leftarrow \text{TFKM}^\star(\tilde{q}_x, \tilde{q}_y, \tilde{q}_z, \boldsymbol{R}_E)$          ▷ Algorithm 8
 9:         $\tilde{\boldsymbol{P}}_E \leftarrow \begin{bmatrix} \boldsymbol{R}_E & \tilde{\boldsymbol{e}}_{3\times1} \\ \mathbf{0}_{1\times3} & 1 \end{bmatrix}$          ▷ Update
10:     $[q_x, q_y, q_z] \leftarrow [\tilde{q}_x, \tilde{q}_y, \tilde{q}_z]$
11:     $[e_x, e_y, e_z] \leftarrow [\tilde{e}_x, \tilde{e}_y, \tilde{e}_z]$
12:     return $[q_x, q_y, q_z, e_x, e_y, e_z]$
```

With an estimated homogeneous transformation matrix ($\tilde{\boldsymbol{P}}_E$), the estimated positions of the six end effector points stored in matrix $\tilde{\boldsymbol{E}}$ are calculated with the help of Algorithm 5 as

$$\tilde{\boldsymbol{E}} = \text{CEP}(\tilde{\boldsymbol{P}}_E)\ . \tag{5.17}$$

The IKM* solution as presented in Section 5.4.2.1 is used to calculate the estimated input joint angles $\tilde{\boldsymbol{q}} = [\tilde{q}_x, \tilde{q}_y, \tilde{q}_z]$ required to achieve the estimated end effector position and desired orientation. It must be recalled that for the derivation of three leg equations Equation 5.10, the two distance constraint equations of each rod constituting a leg are subtracted from each other and hence forcing the two virtual rod lengths of each leg to be equal. Thus, any approximate solution to the inverse kinematic model comes at a cost of incorrect leg lengths.

$$[\tilde{q}_x, \tilde{q}_y, \tilde{q}_z] = \text{IKM}^\star(\tilde{\boldsymbol{P}}_E) \tag{5.18}$$

The estimated input joint angles are now used to estimate the position of six crank points using Algorithm 6.

$$\tilde{\boldsymbol{C}} = \text{CCP}(\tilde{q}_x, \tilde{q}_y, \tilde{q}_z) \tag{5.19}$$

The estimated position vectors of the six end effector points ($\tilde{e}_i$) and the six crank points ($\tilde{c}_i$) are extracted from end effector points matrix $\tilde{\boldsymbol{E}}$ Equation 5.17 and crank points matrix $\tilde{\boldsymbol{C}}$ Equation 5.19 respectively. The length of six virtual rods are calcu-

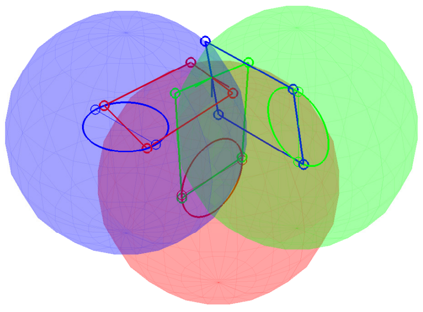


Figure 5.7: Normal working mode, active joint angles: $(q_x, q_y, q_z) \approx (0.0872, 0.1748, 0.2614)$ and end effector shift: $(e_x, e_y, e_z) \approx (0.0127, 0.1515, 0.3807)$

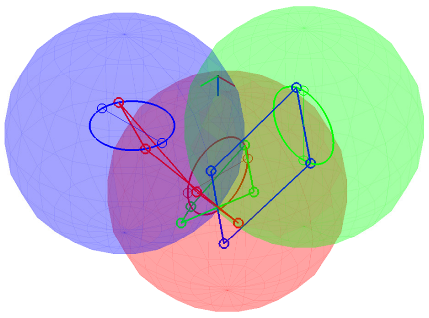


Figure 5.8: Upside – down working mode, active joint angles: $(q_x, q_y, q_z) \approx (0.4566, 0.2377, 0.4663)$ and end effector shift: $(e_x, e_y, e_z) \approx (65.6274, 65.9876, 66.7599)$

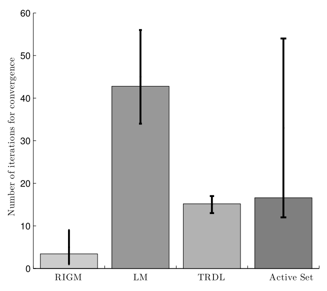


Figure 5.9: Comparison of number of iterations for convergence amongst different RIKM solution strategies.

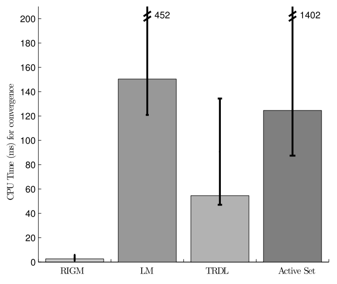


Figure 5.10: Comparison of CPU time for convergence amongst different RIKM solution strategies

lated from $\tilde{\boldsymbol{e}}_i$ and $\tilde{\boldsymbol{c}}_i$ using:

$$\|\tilde{\boldsymbol{e}}_i - \tilde{\boldsymbol{c}}_i\| = \tilde{l}_i \ , \quad i \in \{1, ..., 6\} \tag{5.20}$$

A least square error function to minimize the change in virtual rod lengths is defined as follows:[5]

$$E_{\mathsf{rgd}}(\tilde{\boldsymbol{e}}_i, \tilde{\boldsymbol{c}}_i) = \sum_{i=1}^{6} (\tilde{l}_i - l)^2 \tag{5.21}$$

To minimize the least squared error, the new end effector coordinates ($\tilde{e} = [\tilde{e}_x, \tilde{e}_y, \tilde{e}_z]$) are estimated. Solving for the new end effector position in each iteration is equivalent to solving the relaxed forward kinematic model in translative domain

[5]For improving computational efficiency, the error function can also be chosen to minimize the change in three instead of six rod lengths: the solution of IKM* already ensures that the two rod lengths equal in each leg.

(see Section 5.4.2.2). The solution ensures that the leg lengths become equal to l.

$$\tilde{\boldsymbol{e}} = \text{TFKM}^{\star}(\tilde{q}_x, \tilde{q}_y, \tilde{q}_z, \boldsymbol{R}_E) \tag{5.22}$$

The two solutions in TFKM* lead to two distinct solutions for the RIKM, out of which we are primarily interested in the solution with norm less than d. Each estimation of the end effector position using the method TFKM* minimizes the least squared error function in the next iteration. Hence, the estimated end effector coordinates are substituted back into the Equation 5.16 and the subsequent calculations are iterated until the $E_{\text{rgd}}(\tilde{\boldsymbol{e}}_i, \tilde{\boldsymbol{c}}_i) < \epsilon$ is achieved. The overall rotational inverse kinematic model is implemented in the method RIKM (Algorithm 9). It must be noted that $\tilde{e}$ is initialized as $\mathbf{0}$ at the beginning of the algorithm (Line 2) but this choice does not affect the convergence of the algorithm.[6] The two almost spherical working modes (solutions to the RIKM) for an axis $\boldsymbol{u} \approx (0.2127, 0.5344, 0.8180)^T$ and angle $\phi \approx 0.3140$ are shown in Fig. 5.7 and Fig. 5.8. The numerical convergence towards normal working mode is depicted in Table 5.6.

Iteration	0	1	2
q_x	0.0872	0.0872	0.0872
q_y	0.1745	0.1748	0.1748
q_z	0.2612	0.2613	0.2614
e_x	0.0	0.0071	0.0127
e_y	0.0	0.1499	0.1515
e_z	0.0	0.3678	0.3807
E_{rgd}	0.3370	4.1210^{-04}	4.2910^{-07}

Table 5.6: A numerical example showing the convergence of RIKM for an axis $\boldsymbol{u} \approx (0.2127, 0.5344, 0.8180)^T$ and an angle $\phi \approx 0.3140$.

Benchmarking and Convergence. The solution strategy presented above to solve RIKP is compared with some standard non-linear solvers like Levenberg-Marquardt (LM) and Trust Region Dog Leg (TRDL) implemented in fsolve function of MATLAB as well as constrained optimisation solver using Active Set algorithm implemented in MATLAB function called fmincon. A total of 1000 random orientation samples are chosen from the *physically* feasible workspace of the mechanism and are provided as the input to this problem. RIKM solver demonstrates robust convergence inside the physically feasible workspace of the mechanism. The number of iterations

[6]Instead the convergence to the correct physical configuration is guaranteed by selecting the appropriate intersection point within TFKM*.

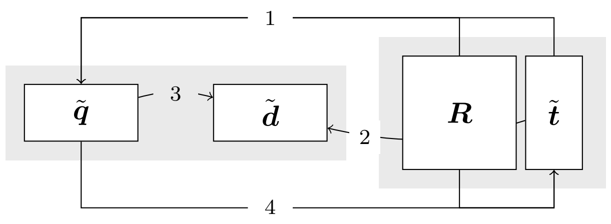


Figure 5.11: Computation scheme of the rotational inverse method RIKM. The matrix of workspace variables is the end-effector pose $\tilde{\boldsymbol{P}}_E \cong (\boldsymbol{R}, \tilde{\boldsymbol{t}})$. The vector of configuration variables $\tilde{q}$ is given by (q_x, q_y, q_z). The vector of design variables, denoted by $\tilde{\boldsymbol{d}}$, includes the vector $\tilde{\boldsymbol{l}}$ that is checked for constraint violation in the abortion criterion of RIKM in Algorithm 9. The computation in RIKM consists of the steps (1) IKM*, (2) CEP, (3) CCP, and (4) TFKM*.

for convergence and the CPU time[7] of RIKM are recorded for benchmarking its efficiency in comparison to standard solvers for a tolerance of $\epsilon = 1.e^{-06}$ mm (See Fig. 5.9 and Fig. 5.10). Error bars used in the two figures are asymmetric and represent min.-max. and average values. With average iterations for convergence equals to 3.42 and CPU time equals to 2.58 milli-seconds, it was found that RIKM performed 21 times faster than TRDL based solver. Hence, it is the most suitable method to achieve a kinematic control on the ACTIVE ANKLE as described in Section 5.6.

Discussion. The computation scheme of the novel RIKM algorithm – that solves the problem of coupled motion kinematics in context of the ACTIVE ANKLE efficiently – is displayed in diagram in Fig. 5.11. From that scheme, it can be observed that the auxiliary variables $\tilde{\boldsymbol{l}}$ – that reflect violations of structural (rigidity) constraints – can be interpreted as virtual joints. The method RIKM ensures that at termination after a few iterations (see Table 5.6), the values of $\tilde{\boldsymbol{l}}$ equal zero. From the viewpoint of kinematic synthesis, this consideration opens a perspective for extending the ACTIVE ANKLE from an almost spherical design to a fully-controllable six-DOF mechanism that, in particular, could also act as a perfect spherical mechanism (compare [Luzi et al., 2018]).

5.5 Forward Kinematics

Problem 3 (Forward Kinematics) *The forward kinematic problem (*FKP*) of the* ACTIVE ANKLE *is to compute a pose of the end effector $\boldsymbol{P}_E$ for given joint angles (q_x, q_y, q_z), as*

$$\boldsymbol{P}_E = \text{FKP}(q_x, q_y, q_z)\ . \tag{5.23}$$

[7]Intel Core i7 CPU 950 @ 3.07GHz x 8PC, 6GB RAM

As noted in Table 5.4, this problem is a well determined problem as the number of unknowns i.e. six parameters identifying the EE pose equals the number of constraint equations of the mechanism. While the inverse kinematic problem IKP (Problem 1) is solved analytically (Section 5.4.1) and the rotative inverse kinematics problem RIKP (Problem 2) is solved with the help of a novel iterative method, the forward kinematics Problem 3 is solved with the tools from computational algebraic geometry which leads to some global insights into mechanism's geometry for e.g. maximum bound on the number of solutions to the forward kinematic problem, real assembly modes of this mechanism and if all these assembly modes are almost spherical in nature. This requires the conversion of constraint equations into a purely algebraic form. One of the important tools to do this task is to exploit Study's kinematic mapping introduced in Section 3.3.1 which maps every displacement in $SE(3)$ to a point in a 7-dimensional projective space $\mathbb{P}^7$ [Husty and Schröcker, 2013, Husty et al., 2007]. Here, a slightly modified version of this mapping is used:

$$\boldsymbol{P}_E = \begin{bmatrix} 1 & 0 & 0 & 0 \\ e_x & {x_0}^2 + {x_1}^2 - {x_2}^2 - {x_3}^2 & -2\,x_0x_3 + 2\,x_1x_2 & 2\,x_0x_2 + 2\,x_1x_3 \\ e_y & 2\,x_0x_3 + 2\,x_1x_2 & {x_0}^2 - {x_1}^2 + {x_2}^2 - {x_3}^2 & -2\,x_0x_1 + 2\,x_3x_2 \\ e_z & -2\,x_0x_2 + 2\,x_1x_3 & 2\,x_0x_1 + 2\,x_3x_2 & {x_0}^2 - {x_1}^2 - {x_2}^2 + {x_3}^2 \end{bmatrix} \tag{5.24}$$

where, e_x, e_y and e_z represent the position of the end effector center in the global frame instead of a position quaternion. Hence, there is no need to consider the equation of Study quadric in this analysis. However, the parameters x_i $(i = 0, ..., 3)$ which constitute the orientation quaternion should satisfy:

$$g_1 := {x_0}^2 + {x_1}^2 + {x_2}^2 + {x_3}^2 - 1 = 0 \tag{5.25}$$

The constraint equations of this mechanism (Equation 5.4–Equation 5.9) derived previously in Section 5.3 are written in terms of direction cosine vectors $(\boldsymbol{s}, \boldsymbol{n}, \boldsymbol{a})$ and the vector of the EE position (e_x, e_y, e_z). Hence, the expressions for the direction cosine vectors are extracted from Equation 5.24 as:

$$\begin{aligned} \hat{\boldsymbol{s}} = [s_x, s_y, s_z]^T &= \left[{x_0}^2 + {x_1}^2 - {x_2}^2 - {x_3}^2, 2\,x_0x_3 + 2\,x_1x_2, -2\,x_0x_2 + 2\,x_1x_3\right] \\ \hat{\boldsymbol{n}} = [n_x, n_y, n_z]^T &= \left[-2\,x_0x_3 + 2\,x_1x_2, {x_0}^2 - {x_1}^2 + {x_2}^2 - {x_3}^2, 2\,x_0x_1 + 2\,x_3x_2\right] \\ \hat{\boldsymbol{a}} = [a_x, a_y, a_z]^T &= \left[2\,x_0x_2 + 2\,x_1x_3, -2\,x_0x_1 + 2\,x_3x_2, {x_0}^2 - {x_1}^2 - {x_2}^2 + {x_3}^2\right] \end{aligned} \tag{5.26}$$

and substituted in the constraint equations (Equation 5.4–Equation 5.9) in order to exploit the above kinematic mapping. The six constraint equations after simplifications along with orientation quaternion normalization equation ($g_1 = 0$) form an ideal $\mathfrak{I} = \langle g_1, g_2, g_3, g_4, g_5, g_6, g_7 \rangle$, where:

$$g_1 := {x_0}^2 + {x_1}^2 + {x_2}^2 + {x_3}^2 - 1 = 0 \tag{5.27}$$

$$\begin{aligned} g_2 :=& (-4e_z r + 4lr)\sin q_x - 4r\cos q_x e_y - 8e_x d(x_0x_3 - x_1x_2) \\ &+ 4e_y d(x_0^2 - x_1^2 + x_2^2 - x_3^2) + 8e_z d(x_0x_1 + x_2x_3) - 8dl(x_0x_1 + x_2x_3) = 0 \end{aligned} \tag{5.28}$$

$$\begin{aligned} g_3 :=& (-4e_y r + 4lr)\sin(q_z) - 4r\cos(q_z)e_x - 8e_z d(x_0x_2 - x_1x_3) \\ &+ 4e_x d(x_0^2 + x_1^2 - x_2^2 - x_3^2) + 8e_y d(x_0x_3 + x_1x_2) - 8dl(x_0x_3 + x_1x_2) = 0 \end{aligned} \tag{5.29}$$

$$\begin{aligned} g_4 :=& (-4e_x r + 4lr)\sin(q_y) - 4r\cos(q_y)e_z - 8e_y d(x_0x_1 - x_2x_3) \\ &+ 4e_z d(x_0^2 - x_1^2 - x_2^2 + x_3^2) + 8e_x d(x_0x_2 + x_1x_3) - 8dl(x_0x_2 - x_1x_3) = 0 \end{aligned} \tag{5.30}$$

$$\begin{aligned} g_5 :=& (-8drx_0x_1 - 8drx_2x_3)\sin(q_x) + 2e_x^2 + 2e_y^2 + 2e_z^2 - 4e_z l + 2d^2 + 2r^2 \\ &+ (-4drx_0^2 + 4drx_1^2 - 4drx_2^2 + 4drx_3^2)\cos(q_x) = 0 \end{aligned} \tag{5.31}$$

$$\begin{aligned} g_6 :=& (-8drx_0x_3 - 8drx_1x_2)\sin(q_z) + 2e_x^2 + 2e_y^2 + 2e_z^2 - 4e_y l + 2d^2 + 2r^2 \\ &+ (-4drx_0^2 - 4drx_1^2 + 4drx_2^2 + 4drx_3^2)\cos(q_z) = 0 \end{aligned} \tag{5.32}$$

$$\begin{aligned} g_7 :=& (-8drx_0x_2 - 8drx_1x_3)\sin(q_y) + 2e_x^2 + 2e_y^2 + 2e_z^2 - 4e_x l + 2d^2 + 2r^2 \\ &+ (-4drx_0^2 + 4drx_1^2 + 4drx_2^2 - 4drx_3^2)\cos(q_y) = 0 \end{aligned} \tag{5.33}$$

The sine and cosine in Eq. (5.28) to (5.33) are replaced with the tangent half-angle expressions: $\sin(q_i) = \frac{2t_i}{1+t_i^2}$ $\cos(q_i) = \frac{1-t_i^2}{1+t_i^2}$ where, $t_i = \tan(\frac{q_i}{2}),\ i = x, y, z$. To this end, t_x, t_y and t_z are the inputs and $x_0, x_1, x_2, x_3, e_x, e_y$ and e_z are the outputs to be solved for in the seven equations $g_i = 0,\ i = 1..7$. The design parameters are substituted as $l = 10$ cm, $d = r = 3.5$ cm.

5.5.1 Rational Univariate Representation of DGP Solution

A Gröbner basis of the ideal $\mathfrak{I} = \langle g_1, g_2, g_3, g_4, g_5, g_6, g_7 \rangle$ is calculated over the field $\mathbb{K}[x_0, x_1, x_2, x_3, e_x, e_y, e_z]$. It was possible to compute the Gröbner basis only after substituting certain values to the inputs q_x, q_y and q_z and to the design parameters $l = 10$ cm, $r = d = 3.5$ cm. For the lexicographic ordering $x_0 <_{lex} \{E_j, x_i\}$ and $x_i <_{lex} \{E_j, x_0\}$ ($i = 1, 2, 3; j = x, y, z$), the univariate polynomial in x_0 turned out to be of degree 28 and 75, respectively which should be halved (due to half-angle substitution) to find unique solutions due to Eq. (5.25). For $E_j <_{lex} x_i$ ($i = 0, 1, 2, 3; j = x, y, z$), this number was 40. Hence, a bound on the maximum number of solutions can be found as $\max\{28/2, 75/2, 40\}$. Thus, the active ankle can have a maximum of 40 direct kinematic solutions.

No.	e_x (cm)	e_y (cm)	e_z (cm)	a_x	a_y	a_z	θ (deg)
1	1.69	1.69	1.69	-0.57	-0.57	-0.57	159.1°
2	4.93	4.93	4.93	-0.57	-0.57	-0.57	148.7°
3	0.06	0.06	0.06	-0.57	-0.57	-0.57	44.3°
4	6.6	6.6	6.6	-0.57	-0.57	-0.57	23.6°
5	0.69	2.12	2.59	-0.28	0.12	-0.94	139.4°
6	2.12	2.59	0.69	0.12	-0.94	-0.28	139.4°
7	2.6	0.69	2.12	0.94	0.28	-0.12	139.4°
8	1.82	3.47	3.78	-0.16	0.32	-0.93	157°
9	3.78	1.82	3.47	0.93	0.16	-0.32	157°
10	3.47	3.78	1.82	0.32	-0.93	-0.16	157°
11	0.63	0.89	1.43	-0.57	0.22	-0.78	107.3°
12	0.89	1.43	0.63	0.22	-0.78	-0.57	107.3°
13	1.43	0.63	0.89	0.78	0.57	-0.22	107.3°
14	5.16	5.88	5.37	0.52	0.06	-0.84	86.1°
15	5.88	5.37	5.16	-0.06	0.84	-0.52	86.1°
16	5.37	5.16	5.88	0.84	-0.52	-0.06	86.1°

Table 5.7: Overview of 16 solutions for the DGP with $q_x = q_y = q_z = 30°$.

5.5.2 Finding Real Solutions

For $t_x = t_y = t_z = \tan(\frac{30°}{2})$, the **RootFinding[Isolate]** function of Maple is used to find out all the real solutions for the set of constraint equations. The algorithm behind this function finds out the rational univariate representation of the set of polynomials and isolates the real roots of these univariate polynomials based on Descartes' rule of sign and the bisection strategy in a unified framework [Rouillier, 2004].
A total of 32 direct kinematic solutions are obtained for $q_x = q_y = q_z = 30°$. Due to Eq. (5.25), this number is to be halved to discard similar solutions. Thus, there are 16 unique assembly modes for the given input. For each assembly mode, the end effector position (e_x, e_y, e_z) and the axis-angle representation (a_x, a_y, a_z, θ) are expressed as follows: $a_x = \frac{x_1}{\sqrt{1-x_0^2}}$, $a_y = \frac{x_2}{\sqrt{1-x_0^2}}$, $a_z = \frac{x_3}{\sqrt{1-x_0^2}}$, $\theta = 2\cos^{-1}(x_0)$. The configuration of these assembly modes is listed in Table 5.7.

Among them, No. 3 and 4 are shown in Figures 5.12 and 5.13. The points corresponding to the position vector $\mathbf{c}_i$ can move on the circumference of those circles drawn. The points $\mathbf{e}_i$ form a spatial cross, the center of which represents the end effector (shown as black sphere). No. 1 – 4 show the assembly modes where $e_x = e_y = e_z$ and $a_x = a_y = a_z$. Since, $q_x = q_y = q_z$, the other assembly modes are observed in triplets with the same axis angle θ and permuted values of (e_x, e_y, e_z) and (a_x, a_y, a_z). Four such triplets are observed in solutions 5 to 7, 8 to 10, 11 to 13 and 14 to 16 in Table 5.7. This pattern may not be visible when $t_i \neq t_j \; \forall i, j = x, y, z$. In addition, this method is used to record the percentage of number of real solutions to DGP by varying q_x, q_y and q_z from -3 radians to 3 radians with an increment of 0.6 radians [Merlet, 2006]. This partitions the configuration space into 1331 permutations of in-

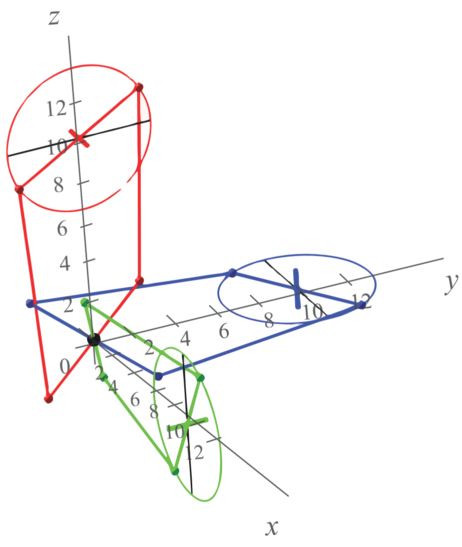


Figure 5.12: Assembly Mode 3 (refer to Table 5.7)

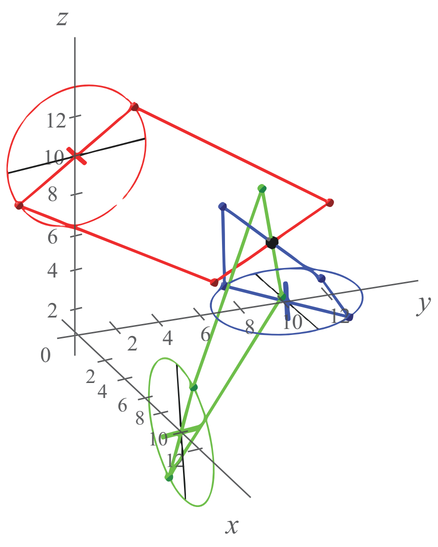


Figure 5.13: Assembly Mode 4 (refer to Table 5.7)

Real solutions	0	2	4	6	8	10	12	14	16	$\sum$
Complex solutions	40	38	36	34	32	30	28	26	24	
Number of poses	258	324	222	222	233	35	24	12	1	1331
Fraction of poses	19.38	24.34	16.68	16.68	17.5	2.63	1.80	0.90	0.08	100%

Table 5.8: Overview of the solvability for $\boldsymbol{q} = (q_x, q_y, q_z)^T \in \mathsf{S}^3$ with discretization $\mathsf{S} = (-3.0, -2.4, \ldots, 2.4, 3.0)$ with cardinality $|\mathsf{S}| = 11$ and $|\mathsf{S}^3| = 1331$.

put angles and the results are shown in Table 5.8. It may be noted that the number of real solutions for any configuration can only be an even number due to an even upper bound on the total number of solutions.

5.5.3 Torsional Motion Analysis

The torsional motion of this manipulator is of practical interest because it corresponds to the adduction-abduction movement when employed as an ankle joint. The torsional motion can be characterized by substituting $e_x = e_y = e_z = E$ and $t_x = t_y = t_z = t$ in seven constraint equations. The Gröbner basis for the ideal $\mathfrak{I}$, now defined over a reduced field $\mathbb{K}[x_0, x_1, x_2, x_3, E]$, is calculated with pure lexicographic order $E <_{lex} x_3 <_{lex} x_2 <_{lex} x_1 <_{lex} x_0$ using Maple software. This yields a Gröbner basis consisting of five polynomials, out of which the first one is an input (t) – output

(E) agnostic description of the mechanism.

$$\begin{aligned} G_1 := &(9t^8 + 36t^6 + 54t^4 + 36t^2 + 9)E^4 + (-1347t^8 - 441t^7 - 4359t^6 - 1029t^5 \\ &- 5877t^4 - 735t^3 - 4065t^2 - 147t - 1200)E^3 + (74251t^8 + 14700t^7 \\ &+ 142899t^6 + 44296t^5 + 139207t^4 + 54096t^3 + 98701t^2 + 24500t + 47350)E^2 \\ &+ (-1710100t^8 + 980000t^7 + 220500t^6 + 19600t^5 + 1239700t^4 - 19600t^3 \\ &+ 739900t^2 - 980000t - 490000)E + 12005000t^8 - 24010000t^7 - 12005000t^6 \\ &+ 48020000t^5 - 12005000t^4 - 24010000t^3 + 12005000t^2 = 0 \end{aligned} \tag{5.34}$$

It shows that a maximum of four assembly modes and a maximum of eight working modes (solutions to the inverse geometric problem) are possible on the subvariety defined by $e_x = e_y = e_z$. The implicit plot of Eq. (5.34) after substituting $t = \tan(q/2)$ is shown in Figure 5.14 for $E = 0, ..., 7$cm and $q = q_x = q_y = q_z = -180°, ..., 180°$. For a value of $q = q_x = q_y = q_z = 30°$, four values of E observed in this figure match with the values noted in Table 5.7. From Figure 5.14, one could also note that the assembly modes shown in Figures 5.12 and 5.13 were actually the *almost-spherical* assembly modes for this mechanism because in these assembly modes the change in end effector's position is minimal.

The second equation of Gröbner's basis in E, t and x_0 is found out to be:

$$\begin{aligned} G_2 := &\left(9\,t^4 + 18\,t^2 + 9\right) E^2 + \left(-600\,t^4 - 294\,t^3 - 906\,t^2 - 600\right) E \\ &+ \left(9800\,t^4 - 9800\right) {x_0}^2 + 4900\,t^4 - 14700\,t^2 + 9800 = 0 \end{aligned} \tag{5.35}$$

Eliminating E from Eq. (5.34) and (5.35) and substituting $t = \tan(q/2)$ and $x_0 = \cos(\theta/2)$ results in an implicit equation in terms of the axis angle θ (representing the rotational workspace) and the actuated variable q. Figure 5.15 shows the implicit plot of q vs. θ for $\theta = -180°, ..., 180°$ and $q = q_x = q_y = q_z = -180°, ..., 180°$.

A Jacobian matrix $\mathbf{J}$ of dimension 5×5 is calculated by partially differentiating the constraint polynomials with respect to the variables of the considered field. When the determinant of this Jacobian vanishes, the mechanism reaches a singularity. Considering the Gröbner basis equations and $det(\mathbf{J}) = 0$, other variables are eliminated to obtain the Eq. (5.36) only in terms of $t_x = t_y = t_z = t = tan(q/2)$.

$$\begin{aligned} det(\mathbf{J}) := &(t-1)(t+1)(t^2+1)(2601t^{12} - 408t^{11} - 55370t^{10} + 54732t^9 \\ &+ 240101t^8 - 491700t^7 + 771464t^6 - 925624t^5 + 751804t^4 \\ &- 497200t^3 + 259600t^2 - 80000t + 10000) = 0 \end{aligned} \tag{5.36}$$

Solving for t and hence q results in six unique solutions which are noticeable as cusps in Figures 5.14 and 5.15. For instance, $q = 90°$ is one of the singularities when E

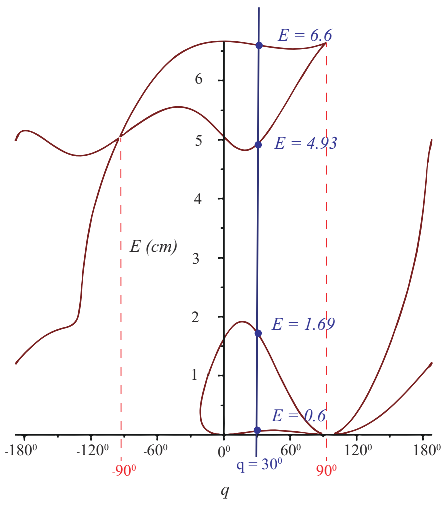


Figure 5.14: Implicit plot between q and E

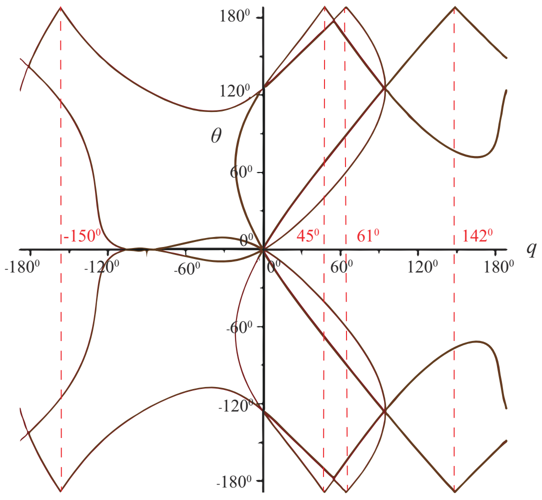


Figure 5.15: Implicit plot between q and θ

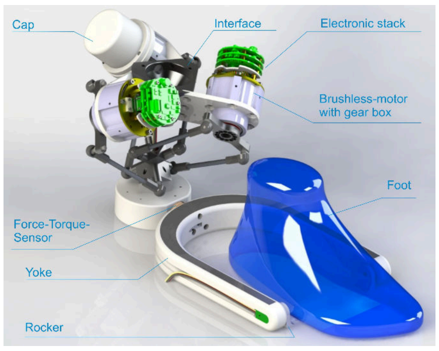


Figure 5.16: Integration of ACTIVE ANKLE as a foot unit of an exoskeleton, with motors, sensors, and electronics.

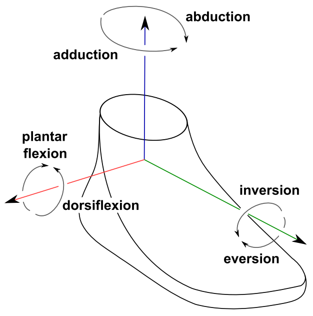


Figure 5.17: Illustration of the three primary rotations of the human ankle joint.

reaches a value of 6.6 cm. Since, other values of E are indeed possible for an input angle of $90°$, it is important to mention the magnitude of the pair $\{E, q\}$ or $\{\theta, q\}$ while representing these singularities.

5.6 ROM Analysis and Control of Active Ankle

ACTIVE ANKLE is used as a 3 DOF active spherical module at hip and ankle joints in the RECUPERA full body exoskeleton, as shown in Fig. 5.1. This section presents its range of motion (ROM) analysis and the experimental results of its kinematic control based on the Rotative Inverse Kinematic Model (RIKM) from Section 5.4.2. For brevity, the results are only reported for its application as an ankle joint.

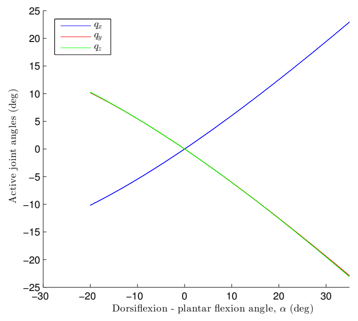


Figure 5.18: Active Joint angles during dorsiflexion – plantarflexion motion.

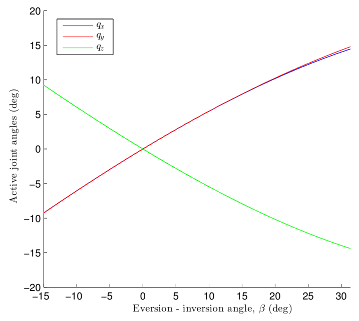


Figure 5.19: Active joint angles during the eversion – inversion motion.

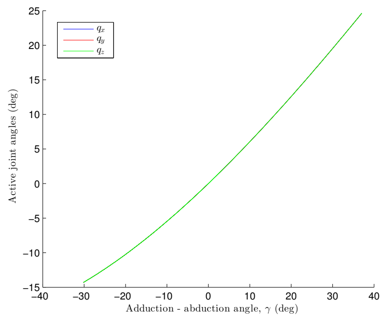


Figure 5.20: Active joint angles during the adduction – abduction motion.

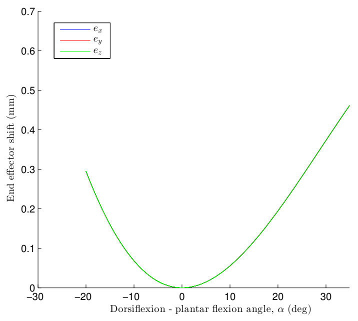


Figure 5.21: End effector shift during the dorsiflexion – plantarflexion motion.

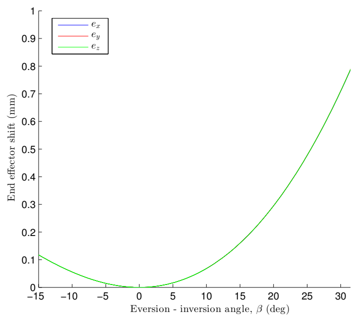


Figure 5.22: End effector shift during the eversion – inversion motion.

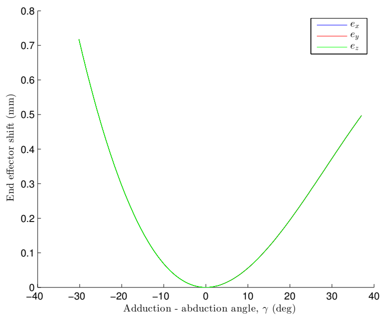


Figure 5.23: End effector shift during the adduction – abduction motion.

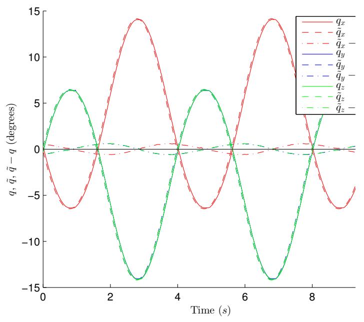


Figure 5.24: Trajectory tracking during dorsiflexion – plantarflexion motion.

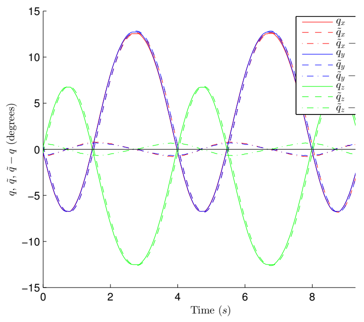


Figure 5.25: Trajectory tracking during the eversion – inversion motion.

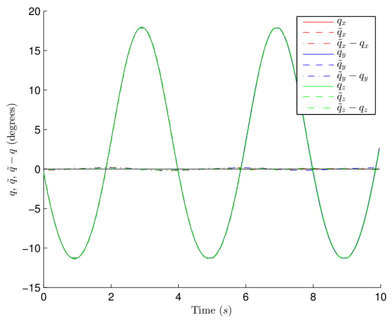


Figure 5.26: Trajectory tracking during the adduction – abduction motion.

5.6.1 Range of motion analysis

The three primary motions available at the human ankle (dorsiflexion–plantarflexion ϕ_α, eversion–inversion ϕ_β, and adduction–abduction ϕ_γ) are shown in Fig. 5.17. Their overall ranges of motion are shown in Table 5.9. During most activities of daily living, only a partial ROM is required: walking on an even surface ($10° - 15°$ plantar flexion and $10°$ dorsiflexion), walking upstairs (37° total ROM), walking downstairs (56° total ROM) [Stauffer et al., 1977].

To demonstrate the suitability of ACTIVE ANKLE mechanism for human ankle related applications (see Fig. 5.16), these movements have been performed on this mechanism using RIKM and its ROM has been evaluated. The ROM offered by the ACTIVE ANKLE is subjected to the physical motion limits of the ball and socket joints. For the presented prototype in Fig. 1.3a, the ball and socket joints used have a motion range of $\pm 25°$. Thus, the maximum possible motion range for the three rotative joints ($J_{01,12}$, $J_{01,32}$, and $J_{01,52}$) lays between $-25°$ and $+25°$. The three task space trajectories in angle-axis representation, where ω represent the angular speed and t is the time, are selected.

$$\begin{aligned}
\boldsymbol{u}_\alpha &= (+1,-1,-1)^T, \quad \phi_\alpha = 28.65° \sin(\omega t) + 8.65° \\
\boldsymbol{u}_\beta &= (+1,+1,-1)^T, \quad \phi_\beta = 25.00° \sin(\omega t) + 10.0° \\
\boldsymbol{u}_\gamma &= (+1,+1,+1)^T, \quad \phi_\gamma = 33.59° \sin(\omega t) + 3.44°
\end{aligned}$$

The input joint angles needed to perform these motions are plotted against the task space angles (ϕ_α, ϕ_β, ϕ_γ) in Fig. 5.18–5.20. The end effector shift encountered while performing these motions have been shown in Fig. 5.21–5.23. In all cases, it can be observed that the shifts are less than 1 mm in magnitude and hence practically insignificant. The available ROM constrained by the physical limits of the ball and socket joints are shown against the human ankle ROM in Table 5.9. It is evident that the presented design can fullfil the range of motion required for most day to day activities. These results show the suitability of ACTIVE ANKLE in full body exoskeletons, ankle rehabilitation devices, and humanoid robots.

Table 5.9: Comparison of primary motion ranges, dorsiflexion – plantarflexion (DF–PF), eversion–inversion (EV–IV), and adduction–abduction (AD–AB), at human and ACTIVE ANKLE.

Motion type	Human Ankle			ACTIVE ANKLE		
	min.	max.	abs.	min.	max.	abs.
DF – PF	−20°	50°	70°	−19.83°	37.23°	57.06°
EV – IV	−15°	35°	50°	−15.00°	35.00°	50.00°
AD – AB	−30°	45°	75°	−29.20°	36.96°	66.16°

5.6.2 Kinematic Control Experiments

The Rotative Inverse Kinematic Model (RIKM) has been implemented within the real time *Robot Construction Kit* (RoCK) [Joyeux and Albiez, 2011] to achieve a kinematic control of the ACTIVE ANKLE. For a tolerance of $\epsilon = 1.e^{-06}$ mm, the algorithm converges in four to six iterations, can be implemented with a control loop frequency of 1 kHz, and is thus suitable for most applications from a control perspective. The kinematic control of ACTIVE ANKLE is implemented on a mid-level control PC communicating with the FPGA electronics of the actuators which implement low level cascaded position, velocity, and torque controllers. These low level controllers ensure safety limits at position, velocity and current levels in any selected control mode [Bargsten and de Gea Fernandez, 2015].

To demonstrate the performance of the kinematic control, the following three task space trajectories corresponding to dorsiflexion–plantarflexion ϕ_α, eversion–inversion ϕ_β, and adduction–abduction ϕ_γ movements are selected:

$$\begin{aligned}
\boldsymbol{u}_\alpha &= (+1, -1, -1)^T, \quad \phi_\alpha = 17.19^\circ \sin(\omega t) + 5.19^\circ \\
\boldsymbol{u}_\beta &= (+1, +1, -1)^T, \quad \phi_\beta = 18.69^\circ \sin(\omega t) + 7.44^\circ \\
\boldsymbol{u}_\gamma &= (+1, +1, +1)^T, \quad \phi_\gamma = 25.19^\circ \sin(\omega t) + 2.58^\circ
\end{aligned}$$

The tracking during the three aforementioned trajectories is shown in Fig. 5.24–5.26 where $(\tilde{q}_x, \tilde{q}_y, \tilde{q}_z)$ are the desired input trajectories to the joints and (q_x, q_y, q_z) demonstrate the real joint states. The maximum absolute errors between desired and real joint trajectories of the three actuators, calculated as $q_e = \max\left(\mid \tilde{q}_x - q_x \mid, \mid \tilde{q}_y - q_y \mid, \mid \tilde{q}_z - q_z \mid\right)$, during the dorsiflexion–plantarflexion, eversion–inversion and adduction–abduction movements are 0.61°, 0.79°, and 0.23°, respectively. The maximum of the mean absolute errors of the three actuators during the three movements are 0.37°, 0.47°, and 0.09°, respectively. These figures demonstrates the robustness of the low level controller.

5.7 Conclusion

This chapter presents a thorough study of the ACTIVE ANKLE, a novel parallel manipulator with mobility three that moves in an almost spherical manner. The type synthesis, mobility analysis and design considerations are presented, unveiling its distinctive features and its suitability as a spherical joint module in various applications. Then, a thorough inverse kinematics analysis is performed where an analytical solution to the full inverse kinematic problem is derived. Also, it is identified that the inverse kinematic problem is not sufficient for its kinematic control due to the almost spherical nature of the device. Subsequently, a rotative inverse kinematic problem is solved with a novel tailored iterative technique which exploits the geometric properties of this mechanism. The solution to the rotative inverse kinematic problem enables the kinematic control of this mechanism in its spherical task space. Further, it presents some global insights into the geometry of the ACTIVE ANKLE mechanism through its forward kinematic analysis using tools from computational algebraic geometry. It is established that the upper bound to the number of unique solutions to forward kinematic problem is 40 which supports our observation that once the actuator angles are fixed in the three legs, ACTIVE ANKLE behaves as a special instance of $6-6$ Stewart platform. In practice, a maximum of 16 real solutions of the forward kinematic problem were found. In addition, the results of the torsional motion analysis which is of practical interest is presented and torsional singularities of the mechanism are highlighted. Moreover, the assembly modes where the mechanism behaves as an *almost-spherical* device are identified. Lastly, the kinematic control of ACTIVE ANKLE is presented with a comparison of its motion range to that of a human ankle joint. The work presented in this chapter can be used to derive the loop closure function (which will be introduced in Section 7.1.3 of Chapter 7) of the mechanism which will be integrated in the HyRoDyn software framework (described in Section 8.4.1 of Chapter 8) as a standalone submechanism library.

Part III

Kinematics and Dynamics

Chapter 6

Screw Theory and Lie Group Methods for Tree Systems

This chapter provides the preliminaries for the Kinematics and Dynamics part of the thesis by visiting the screw theory and Lie group methods for modeling the kinematics and dynamics of tree type systems. The content presented here is largely based on [Müller, 2017, Müller, 2018, Lynch and Park, 2017, Featherstone, 2008] and is included for the completeness of the thesis. The reader with some background in this area may consider skipping this chapter. It starts with graph based topological description of tree type robotic systems in Section 6.1 and then in Section 6.2 presents the recursive relations for computing the kinematics of different bodies in the system. Section 6.3 presents the equations of motion for describing the dynamics of a single rigid body. Section 6.4 presents the recursive Newton Euler algorithm for computing the spatial velocities and wrenches for the robot in order to provide the solution to the inverse dynamics problem. Section 6.5 presents the equations of motion in closed form.

6.1 Graph Based Topological Description

The first step in developing the equations of motion is to start with a system model which describes the existence of various components and how they are connected. This is also referred to as topological description of the multi-body system. The topology of a system is best described with the help of graphs [Jain, 2011a, Featherstone, 2008, Müller, 2018]. The topological or connectivity graph $\mathcal{G}$ has the following properties:

1. The nodes represent bodies and the edges represent joints.

2. The graph is directed i.e. edge is directed from the parent node to its child.

3. The graph is connected i.e. a path exists between any two nodes.
4. Exactly one body represents fixed body which is also known as the root link or base.

Graphs can be used to describe the kinematic topology of any kind of mechanical system including free-floating systems, closed loop mechanisms etc. However, when there is no closed loop in the system, the graph becomes a tree $\mathcal{T}$ i.e. there exists only one path between any two nodes in the graph.

6.1.1 Numbering Scheme for Topological Graphs

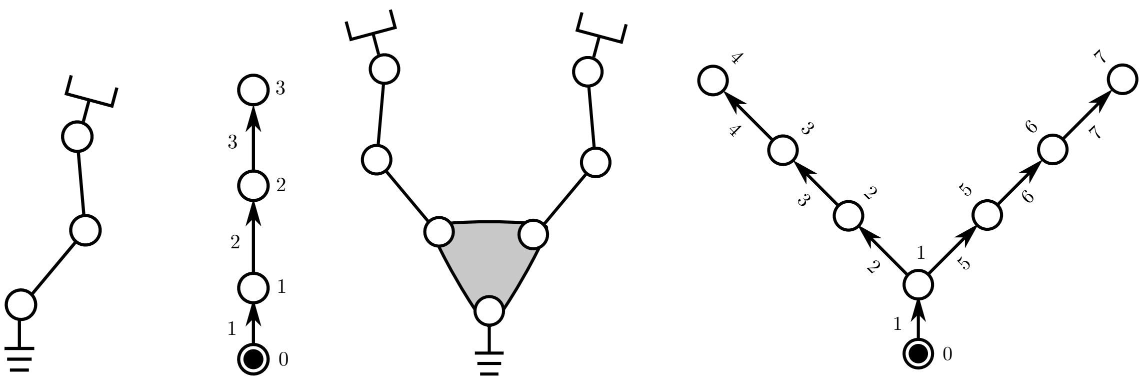
A convenient way to identify bodies and joints in the topological graph is to number them. A well-suited numbering scheme suitable for multi-body dynamics algorithms based on [Featherstone, 2008] called the *regular numbering* is presented here. According to this scheme, the bodies are numbered from 0 to N_B and the joints from 1 to N_J, and respecting the following rules:

1. Choose a spanning tree $\mathcal{T}$.
2. Number 0 is assigned to the fixed root link.
3. Number the remaining nodes from 1 to N_B in any order such that each node has a higher number than its parent in $\mathcal{T}$.
4. Number the edges in $\mathcal{T}$ from 1 to N_B such that the edge i connects between node i and its parent.
5. Number all the remaining edges from $N_B + 1$ to N_J in any order.
6. Each body gets the same number as its node, each joint gets the same number as its edge.

This scheme is only unique for a serial kinematic chain, in all other cases, it is non-unique. Fig. 6.1 shows the examples of regular numbering scheme applied to a serial and tree type mechanism. It is clear from Fig. 6.1a that the numbering scheme is unique. However, it can be noticed in Fig. 6.1b that the numbering scheme is non-unique at the branching i.e. both left and right branches could have started from 2 (in the figure it is the left branch).

6.1.2 Representation of Topological Graphs

It is useful to derive a certain representation for topological graphs which can provide information about connectivity in the graph. One of the simplest way to this is to



(a) Serial Mechanism (b) Tree Mechanism

Figure 6.1: Examples for regular numbering scheme

Table 6.1: Examples for topological graph representation and its properties

Property	Serial mechanism in Fig. 6.1a	Tree mechanism in Fig. 6.1b
p	$\{0,1,2\}$	$\{0,1,2,3,1,5,6\}$
s	$\{1,2,3\}$	$\{1,2,3,4,5,6,7\}$
λ	$\{0,1,2\}$	$\{0,1,2,3,1,5,6\}$
μ	$\{\{1\},\{2\},\{3\}\}$	$\{\{1\},\{2,5\},\{3\},\{4\},$ $\{\},\{6\},\{7\},\{\}\}$
κ	$\{\{1\},\{1,2\},\{1,2,3\}\}$	$\{\{1\},\{1,2\},\{1,2,3\},\{1,2,3,4\},$ $\{1,5\},\{1,5,6\},\{1,5,6,7\}\}$

record the predecessor and successor body numbers in a pair of arrays p and s such that their elements $p(i)$ and $s(i)$ are the predecessor and successor body numbers of joint i. Together, p and s provide a complete description of the topological graph, from which many other useful quantities can be calculated.

- **Parent array** λ identifies the parent of each body in the spanning tree. According to rule 3 and 4 of the regular number scheme, the tree joint i connects the two bodies i and $\lambda(i)$ such that $\lambda(i) < i$, so

$$\lambda(i) = \min(p(i), s(i)) \ , 1 \leq i \leq N_B \ . \tag{6.1}$$

- **Child array** μ identifies the children of each body in the spanning tree.

- **Support array** κ: For any body i except base, $\kappa(i)$ is the set of all tree joints between body i and base.

Table 6.1 shows the representation of topological graphs and their properties for the serial and tree type mechanism examples introduced in Fig. 6.1a and Fig. 6.1b.

6.2 Recursive Kinematics Computation

This section presents the formulation for the computation of position, velocity and acceleration of different bodies in a kinematic chain from its topological description. These formulations are based on screw theory which provides the geometric setting and Lie group theory which forms the analytic foundation (as argued in Section 3.4) for an overall intuitiave and efficient modeling of rigid body mechanisms. A first mathematical introduction to Lie groups and screw theory can be found in Appendix A and Appendix D respectively. To keep the description simple, we will restrict our attention to unbranched kinematic chains or serial robots.

Notation In this thesis, bold letters denote vectors and matrices. However, in the remainder of this chapter, whenever there is a chance of ambiguity, [] denotes the matrix form of the variable. For example, a screw represented by $\boldsymbol{S} = [\boldsymbol{\omega}, \boldsymbol{v}]^T \in \mathbb{R}^6$, when represented as an element of $se(3)$ is given by:

$$[\boldsymbol{S}] = \begin{bmatrix} [\boldsymbol{\omega}] & \boldsymbol{v} \\ \boldsymbol{0} & 0 \end{bmatrix} \tag{6.2}$$

as introduced earlier in Equation 3.15. In above equation, again the angular velocity vector $\boldsymbol{\omega} \in \mathbb{R}^3$ when represented as an element of $so(3)$ is given by:

$$[\boldsymbol{\omega}] = \begin{bmatrix} 0 & -\omega_3 & \omega_2 \\ \omega_3 & 0 & -\omega_1 \\ -\omega_2 & \omega_1 & 0 \end{bmatrix} . \tag{6.3}$$

6.2.1 Position of a Kinematic Chain

The matrix exponential mapping, defined as $\exp : [\boldsymbol{S}]q \in se(3) \rightarrow \boldsymbol{T} \in SE(3)$, can be used to compute the configuration of a rigid body $\boldsymbol{T}$ from the description of its screw axis $\boldsymbol{S}$ and screw coordinate q (see Definition 11). The forward kinematics of a kinematic chain can be computed recursively exploiting the matrix exponential mapping using the product of exponential (POE) formula. The key concept behind it is to regard each joint as applying screw motion to all the outward links.

Definition 13 (Product of Exponentials) *Given the zero configuration of the end-effector of the kinematic chain ${}^0\boldsymbol{T}_n(\boldsymbol{0}) \in SE(3)$ in base frame, its forward kinematics $\boldsymbol{q} \in \mathcal{Q} \mapsto {}^0\boldsymbol{T}_n(\boldsymbol{q}) \in SE(3)$ can be computed using the* ***product of exponentials*** *formula as:*

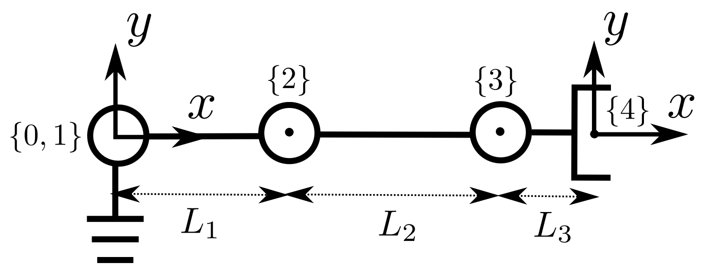
$${}^0\boldsymbol{T}_n(\boldsymbol{q}) = \exp([\boldsymbol{S}_1]q_1)\exp([\boldsymbol{S}_2]q_2)\ldots\exp([\boldsymbol{S}_n]q_n){}^0\boldsymbol{T}_n(\boldsymbol{0}) \tag{6.4}$$

where $\boldsymbol{S}_i$ represents the screw coordinates of the i^{th} joint expressed in the base frame (also known as spatial representation) and q_i is the respective joint displacement relative to the zero configuration.

The POE formula is not minimal i.e. it requires $6n$ scalars to describe n screw axes in addition to n joint coordinate values. However, the advantage of this formula is that it does not require the definition of any link frames unlike the Denavit Hartenberg convention and geometric data can be extracted easily from any CAD software. Further, it is very simple to compute the inverse of the homogenous transformation matrix[1] representing the end effector transformation using:

$$^0\boldsymbol{T}_n^{-1}(\boldsymbol{q}) = {}^0\boldsymbol{T}_n^{-1}(\boldsymbol{0}) \exp(-[\boldsymbol{S}_n]q_n) \ldots \exp(-[\boldsymbol{S}_2]q_2) \exp(-[\boldsymbol{S}_1]q_1) \ . \tag{6.5}$$

Joint	h	$\boldsymbol{s}_o$	$\hat{\boldsymbol{s}}$	$\boldsymbol{S} = \begin{bmatrix} \hat{\boldsymbol{s}} \\ \boldsymbol{s}_o \times \hat{\boldsymbol{s}} + h\hat{\boldsymbol{s}} \end{bmatrix}$
1	0	$[0,0,0]^T$	$[0,0,1]^T$	$[0,0,1,0,0,0]^T$
2	0	$[L_1,0,0]^T$	$[0,0,1]^T$	$[0,0,1,0,-L_1,0]^T$
3	0	$[L_1+L_2,0,0]^T$	$[0,0,1]^T$	$[0,0,1,0,-(L_1+L_2),0]^T$



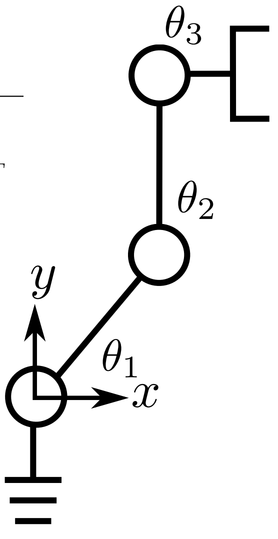


Figure 6.2: Screw description for an unbranched kinematic chain

Example 1 *Consider the unbranched kinematic chain shown in Fig. 6.2. The zero pose configuration of the end effector in the base frame is given by:*

$$^0\boldsymbol{T}_4(\boldsymbol{0}) = \begin{bmatrix} 1 & 0 & 0 & (L_1+L_2+L_3) \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} . \tag{6.6}$$

[1]Recall that the inverse of the matrix exponential function is given by $(\exp(\boldsymbol{A}\theta))^{-1} = \exp(-\boldsymbol{A}\theta)$ where $\boldsymbol{A} \in \mathbb{R}^{n\times n}$.

Using Equation 3.15, the screw axes in matrix screw representation $[\boldsymbol{S}] \in se(3)$ are given by:

$$[\boldsymbol{S}_1] = \begin{bmatrix} 0 & -1 & 0 & 0 \\ 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{bmatrix}, [\boldsymbol{S}_2] = \begin{bmatrix} 0 & -1 & 0 & 0 \\ 1 & 0 & 0 & -L_1 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{bmatrix}, [\boldsymbol{S}_3] = \begin{bmatrix} 0 & -1 & 0 & 0 \\ 1 & 0 & 0 & -(L_1+L_2) \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{bmatrix}. \tag{6.7}$$

The forward kinematics of the mechanism is then given by

$${}^0\boldsymbol{T}_4(\boldsymbol{q}) = \exp([\boldsymbol{S}_1]\theta_1)\exp([\boldsymbol{S}_2]\theta_2)\exp([\boldsymbol{S}_3]\theta_3){}^0\boldsymbol{T}_4(\boldsymbol{0}) \; . \tag{6.8}$$

For $\boldsymbol{q} = [\pi/4, \pi/4, -\pi/2]$*, the forward kinematics of the robot is given by* $\boldsymbol{X} = [x, y, \phi] = [L_1/\sqrt{2} + L_3, L_1/\sqrt{2} + L_2, 0]$ *which represents the right configuration of Fig. 6.2.*

6.2.2 Velocity of a Kinematic Chain

The POE formula[2] (Equation 6.4) used to describe the position of a kinematic chain can be differentiated with respect to time to establish a relationship between joint velocity $\dot{\boldsymbol{q}}$ and end-effector's spatial twist $\boldsymbol{V}$. The subscript and superscript are dropped in Equation 6.4 to simplify the notation.

$$\begin{aligned} \boldsymbol{T}(\boldsymbol{q}) &= \exp([\boldsymbol{S}_1]q_1)\exp([\boldsymbol{S}_2]q_2)\dots\exp([\boldsymbol{S}_n]q_n)\boldsymbol{T}(\boldsymbol{0}) \\ \dot{\boldsymbol{T}} &= \Big(\frac{d(\exp([\boldsymbol{S}_1]q_1))}{dt}\exp([\boldsymbol{S}_2]q_2)\dots\exp([\boldsymbol{S}_n]q_n)\boldsymbol{T}(\boldsymbol{0})\Big) + \\ &\Big(\exp([\boldsymbol{S}_1]q_1)\frac{d(\exp([\boldsymbol{S}_2]q_2))}{dt}\dots\exp([\boldsymbol{S}_n]q_n)\boldsymbol{T}(\boldsymbol{0})\Big) + \dots \end{aligned}$$

The spatial twist $\boldsymbol{V}$ is given by $[\boldsymbol{V}] = \dot{\boldsymbol{T}}\boldsymbol{T}^{-1} \in se(3)$. Using Equation 6.5, one could compute the spatial twist $[\boldsymbol{V}]$ as:

$$\begin{aligned} [\boldsymbol{V}] = {} & \exp([\boldsymbol{S}_1]q_1)[\boldsymbol{S}_1]\exp(-[\boldsymbol{S}_1]q_1)\dot{q}_1 + \\ & \exp([\boldsymbol{S}_1]q_1)\exp([\boldsymbol{S}_2]q_2)[\boldsymbol{S}_2]\exp(-[\boldsymbol{S}_2]q_2)\exp(-[\boldsymbol{S}_1]q_1)\dot{q}_2 + \\ & \exp([\boldsymbol{S}_1]q_1)\exp([\boldsymbol{S}_2]q_2)\exp([\boldsymbol{S}_3]q_3)[\boldsymbol{S}_3]\exp(-[\boldsymbol{S}_3]q_3)\exp(-[\boldsymbol{S}_2]q_2)\exp(-[\boldsymbol{S}_1]q_1)\dot{q}_2 + \dots \end{aligned}$$

The above can be expressed in vector form $\boldsymbol{V} \in \mathbb{R}^6$ with the help of adjoint mapping:

$$\boldsymbol{V} = \underbrace{\mathbf{Ad}_{\exp([\boldsymbol{S}_1]q_1)}(\boldsymbol{S_1})}_{\boldsymbol{J}_1}\dot{q}_1 + \underbrace{\mathbf{Ad}_{\exp([\boldsymbol{S}_1]q_1)\exp([\boldsymbol{S}_2]q_2)}(\boldsymbol{S_2})}_{\boldsymbol{J}_2}\dot{q}_2 + \dots \tag{6.9}$$

[2]Recall that the derivative of a matrix exponential function is given by $\frac{d(\exp(\boldsymbol{A}\theta))}{dt} = \boldsymbol{A}\exp(\boldsymbol{A}\theta)\dot{\theta} = \exp(\boldsymbol{A}\theta)\boldsymbol{A}\dot{\theta}$ where $\boldsymbol{A} \in \mathbb{R}^{n\times n}$ is a constant matrix and $\theta(t)$ is a scalar function of t.

where $\boldsymbol{J}_i = \mathbf{Ad}_{\boldsymbol{T}_i}(\boldsymbol{S}_i)$ with $\boldsymbol{T}_i = \exp([\boldsymbol{S}_1]q_1)\dots\exp([\boldsymbol{S}_i]q_i)$ is the instantaneous screw coordinate vector of the i^{th} joint [3]. It can be observed that the above twist is a sum of n spatial twists and can be written in the following matrix form:

$$\boldsymbol{V} = [\boldsymbol{J}_1 \quad \boldsymbol{J}_2 \quad \dots \quad \boldsymbol{J}_n] \begin{bmatrix} \dot{q}_1 \\ \dot{q}_2 \\ \vdots \\ \dot{q}_n \end{bmatrix} = \boldsymbol{J}(\boldsymbol{q})\dot{\boldsymbol{q}} \tag{6.10}$$

where $\boldsymbol{J}$ is the spatial Jacobian of the kinematic chain with dimension $n \times 6$.

Recursive Nature Inspecting Equation 6.10, the velocity of any body i in the kinematic chain can be expressed in the following summand form.

$$\boldsymbol{V}_i = \sum_{j \leq i} \mathbf{Ad}_{\boldsymbol{T}_i}(\boldsymbol{S}_i)\dot{q}_i \tag{6.11}$$

This also reveals the recursive nature of the velocity computation i.e. the velocity of any body i can be expressed as a sum of the velocity of previous body $i-1$ and velocity across the joint $\dot{q}_i$.

$$\boldsymbol{V}_i = \boldsymbol{V}_{i-1} + \mathbf{Ad}_{\boldsymbol{T}_i}(\boldsymbol{S}_i)\dot{q}_i \tag{6.12}$$

The recursive nature of Equation 6.12 is also exploited in the dynamics algorithms.

Geometric Construction The spatial twist of the end effector can be geometrically constructed using the *instantaneous* screw coordinates expressed in the base frame. It is to be noted that they are configuration dependent i.e. a function of $\boldsymbol{q}$ and equal to the $\boldsymbol{S}_i$ only in the zero configuration.

$$\boldsymbol{V}_n = \begin{bmatrix} \hat{\boldsymbol{s}}_1 \\ \boldsymbol{s}_{o1} \times \hat{\boldsymbol{s}}_1 + h_1\hat{\boldsymbol{s}}_1 \end{bmatrix} \dot{q}_1 + \begin{bmatrix} \hat{\boldsymbol{s}}_2 \\ \boldsymbol{s}_{o2} \times \hat{\boldsymbol{s}}_2 + h_2\hat{\boldsymbol{s}}_2 \end{bmatrix} \dot{q}_2 + \dots + \begin{bmatrix} \hat{\boldsymbol{s}}_n \\ \boldsymbol{s}_{on} \times \hat{\boldsymbol{s}}_n + h_n\hat{\boldsymbol{s}}_n \end{bmatrix} \dot{q}_n \tag{6.13}$$

Example 1 (Continued) *Consider the unbranched kinematic chain shown in Fig. 6.2 again. The spatial Jacobian of the kinematic chain in the zero configuration*

[3]An equivalent formula for computing instantaneous screw coordinates is

$$\boldsymbol{J}_i = \begin{cases} \boldsymbol{S}_1 & : i = 1 \\ \mathbf{Ad}_{\boldsymbol{T}_{i-1}}(\boldsymbol{S}_i) & : i \geq 2 \end{cases}$$

which arises when we use $\frac{d(\exp(\boldsymbol{A}\theta))}{dt} = \boldsymbol{A}\exp(\boldsymbol{A}\theta)\dot{\theta}$ instead of $\exp(\boldsymbol{A}\theta)\boldsymbol{A}\dot{\theta}$. For details, see the derivation in [Lynch and Park, 2017, Murray et al., 1994]

Table 6.2: Instantaneous Screw Description $\boldsymbol{S}(\boldsymbol{q})$
$c_1 = \cos q_1$, $s_1 = \sin q_1$, $c_{12} = \cos(q_1 + q_2)$, $s_{12} = \sin(q_1 + q_2)$

Joint (i)	$\boldsymbol{s}_{oi}$	$\boldsymbol{s}_i$	$\boldsymbol{S}_i = \begin{bmatrix} \hat{\boldsymbol{s}}_i \\ \boldsymbol{s}_{oi} \times \hat{\boldsymbol{s}}_i + h_i \hat{\boldsymbol{s}}_i \end{bmatrix}$
1	$[0,0,0]^T$	$[0,0,0]^T$	$[0,0,1,0,0,0]^T$
2	$[L_1 c_1 s_1, 0]^T$	$[0,0,0]^T$	$[0,0,1,L_1 s_1, -L_1 c_1, 0]^T$
3	$[L_1 c_1 + L_2 c_{12}, L_1 s_1 + L_2 s_{12}, 0]^T$	$[0,0,0]^T$	$[0,0,1,L_1 s_1 + L_2 s_{12}, -L_1 c_1 - L_2 c_{12}, 0]^T$

can be built by vertical concatenation of the joint screws.

$$\boldsymbol{J}(\boldsymbol{0}) = \begin{bmatrix} 0 & 0 & 0 \\ 0 & 0 & 0 \\ 1 & 1 & 1 \\ 0 & 0 & 0 \\ 0 & -L_1 & -(L_1 + L_2) \\ 0 & 0 & 0 \end{bmatrix}_{3\times 6} \tag{6.14}$$

As noted above, the spatial Jacobian is configuration dependent i.e. the instantaneous joint screws as a function of q needs to be derived (as opposed to zero configuration joint screw description from Fig. 6.2). These are provided in Table 6.2. Thus, the expression of the spatial Jacobian in any configuration q is the following.

$$\boldsymbol{J}(\boldsymbol{q}) = \begin{bmatrix} 0 & 0 & 0 \\ 0 & 0 & 0 \\ 1 & 1 & 1 \\ 0 & L_1 \sin q_1 & L_1 \sin q_1 + L_2 \sin(q_1 + q_2) \\ 0 & -L_1 \cos q_1 & -L_1 \cos q_1 - L_2 \cos(q_1 + q_2) \\ 0 & 0 & 0 \end{bmatrix}_{3\times 6} \tag{6.15}$$

System Level Composition The spatial twists of every body can be summarized in overall spatial twist vector $\boldsymbol{V}^{\text{sys}} = [\boldsymbol{V}_1, \ldots, \boldsymbol{V}_n]^T \in \mathbb{R}^{6n}$ and can be computed as

$$\boldsymbol{V}^{\text{sys}} = \boldsymbol{J}^{\text{sys}} \dot{\boldsymbol{q}} \tag{6.16}$$

where $\boldsymbol{J}^{\text{sys}} \in \mathbb{R}^{6n \times n}$ is the *spatial system Jacobian*. The spatial system Jacobian can be factorized and written as

$$\boldsymbol{J}^{\text{sys}} = \boldsymbol{A}^{\text{sys}} \boldsymbol{S}^{\text{sys}} \tag{6.17}$$

where $\boldsymbol{S}^{\text{sys}} = \text{diag}(\boldsymbol{S}_1, \dots, \boldsymbol{S}_n)$ is a block diagonal matrix and $\boldsymbol{A}^{\text{sys}}$ is a lower triangular matrix with the following expressions:

$$\boldsymbol{A}^{\text{sys}} = \begin{bmatrix} [\mathbf{Ad}_{\boldsymbol{T}_1}] & \mathbf{0} & \mathbf{0} & \mathbf{0} \\ [\mathbf{Ad}_{\boldsymbol{T}_1}] & [\mathbf{Ad}_{\boldsymbol{T}_2}] & \mathbf{0} & \mathbf{0} \\ [\mathbf{Ad}_{\boldsymbol{T}_1}] & [\mathbf{Ad}_{\boldsymbol{T}_2}] & [\mathbf{Ad}_{\boldsymbol{T}_3}] & \mathbf{0} \\ \vdots & \vdots & \vdots & \vdots \\ [\mathbf{Ad}_{\boldsymbol{T}_1}] & [\mathbf{Ad}_{\boldsymbol{T}_2}] & [\mathbf{Ad}_{\boldsymbol{T}_3}] & [\mathbf{Ad}_{\boldsymbol{T}_n}] \end{bmatrix}_{6n\times 6n}, \boldsymbol{S}^{\text{sys}} = \begin{bmatrix} \boldsymbol{S}_1 & \mathbf{0} & \mathbf{0} & \mathbf{0} \\ \mathbf{0} & \boldsymbol{S}_2 & \mathbf{0} & \mathbf{0} \\ \mathbf{0} & \mathbf{0} & \boldsymbol{S}_3 & \mathbf{0} \\ \vdots & \vdots & \vdots & \vdots \\ \mathbf{0} & \mathbf{0} & \mathbf{0} & \boldsymbol{S}_n \end{bmatrix}_{6n\times n} \tag{6.18}$$

where $[\mathbf{Ad}_{\boldsymbol{T}_i}]$ is a 6×6 adjoint matrix for the screw transformation of the i^{th} joint (also called as *Plücker transformation* matrix in [Featherstone, 2008]).

6.2.3 Acceleration of a Kinematic Chain

The benefit of POE formula is not only an easy computation of the velocity of a kinematic chain but it also facilitates the recursive computation of any higher-order derivatives. Here, the acceleration computation of the kinematic chain will be discussed. Differentiating Equation 6.9 with respect to time, one gets:

$$\begin{aligned} \dot{\boldsymbol{V}} = \underbrace{\mathbf{Ad}_{\exp([\boldsymbol{S}_1]q_1)}(\boldsymbol{S_1})}_{\boldsymbol{J}_1}\,\ddot{q}_1 + \underbrace{\frac{d}{dt}\mathbf{Ad}_{\exp([\boldsymbol{S}_1]q_1)}(\boldsymbol{S_1})}_{\frac{d}{dt}\boldsymbol{J}_1}\,\dot{q}_1 + \\ \underbrace{\mathbf{Ad}_{\exp([\boldsymbol{S}_1]q_1)\exp([\boldsymbol{S}_2]q_2)}(\boldsymbol{S_2})}_{\boldsymbol{J}_2}\,\ddot{q}_2 + \underbrace{\frac{d}{dt}\mathbf{Ad}_{\exp([\boldsymbol{S}_1]q_1)\exp([\boldsymbol{S}_2]q_2)}(\boldsymbol{S_2})}_{\frac{d}{dt}\boldsymbol{J}_2}\,\dot{q}_2 + \dots \end{aligned} \tag{6.19}$$

which could be written in matrix form as (also equivalent to time differentiation of Equation 6.10):

$$\begin{aligned} \dot{\boldsymbol{V}} = [\boldsymbol{J}_1 \quad \boldsymbol{J}_2 \quad \dots \quad \boldsymbol{J}_n] \begin{bmatrix} \ddot{q}_1 \\ \ddot{q}_2 \\ \vdots \\ \ddot{q}_n \end{bmatrix} + \left[\frac{d}{dt}\boldsymbol{J}_1 \quad \frac{d}{dt}\boldsymbol{J}_2 \quad \dots \quad \frac{d}{dt}\boldsymbol{J}_n\right] \begin{bmatrix} \dot{q}_1 \\ \dot{q}_2 \\ \vdots \\ \dot{q}_n \end{bmatrix} \\ \dot{\boldsymbol{V}} = \boldsymbol{J}\ddot{\boldsymbol{q}} + \dot{\boldsymbol{J}}\dot{\boldsymbol{q}} \end{aligned} \tag{6.20}$$

where $\frac{d\boldsymbol{J}_i}{dt} = \dot{\boldsymbol{J}}_i$ is the time derivative of instantaneous screw coordinates and $\dot{\boldsymbol{J}}$ is the time derivative of the Jacobian matrix of the kinematic chain. It should be recalled that columns of $\boldsymbol{J}$ are configuration dependent and hence an implicit function of time. Hence, it has something to do with the tangential aspect of frame transformation of the screw coordinates.

$$\frac{d\boldsymbol{J}_i}{dt} = \frac{d}{dt}\mathbf{Ad}_{\exp([\boldsymbol{S}_1]q_1)\exp([\boldsymbol{S}_2]q_2)\ldots\exp([\boldsymbol{S}_i]q_i)}(\boldsymbol{S}_i) \tag{6.21}$$

One useful tool in deriving the expression for $\dot{\boldsymbol{J}}$ is Lie bracket. An abstract definition of Lie Brackets can be found in Appendix B, Section B.3. However, in following, we define it in the form that is relevant to formulations for rigid body kinematics.

Definition 14 (Lie Bracket) *Given two twists $\boldsymbol{V}_1 = (\boldsymbol{\omega}_1, \boldsymbol{v}_1)$ and $\boldsymbol{V}_2 = (\boldsymbol{\omega}_2, \boldsymbol{v}_2)$, the* ***Lie Bracket*** *$[\boldsymbol{V}_1, \boldsymbol{V}_2]$ of $\boldsymbol{V}_1$ and $\boldsymbol{V}_2$ written either as $[\mathbf{ad}_{\boldsymbol{V}_1}]\,\boldsymbol{V}_2$ in matrix form or $\mathbf{ad}_{\boldsymbol{V}_1}(\boldsymbol{V}_2)$ in the form of a mapping, is defined as*

$$\begin{bmatrix} [\boldsymbol{\omega}_1] & \mathbf{0} \\ [\boldsymbol{v}_1] & [\boldsymbol{\omega}_1] \end{bmatrix} \begin{bmatrix} \boldsymbol{\omega}_2 \\ \boldsymbol{v}_2 \end{bmatrix} = [\mathbf{ad}_{\boldsymbol{V}_1}]\,\boldsymbol{V}_2 = \mathbf{ad}_{\boldsymbol{V}_1}(\boldsymbol{V}_2) \in \mathbb{R}^{6\times 6} \tag{6.22}$$

where

$$[\mathbf{ad}_{\boldsymbol{V}}] = \begin{bmatrix} [\boldsymbol{\omega}] & \mathbf{0} \\ [\boldsymbol{v}] & [\boldsymbol{\omega}] \end{bmatrix} \tag{6.23}$$

The Lie Bracket is also called as "adjoint mapping", "screw product" or "spatial cross product". The result of the Lie Bracket $[\boldsymbol{V}_1, \boldsymbol{V}_2]$ in vector and matrix notations are given by:

$$[\boldsymbol{V}_1, \boldsymbol{V}_2] = \begin{bmatrix} \boldsymbol{\omega}_1 \times \boldsymbol{\omega}_2 \\ \boldsymbol{v}_1 \times \boldsymbol{\omega}_2 + \boldsymbol{\omega}_1 \times \boldsymbol{v}_2 \end{bmatrix} \in \mathbb{R}^6 = \begin{bmatrix} [\boldsymbol{\omega}_1 \times \boldsymbol{\omega}_2] & \boldsymbol{v}_1 \times \boldsymbol{\omega}_2 + \boldsymbol{\omega}_1 \times \boldsymbol{v}_2 \\ \mathbf{0} & 0 \end{bmatrix}. \tag{6.24}$$

In fact, it can be shown that the i^{th} column of the matrix $\dot{\boldsymbol{J}}$ is simply given by the expression $\dot{\boldsymbol{J}}_i = [\boldsymbol{V}_i, \boldsymbol{J}_i]$ also sometimes written as $\boldsymbol{V}_i \times \boldsymbol{J}_i$ (in the spatial cross product notation from [Featherstone, 2008]).

Recursive Nature Inspecting Equation 6.20, the acceleration of any body i in the kinematic chain can be expressed in the following summand form:

$$\dot{\boldsymbol{V}}_i = \sum_{j \leq i} (\boldsymbol{J}_i \ddot{q}_i + [\boldsymbol{V}_i, \boldsymbol{J}_i]\dot{q}_i) \tag{6.25}$$

which again reveals the recursive nature of the acceleration computation i.e. the acceleration of any body i can be expressed as a sum of the acceleration of previous body $i-1$ and acceleration across the joint $\ddot{q}_i$.

$$\dot{\boldsymbol{V}}_i = \dot{\boldsymbol{V}}_{i-1} + \boldsymbol{J}_i \ddot{q}_i + [\boldsymbol{V}_i, \boldsymbol{J}_i]\dot{q}_i \tag{6.26}$$

Substituting Equation 6.12 in the above equation, and utilizing the bilinearity and anticommutative property of Lie Brackets, one could derive a further simplified recursive equation.

$$\begin{aligned} \dot{\boldsymbol{V}}_i &= \dot{\boldsymbol{V}}_{i-1} + \boldsymbol{J}_i \ddot{q}_i + [\boldsymbol{V}_{i-1}, \boldsymbol{V}_i] \\ \text{or} \quad \dot{\boldsymbol{V}}_i &= \dot{\boldsymbol{V}}_{i-1} + \mathbf{Ad}_{\boldsymbol{T}_i}(\boldsymbol{S}_i)\ddot{q}_i + \mathbf{ad}_{\boldsymbol{V}_{i-1}}(\boldsymbol{V}_i) \end{aligned} \tag{6.27}$$

System Level Composition The spatial acceleration of every body can be summarized in overall spatial acceleration vector $\dot{\boldsymbol{V}}^{\text{sys}} = [\dot{\boldsymbol{V}}_1, \dots, \dot{\boldsymbol{V}}_n]^T \in \mathbb{R}^{6n}$ and can be computed as

$$\dot{\boldsymbol{V}}^{\text{sys}} = \boldsymbol{J}^{\text{sys}}\ddot{\boldsymbol{q}} + \boldsymbol{L}^{\text{sys}}\boldsymbol{b}^{\text{sys}}\boldsymbol{J}^{\text{sys}}\dot{\boldsymbol{q}} = \boldsymbol{J}^{\text{sys}}\ddot{\boldsymbol{q}} + \boldsymbol{L}^{\text{sys}}\boldsymbol{b}^{\text{sys}}\boldsymbol{V}^{\text{sys}} \tag{6.28}$$

with $\boldsymbol{b}^{\text{sys}} = \text{diag}(\mathbf{ad}_{\boldsymbol{V}_1}, \mathbf{ad}_{\boldsymbol{V}_2}, \dots, \mathbf{ad}_{\boldsymbol{V}_n})$ is a diagonal matrix and $\boldsymbol{L}^{\text{sys}}$ being the lower triangular block unit matrix [Müller, 2018] with the following expressions:

$$\boldsymbol{L}^{\text{sys}} = \begin{bmatrix} \boldsymbol{I}_{6\times6} & \mathbf{0}_{6\times6} & \mathbf{0}_{6\times6} & \mathbf{0}_{6\times6} \\ \boldsymbol{I}_{6\times6} & \boldsymbol{I}_{6\times6} & \mathbf{0}_{6\times6} & \mathbf{0}_{6\times6} \\ \boldsymbol{I}_{6\times6} & \boldsymbol{I}_{6\times6} & \boldsymbol{I}_{6\times6} & \mathbf{0}_{6\times6} \\ \vdots & \vdots & \vdots & \vdots \\ \boldsymbol{I}_{6\times6} & \boldsymbol{I}_{6\times6} & \boldsymbol{I}_{6\times6} & \boldsymbol{I}_{6\times6} \end{bmatrix}_{6n\times6n}, \boldsymbol{b}^{\text{sys}} = \begin{bmatrix} \mathbf{ad}_{\boldsymbol{V}_1} & \mathbf{0} & \mathbf{0} & \mathbf{0} \\ \mathbf{0} & \mathbf{ad}_{\boldsymbol{V}_2} & \mathbf{0} & \mathbf{0} \\ \mathbf{0} & \mathbf{0} & \mathbf{ad}_{\boldsymbol{V}_3} & \mathbf{0} \\ \vdots & \vdots & \vdots & \vdots \\ \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{ad}_{\boldsymbol{V}_n} \end{bmatrix}_{6n\times6n}. \tag{6.29}$$

6.3 Dynamics of a Single Rigid Body

This section presents the Newton-Euler equations for describing the dynamics of a single rigid body motion in $SE(3)$. In this regard, first preliminary physical concepts necessary to describe the dynamics of a rigid body are briefly introduced. To motivate the advantages of the screw theory and Lie group based approach to robot dynamics, we start with the classical formulation of dynamics of a single rigid body. Then, a twist-wrench based formulation exploiting the Lie Group theory is presented.

6.3.1 Physical Properties of a Rigid Body

Rigid body is defined as an object in which the distance between any two given points remains constant in time regardless of external forces exerted on it. A rigid body can be seen as a set of point masses with fixed distances between each other. There are three important physical properties of a rigid body that must be taken into account to describe its dynamics.

- **Mass** Mass of a physical body is the measure of its resistance to a change in state of its motion (acceleration). If we think of a rigid body as a composition of rigidly connected point masses, then the total mass of the rigid body is given by $m = \sum m_i$ where m_i is the mass of i^{th} point mass.

- **Centre of mass (COM)** The center of mass $\boldsymbol{c}$ of a rigid body in space is the unique point where the weighted relative position of the distributed mass sums to zero. Formally speaking, the center of mass is the location of a point such that $\sum m_i \boldsymbol{r}_i = 0$ where $\boldsymbol{r}_i = (x_i, y_i, z_i)$ is the location of a point mass m_i with respect to the COM. It can also be interpreted as the first moment of mass.

- **Moment of Inertia or Rotational Interia** Moment of Inertia of a physical body is the measure of its resistance to a change in rotational motion (angular acceleration). It can also be interpreted as the second moment of mass and is given by

$$\boldsymbol{I}_b = \begin{bmatrix} I_{xx} & I_{xy} & I_{xz} \\ I_{xy} & I_{yy} & I_{yz} \\ I_{xz} & I_{yz} & I_{zz} \end{bmatrix} = \begin{bmatrix} \sum m_i(y_i^2 + z_i^2) & -\sum m_i x_i y_i & -\sum m_i x_i z_i \\ -\sum m_i x_i y_i & \sum m_i(x_i^2 + z_i^2) & -\sum m_i y_i z_i \\ -\sum m_i x_i z_i & -\sum m_i y_i z_i & \sum m_i(x_i^2 + y_i^2) \end{bmatrix} \in \mathbb{R}^{3\times 3} . \tag{6.30}$$

The summations can be replaced by volume integrals over the body b using the differential volume element dV, with point masses m_i replaced by a mass density function $\rho(x, y, z)$. In either case of these cases, $\boldsymbol{I}_B$ is *symmetric* and *positive-definite* for any rigid body.

All of these properties are intrinsic to the rigid body itself and do not change under the influence of external forces or time. Overall, one needs a set of 10 real valued parameters to describe the mass-inertial properties of a rigid body.

$$\Phi = [m, c_x, c_y, c_z, I_{xx}, I_{yy}, I_{zz}, I_{xy}, I_{yz}, I_{xz}] \in \mathbb{R}^{10} \tag{6.31}$$

6.3.2 Classical Formulation

Consider a single rigid body of mass m consisting of a number of rigidly connected point masses m_i. Assume that this body is moving with a linear velocity $\boldsymbol{v}_b$ and angular velocity $\boldsymbol{\omega}_b$, and let $\boldsymbol{p}_i(t)$ denote the time varying position of m_i, initially located $\boldsymbol{r}_i$ in the inertial frame $\{b\}$. Then, the velocity and acceleration of any point on this rigid body can be described as

$$\begin{aligned} \dot{\boldsymbol{p}}_i &= \boldsymbol{v}_b + \boldsymbol{\omega}_b \times \boldsymbol{p}_i \\ \ddot{\boldsymbol{p}}_i &= \dot{\boldsymbol{v}}_b + \dot{\boldsymbol{\omega}}_b \times \boldsymbol{p}_i + \boldsymbol{\omega}_b \times (\boldsymbol{v}_b + \boldsymbol{\omega}_b \times \boldsymbol{p}_i) \end{aligned} \tag{6.32}$$

which can be written in the matrix form as

$$\ddot{\boldsymbol{p}}_i = \dot{\boldsymbol{v}}_b + [\dot{\boldsymbol{\omega}}_b]\boldsymbol{r}_i + [\boldsymbol{\omega}_b]\boldsymbol{v}_b + [\boldsymbol{\omega}_b]^2\boldsymbol{r}_i \ . \tag{6.33}$$

From Newton's 2nd law of motion, the force acting on this point is $\boldsymbol{f}_i = m_i\ddot{\boldsymbol{p}}_i$ and implies a moment $\boldsymbol{m}_i = \boldsymbol{r}_i \times \boldsymbol{f}_i = [\boldsymbol{r}_i]\boldsymbol{f}_i$. The total force acting on this body $\boldsymbol{f}_b$ is given by

$$\boldsymbol{f}_b = \sum_i m_i(\dot{\boldsymbol{v}}_b + [\dot{\boldsymbol{\omega}}_b]\boldsymbol{r}_i + [\boldsymbol{\omega}_b]\boldsymbol{v}_b + [\boldsymbol{\omega}_b]^2\boldsymbol{r}_i) \tag{6.34}$$

which can be simplified if the body frame is assumed to coincide with the COM i.e. $\sum m_i\boldsymbol{r}_i = 0$. The simplified expression for the total force is

$$\boldsymbol{f}_b = m(\dot{\boldsymbol{v}}_b + [\dot{\boldsymbol{\omega}}_b]\boldsymbol{v}_b) \ . \tag{6.35}$$

Similarly, the total moment acting on the body $\boldsymbol{m}_b$ is given by

$$\boldsymbol{m}_b = \sum_i m_i[\boldsymbol{r}_i](\dot{\boldsymbol{v}}_b + [\dot{\boldsymbol{\omega}}_b]\boldsymbol{r}_i + [\boldsymbol{\omega}_b]\boldsymbol{v}_b + [\boldsymbol{\omega}_b]^2\boldsymbol{r}_i) \tag{6.36}$$

which also gets simplified due to the above assumption to

$$\begin{aligned} \boldsymbol{m}_b = \left(-\sum_i m_i[\boldsymbol{r}_i]^2\right)\dot{\boldsymbol{\omega}}_b + [\dot{\boldsymbol{\omega}}_b]\left(-\sum_i m_i[\boldsymbol{r}_i]^2\right)\boldsymbol{\omega}_b \\ \text{or} \quad \boldsymbol{m}_b = \boldsymbol{I}_b\dot{\boldsymbol{\omega}}_b + [\dot{\boldsymbol{\omega}}_b]\boldsymbol{I}_b\boldsymbol{\omega}_b \ . \end{aligned} \tag{6.37}$$

The above equation is also known as Euler's equation for a rotating body. Equation 6.35 and Equation 6.37 together constitute the Newton-Euler equations of motion for the single rigid body system and look fairly simple. However, if the same equations were expressed in any other frame, then they would be quite complicated (for e.g. revisit Equation 6.34 and Equation 6.36). It is not hard to imagine that this complexity manifests stronger in case of multibody dynamics where multiple bodies are involved. The forces and moments acting on a specific link are typically expressed in different reference frames, and these must be expressed in terms of a common frame before they can be summed.

6.3.3 Twist-Wrench Formulation

Let us collect the angular velocity and linear velocity of the body in a body twist vector $\boldsymbol{V}_b = (\boldsymbol{\omega}_b, \boldsymbol{v}_b)$ and moment and force vectors in a body wrench vector $\boldsymbol{W}_b = (\boldsymbol{m}_b, \boldsymbol{f}_b)$.

Equation 6.35 and Equation 6.37 can be collected and written in a combined form as

$$\begin{bmatrix} \boldsymbol{m}_b \\ \boldsymbol{f}_b \end{bmatrix} = \begin{bmatrix} \boldsymbol{I}_b & \boldsymbol{0} \\ \boldsymbol{0} & m\boldsymbol{I} \end{bmatrix} \begin{bmatrix} \dot{\boldsymbol{\omega}}_b \\ \dot{\boldsymbol{v}}_b \end{bmatrix} + \begin{bmatrix} [\boldsymbol{\omega}_b] & \boldsymbol{0} \\ \boldsymbol{0} & [\boldsymbol{\omega}_b] \end{bmatrix} \begin{bmatrix} \boldsymbol{I}_b & \boldsymbol{0} \\ \boldsymbol{0} & m\boldsymbol{I} \end{bmatrix} \begin{bmatrix} \boldsymbol{\omega}_b \\ \boldsymbol{v}_b \end{bmatrix} \tag{6.38}$$

where $\boldsymbol{I}$ is a 3×3 identity matrix. Exploiting the $[\boldsymbol{v}]\boldsymbol{v} = \boldsymbol{v} \times \boldsymbol{v} = 0$ and $[\boldsymbol{v}]^T = -[\boldsymbol{v}]$ properties of a skew-symmetric matrix, we can rewrite the above equation as:

$$\begin{aligned} \begin{bmatrix} \boldsymbol{m}_b \\ \boldsymbol{f}_b \end{bmatrix} &= \begin{bmatrix} \boldsymbol{I}_b & \boldsymbol{0} \\ \boldsymbol{0} & m\boldsymbol{I} \end{bmatrix} \begin{bmatrix} \dot{\boldsymbol{\omega}}_b \\ \dot{\boldsymbol{v}}_b \end{bmatrix} + \begin{bmatrix} [\boldsymbol{\omega}_b] & [\boldsymbol{v}_b] \\ \boldsymbol{0} & [\boldsymbol{\omega}_b] \end{bmatrix} \begin{bmatrix} \boldsymbol{I}_b & \boldsymbol{0} \\ \boldsymbol{0} & m\boldsymbol{I} \end{bmatrix} \begin{bmatrix} \boldsymbol{\omega}_b \\ \boldsymbol{v}_b \end{bmatrix} \\ \underbrace{\begin{bmatrix} \boldsymbol{m}_b \\ \boldsymbol{f}_b \end{bmatrix}}_{\boldsymbol{W}_b} &= \underbrace{\begin{bmatrix} \boldsymbol{I}_b & \boldsymbol{0} \\ \boldsymbol{0} & m\boldsymbol{I} \end{bmatrix}}_{\boldsymbol{M}_b} \underbrace{\begin{bmatrix} \dot{\boldsymbol{\omega}}_b \\ \dot{\boldsymbol{v}}_b \end{bmatrix}}_{\dot{\boldsymbol{V}}_b} - \underbrace{\begin{bmatrix} [\boldsymbol{\omega}_b] & \boldsymbol{0} \\ [\boldsymbol{v}_b] & [\boldsymbol{\omega}_b] \end{bmatrix}^T}_{\mathbf{ad}^T_{\boldsymbol{V}_b}} \underbrace{\begin{bmatrix} \boldsymbol{I}_b & \boldsymbol{0} \\ \boldsymbol{0} & m\boldsymbol{I} \end{bmatrix}}_{\boldsymbol{M}_b} \underbrace{\begin{bmatrix} \boldsymbol{\omega}_b \\ \boldsymbol{v}_b \end{bmatrix}}_{\boldsymbol{V}_b} \end{aligned} \tag{6.39}$$

where each term can be identified as a spatial quantity. It shares a peculiar resemblence with Equation 6.37 and could be seen as spatial version of this equation. In particular, we discuss the following two terms:

- **Spatial mass-inertia matrix** of a rigid body $\boldsymbol{M}_b \in \mathbb{R}^{6\times 6}$ is defined as:

$$\boldsymbol{M}_b = \begin{bmatrix} \boldsymbol{I}_b & \boldsymbol{0} \\ \boldsymbol{0} & m\boldsymbol{I} \end{bmatrix} \tag{6.40}$$

  and is symmetric and positive definite. Using it, the total kinetic energy of the rigid body can be written as

$$K = \frac{1}{2}\boldsymbol{V}_b^T \boldsymbol{M}_b \boldsymbol{V}_b = \frac{1}{2}\boldsymbol{v}_b^T m \boldsymbol{v}_b + \frac{1}{2}\boldsymbol{\omega}_b^T \boldsymbol{I}_b \boldsymbol{\omega}_b \in \mathbb{R}^+ \ . \tag{6.41}$$

- **Spatial momentum** coscrew $\boldsymbol{P}_b = (\boldsymbol{L}_b, \boldsymbol{p}_b)^T \in se^*(3)$ composed of the angular momentum of the body $\boldsymbol{L}_b$ and its linear momentum $\boldsymbol{p}_b$ is defined as

$$\boldsymbol{P}_b = \begin{bmatrix} \boldsymbol{I}_b & \boldsymbol{0} \\ \boldsymbol{0} & m\boldsymbol{I} \end{bmatrix} \begin{bmatrix} \boldsymbol{\omega}_b \\ \boldsymbol{v}_b \end{bmatrix} = \boldsymbol{M}_b \boldsymbol{V}_b \in \mathbb{R}^6 \ . \tag{6.42}$$

  Using the spatial momentum in body frame, one can compute the wrench acting on the body $\boldsymbol{W}_b$ as:

$$\boldsymbol{W}_b = \dot{\boldsymbol{P}}_b - \left[\mathbf{ad}^T_{\boldsymbol{V}_b}\right] \boldsymbol{P}_b \tag{6.43}$$

Thus, the EOM can be written in the following compact form.

$$\boldsymbol{W}_b = \boldsymbol{M}_b \dot{\boldsymbol{V}}_b - \left[\mathbf{ad}^T_{\boldsymbol{V}_b}\right] \boldsymbol{M}_b \boldsymbol{V}_b \tag{6.44}$$

In contrast to two equations Equation 6.35 and Equation 6.37 describing the dynamics of the rigid body in the classical formulation, the modern geometric approach allows the description of the dynamics using a single compact equation like Equation 6.44. However, the bigger advantage of utilizing the Lie Group methods in building the equation of motion is the fact that unlike Equation 6.37, the dynamics of a rigid body can always be expressed in the form of Equation 6.44 regardless of the frame in which it is expressed.

Frame Invariance It was shown in Equation 6.41 that the spatial form can also be used to write a compact expression for the kinetic energy. Since, the kinetic energy of the rigid body must be indepedent of the frame of representation {a} or {b}, one can write

$$\begin{aligned} K &= \frac{1}{2}\boldsymbol{V}_a^T \boldsymbol{M}_a \boldsymbol{V}_a = \frac{1}{2}\boldsymbol{V}_b^T \boldsymbol{M}_b \boldsymbol{V}_b \\ &= \frac{1}{2}\boldsymbol{V}_a^T \underbrace{\left[\mathbf{Ad}_{{}^b\boldsymbol{T}_a}\right]^T \boldsymbol{M}_b \left[\mathbf{Ad}_{{}^b\boldsymbol{T}_a}\right]}_{\boldsymbol{M}_a} \boldsymbol{V}_a \end{aligned} \tag{6.45}$$

where the spatial mass inertia matrix in the frame {a} is related to the spatial inertia matrix in frame {b} by the relation:

$$\boldsymbol{M}_a = \left[\mathbf{Ad}_{{}^a\boldsymbol{T}_b}\right]^{-T} \boldsymbol{M}_b \left[\mathbf{Ad}_{{}^a\boldsymbol{T}_b}\right]^{-1} \ . \tag{6.46}$$

Spatial Representation Equation 6.44 can also be written in the spatial representation which in fact reveals an interesting property. Using Equation 6.46, the spatial mass-inertia matrix in the spatial representation $\boldsymbol{M}$ can be computed from spatial mass-inertia matrix in the body frame $\boldsymbol{M}_b$ as:

$$\boldsymbol{M} = \left[\mathbf{Ad}_{\boldsymbol{T}}\right]^{-T} \boldsymbol{M}_b \left[\mathbf{Ad}_{\boldsymbol{T}}\right]^{-1} \tag{6.47}$$

where $\boldsymbol{T}$ denotes the homogenous transformation of the body frame with respect to the base frame. The spatial momentum of the rigid body in the spatial representation is defined as

$$\boldsymbol{P} = \boldsymbol{M}\boldsymbol{V} = \left[\mathbf{Ad}_{\boldsymbol{T}}\right]^{-T} \boldsymbol{M}_b \left[\mathbf{Ad}_{\boldsymbol{T}}\right]^{-1} \boldsymbol{V} \ . \tag{6.48}$$

Using the fact that $\boldsymbol{V}_b = \left[\mathbf{Ad}_{\boldsymbol{T}}\right] \boldsymbol{V}$ and Equation 6.42, one could express the spatial momentum in spatial representation $\boldsymbol{P}$ in terms of spatial momentum in body fixed representation $\boldsymbol{P}_b$ as:

$$\begin{aligned} \boldsymbol{P} &= \left[\mathbf{Ad}_{\boldsymbol{T}}\right]^{-T} \boldsymbol{M}_b \boldsymbol{V}_b \\ &= \left[\mathbf{Ad}_{\boldsymbol{T}}\right]^{-T} \boldsymbol{P}_b \ . \end{aligned} \tag{6.49}$$

Differentiating Equation 6.48 with respect to time, one could arrive at the Newton-Euler equation of the single rigid body dynamics in spatial representation:

$$\boldsymbol{W} = \dot{\boldsymbol{P}} = \boldsymbol{M}\dot{\boldsymbol{V}} + \dot{\boldsymbol{M}}\boldsymbol{V} \tag{6.50}$$

which is the simplest form possible (unlike Equation 6.43) and can be achieved by the spatial representation of the twist, wrench and momentum. The time derivative of spatial mass-inertia matrix $\dot{\boldsymbol{M}}$ can be computed using

$$\dot{\boldsymbol{M}} = -\left[\mathbf{ad}_{\boldsymbol{V}}^{T}\right]\boldsymbol{M} - \boldsymbol{M}\left[\mathbf{ad}_{\boldsymbol{V}}\right] \ . \tag{6.51}$$

Substituting Equation 6.51 in Equation 6.50 and using the fact that $[\mathbf{ad}_{\boldsymbol{V}}]\,\boldsymbol{V} = 0$, one could arrive at the NE equations of motion in spatial representation.

$$\begin{aligned} \boldsymbol{W} &= \boldsymbol{M}\dot{\boldsymbol{V}} - \left[\mathbf{ad}_{\boldsymbol{V}}^{T}\right]\boldsymbol{M}\boldsymbol{V} - \boldsymbol{M}\left[\mathbf{ad}_{\boldsymbol{V}}\right]\boldsymbol{V} \\ \boldsymbol{W} &= \boldsymbol{M}\dot{\boldsymbol{V}} - \left[\mathbf{ad}_{\boldsymbol{V}}^{T}\right]\boldsymbol{M}\boldsymbol{V} \end{aligned} \tag{6.52}$$

This has exactly the same form as body frame version of EOM as shown in Equation 6.44.

6.4 Inverse Dynamics

Inverse dynamics is the problem of finding the forces required to produce a given motion in a rigid-body system. For a serial or tree type robot, all joints are assumed to be actuated and hence these forces can be attributed to the motor torques and is useful in robot analysis and control. In the previous sections, it was shown how a kinematic chain can be topolgically described using the concepts from graph theory, then recursive formulations for computing the position, velocity and acceleration kinematics were presented. Then, equations of motion for describing a single rigid body dynamics were presented. In this section, equations of motion for a serial or tree type system would be solved recursively exploiting the concepts presented in previous sections. The inverse dynamics of a general kinematic tree can be obtained in two main steps:

1. Calculate the position, velocity and acceleration state of each body in the kinematic tree. This is called the **forward recursion**.

2. Calculate the wrenches required to produce these accelerations and subsequently the wrenches transmitted across the joints from the wrenches acting on the bodies. This step is called **backward recursion**.

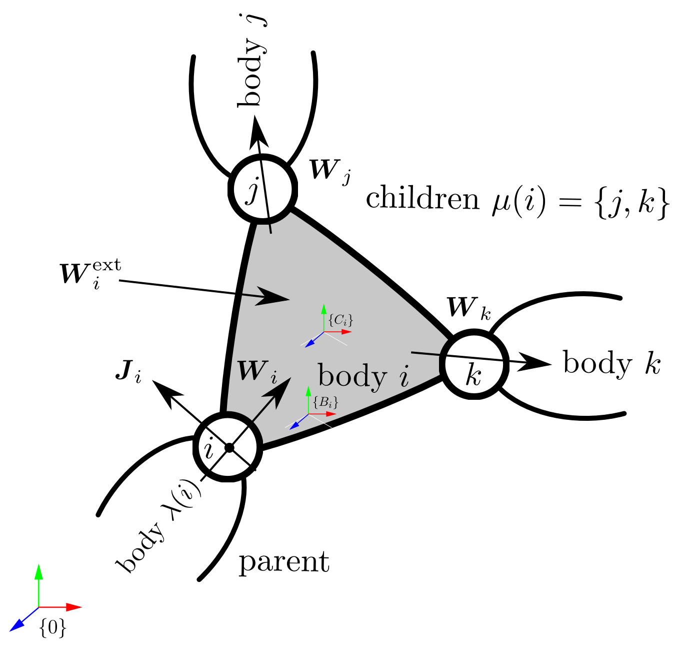


Figure 6.3: Forces acting on a body i

6.4.1 Initialization

Fig. 6.3 shows the free body diagram of a body i which is a part of the kinematic tree. Recall that in a kinematic tree, every body has a unique parent. Revisiting the notation introduced in Section 6.1, $\lambda(i)$ denotes the parent of the body i and its children are given by the array $\mu(i)$. Let us define a body frame $\{B_i\}$ at every body and let ${}^{i}\boldsymbol{S}_i$ denote the constant joint screw of joint i represented in the body frame $\{B_i\}$ given by:

$$ {}^{i}\boldsymbol{S}_i = \begin{bmatrix} {}^{i}\hat{\boldsymbol{s}}_i \\ {}^{i}\boldsymbol{s}_{oi} \times {}^{i}\hat{\boldsymbol{s}}_i + h^{i}\hat{\boldsymbol{s}}_i \end{bmatrix} \in \mathbb{R}^6 \tag{6.53} $$

where ${}^{i}\boldsymbol{s}_{oi}$ is the 3D position vector of a point on the joint axis and ${}^{i}\hat{\boldsymbol{s}}_i$ is the unit direction vector of the joint axis both with respect to the body frame $\{B_i\}$. Further, we denote the reference configuration of the body i with respect to the parent body $\lambda(i)$ in the zero configuration with the homogenous transformation matrix ${}^{\lambda(i)}\boldsymbol{B}_i \in SE(3)$. This constitutes all the geometric information that is needed to compute the kinematics of the tree and is readily available from a CAD model of the robot.

Let us define a center of mass (COM) frame of body i with $\{C_i\}$ and express the spatial mass-inertia matrix of the body with respect to this COM frame with $\boldsymbol{M}_{ci}$ given by:

$$ \boldsymbol{M}_{ci} = \begin{bmatrix} \boldsymbol{I}_{ci} & \boldsymbol{0} \\ \boldsymbol{0} & m_i\boldsymbol{I} \end{bmatrix} \tag{6.54} $$

where $m_i \in \mathbb{R}^+$ is the mass of the body i and $\boldsymbol{I}_{ci} \in \mathbb{R}^{3\times3}$ is its inertia matrix expressed in COM frame. Let ${}^{bi}\boldsymbol{T}_{ci} = ({}^{bi}\boldsymbol{R}_{ci}, {}^{bi}\boldsymbol{c}_{ci})$ denote the pose of the COM frame in the body frame for a body i. Using Equation 6.45, the spatial mass-inertia matrix of the body in the body frame $\boldsymbol{M}_{bi}$ can be computed as:

$$\begin{aligned}\boldsymbol{M}_{bi} &= \left[\mathbf{Ad}_{{}^{bi}\boldsymbol{T}_{ci}}\right]^{-T} \boldsymbol{M}_{ci} \left[\mathbf{Ad}_{{}^{bi}\boldsymbol{T}_{ci}}\right]^{-1} \\ &= \begin{bmatrix} \boldsymbol{I}_{bi} & m_i \left[{}^{bi}\boldsymbol{c}_{ci}\right] \\ -m_i \left[{}^{bi}\boldsymbol{c}_{ci}\right] & m_i \boldsymbol{I} \end{bmatrix}\end{aligned} \tag{6.55}$$

where $\boldsymbol{I}_{bi} = {}^{bi}\boldsymbol{R}_{ci}\boldsymbol{I}_{ci}{}^{bi}\boldsymbol{R}_{ci}^T - m_i \left[{}^{bi}\boldsymbol{c}_{ci}\right]^2 \in \mathbb{R}^{3\times3}$ is the inertia matrix of the body in the body frame (a consequence of parallel axis theorem). This constitutes all the physical information that is needed to compute the dynamics of the kinematic chain and can be extracted from a CAD model of a robot.

6.4.2 Forward Recursion

In the forward recursion, the spatial motion state of the bodies are calculated moving forward from the base link to the tip link ($i = 1, \ldots, n$). Denote with $\boldsymbol{V}_i = \left(\boldsymbol{\omega}_i^T, \boldsymbol{v}_i^T\right)^T \in se(3)$ the twist vector of body i, and with $\boldsymbol{J}_i$ the instantaneous screw coordinate vector of joint i, both in spatial representation.

The first step is to compute the pose of the body in the base frame. This can be done recursively thanks to the body fixed version of the POE formula.

$$\boldsymbol{T}_i = \boldsymbol{T}_{\lambda(i)}{}^{\lambda(i)}\boldsymbol{B}_i \exp({}^i\boldsymbol{S}_i q_i) \tag{6.56}$$

Next, the instantaneous screw coordinate vector $\boldsymbol{J}_i$ is computed in the base frame.

$$\boldsymbol{J}_i = \mathbf{Ad}_{\boldsymbol{T}_i}({}^i\boldsymbol{S}_i) \tag{6.57}$$

Then, the spatial velocity of body i is computed utilizing the spatial velocity of the previous body and velocity of the current joint.

$$\boldsymbol{V}_i = \boldsymbol{V}_{\lambda(i)} + \boldsymbol{J}_i \dot{q}_i \tag{6.58}$$

Finally, the spatial acceleration of the body is computed utilizing the spatial acceleration of the previous body and the acceleration across the joint which makes use of instantaneous screw coordinate computed in Equation 6.57 and the Lie brackets/adjoint mapping.

$$\dot{\boldsymbol{V}}_i = \dot{\boldsymbol{V}}_{\lambda(i)} + \boldsymbol{J}_i \ddot{q}_i + \mathbf{ad}_{\boldsymbol{V}_{\lambda(i)}} \boldsymbol{V}_i \tag{6.59}$$

6.4.3 Backward Recursion

Once we have the spatial velocities and accelerations of all the bodies computed moving forward from base to the tip, we can calculate the joint torques or forces by moving backwards from the tip to base link ($i = n, \dots, 1$). Denote with $\boldsymbol{W}_i = \left(\boldsymbol{m}_i^T, \boldsymbol{f}_i^T\right)^T \in se^*(3)$ the spatial wrench vector of body i.

The first step in the backward recursion is to compute the spatial mass-inertia matrix in the base frame $\boldsymbol{M}_i$ from spatial mass-inertia matrix in the body frame $\boldsymbol{M}_{bi}$.

$$\boldsymbol{M}_i = [\mathbf{Ad}_{\boldsymbol{T}_i}]^{-T} \boldsymbol{M}_{bi} [\mathbf{Ad}_{\boldsymbol{T}_i}]^{-1} \tag{6.60}$$

The wrench acting on any single rigid body i due to its motion $(\boldsymbol{V}_i, \dot{\boldsymbol{V}}_i)$ is given by Equation 6.44. However, the total wrench acting on the body i is the sum of the wrench transmitted through the joint, wrench applied to it through its children links and net external wrench acting on that body resolved in the base frame.

$$\boldsymbol{W}_i = \sum_{j \in \mu(i)} \boldsymbol{W}_j + \boldsymbol{M}_i \dot{\boldsymbol{V}}_i - \mathbf{ad}_{\boldsymbol{V}_i}^T \boldsymbol{M}_i \boldsymbol{V}_i + \boldsymbol{W}_i^{\text{ext}} \tag{6.61}$$

The final step is to compute the force/torque needed by the actuator τ_i from the wrench acting on the body $\boldsymbol{W}_i$.

$$\tau_i = \boldsymbol{J}_i^T \boldsymbol{W}_i \tag{6.62}$$

6.4.4 Computational Complexity

The approach presented in this section is also known as the spatial version of inverse dynamics algorithm. It has been argued in [Müller, 2018] that this version of the algorithm has the same $O(n)$ complexity as the inverse dynamics algorithm in body coordinates presented in [Featherstone, 2008]. However, this version of the algorithm is much more simpler and elegant when compared with the body coordinates version.

6.5 EOM in Closed Form

In this section it is shown how the spatial representation of the recursive Newton-Euler algorithm for inverse dynamics presented in Section 6.4 can be organized into a set of dynamics equations in closed form in generalized coordinates.

$$\boldsymbol{M}(\boldsymbol{q})\ddot{\boldsymbol{q}} + \boldsymbol{C}(\boldsymbol{q}, \dot{\boldsymbol{q}})\dot{\boldsymbol{q}} + \boldsymbol{g}(\boldsymbol{q}) + \boldsymbol{\tau}_{\text{ext}} = \boldsymbol{\tau} \tag{6.63}$$

Here $\boldsymbol{q}, \dot{\boldsymbol{q}}, \ddot{\boldsymbol{q}}$ are $(n \times 1)$ vectors of joint position, velocity and acceleration variables of the system, $\boldsymbol{M}(\boldsymbol{q})$ is the $(n \times n)$ generalized mass-inertia matrix, $\boldsymbol{C}(\boldsymbol{q}, \dot{\boldsymbol{q}})$ is a $(n \times n)$

matrix for Coriolis-centrifugal forces, $\boldsymbol{g}(\boldsymbol{q})$ is the $(n \times 1)$ vector of gravity efforts, $\boldsymbol{\tau}_{\text{ext}}$ is the $(n \times 1)$ vector of external forces projected on the tree joints, and $\boldsymbol{\tau}$ is the $(n \times 1)$ vector of force/torque variables.

The first step in RNEA is the recursive computation of twist of each body using Equation 6.58. Similar to Equation 6.16, this could be written in closed form as:

$$\boldsymbol{V}^{\text{sys}} = \boldsymbol{J}^{\text{sys}}\dot{\boldsymbol{q}} + \boldsymbol{V}^{\text{sys}}_{\text{base}} = \boldsymbol{A}^{\text{sys}}\boldsymbol{S}^{\text{sys}}\dot{\boldsymbol{q}} + \boldsymbol{V}^{\text{sys}}_{\text{base}} \tag{6.64}$$

where $\boldsymbol{V}^{\text{sys}}_{\text{base}} = [\boldsymbol{V}_{\text{base}}, \boldsymbol{0}, \dots, \boldsymbol{0}]^T \in \mathbb{R}^{6n}$ is the velocity of the system base.

The second step in RNEA is the recursive computation of spatial acceleration of every body using Equation 6.59. Similar to Equation 6.28, this equation can be written in its closed form as:

$$\dot{\boldsymbol{V}}^{\text{sys}} = \dot{\boldsymbol{V}}^{\text{sys}}_{\text{base}} + \boldsymbol{J}^{\text{sys}}\ddot{\boldsymbol{q}} + \boldsymbol{L}^{\text{sys}}\boldsymbol{b}^{\text{sys}}\boldsymbol{J}^{\text{sys}}\dot{\boldsymbol{q}} = \dot{\boldsymbol{V}}^{\text{sys}}_{\text{base}} + \boldsymbol{J}^{\text{sys}}\ddot{\boldsymbol{q}} + \boldsymbol{L}^{\text{sys}}\boldsymbol{b}^{\text{sys}}\boldsymbol{V}^{\text{sys}} \tag{6.65}$$

where $\dot{\boldsymbol{V}}^{\text{sys}}_{\text{base}} = [\dot{\boldsymbol{V}}_{\text{base}}, \boldsymbol{0}, \dots, \boldsymbol{0}]^T \in \mathbb{R}^{6n}$ is the acceleration of the system base.

The third step in RNEA is the recursive computation of wrench acting on each body using Equation 6.61. This equation can be written in its closed form as:

$$\boldsymbol{W}^{\text{sys}} = \boldsymbol{M}^{\text{sys}}\dot{\boldsymbol{V}}^{\text{sys}} - [\boldsymbol{b}^{\text{sys}}]^T\boldsymbol{M}^{\text{sys}}\boldsymbol{V}^{\text{sys}} + \boldsymbol{W}^{\text{sys}}_{\text{ext}} \tag{6.66}$$

where $\boldsymbol{W}^{\text{sys}}_{\text{ext}} = [\boldsymbol{W}^{\text{ext}}_1, \boldsymbol{W}^{\text{ext}}_2, \dots, \boldsymbol{W}^{\text{ext}}_n]^T \in \mathbb{R}^{6n}$ is the vector of external wrenches acting on the system and $\boldsymbol{M}^{\text{sys}} = \text{diag}(\boldsymbol{M}_1, \boldsymbol{M}_2, \dots, \boldsymbol{M}_n) \in \mathbb{R}^{6n \times 6n}$ is the spatial mass-inertia matrix of the system.

The final step in RNEA is to compute the actuation forces/torques using Equation 6.62 which can be written in its closed form as:

$$\boldsymbol{\tau} = [\boldsymbol{J}^{\text{sys}}]^T\boldsymbol{W}^{\text{sys}} \tag{6.67}$$

where $\boldsymbol{\tau}$ is the vector of actuator forces/torques.

After deriving the individual equations in RNEA in closed form, one could start building the closed form Lagrangian equation in generalized coordinates. Substituting Equation 6.66 into Equation 6.67, one gets:

$$\boldsymbol{\tau} = [\boldsymbol{J}^{\text{sys}}]^T\left(\boldsymbol{M}^{\text{sys}}\dot{\boldsymbol{V}}^{\text{sys}} - [\boldsymbol{b}^{\text{sys}}]^T\boldsymbol{M}^{\text{sys}}\boldsymbol{V}^{\text{sys}} + \boldsymbol{W}^{\text{sys}}_{\text{ext}}\right) \tag{6.68}$$

in which, one can again substitute Equation 6.65 to arrive at:

$$\begin{aligned}\boldsymbol{\tau} &= [\boldsymbol{J}^{\text{sys}}]^T \left(\boldsymbol{M}^{\text{sys}} \left[\dot{\boldsymbol{V}}^{\text{sys}}_{\text{base}} + \boldsymbol{J}^{\text{sys}}\ddot{\boldsymbol{q}} + \boldsymbol{L}^{\text{sys}}\boldsymbol{b}^{\text{sys}}\boldsymbol{V}^{\text{sys}}\right] - [\boldsymbol{b}^{\text{sys}}]^T\boldsymbol{M}^{\text{sys}}\boldsymbol{V}^{\text{sys}} + \boldsymbol{W}^{\text{sys}}_{\text{ext}}\right)\\ \text{or },\boldsymbol{\tau} &= [\boldsymbol{J}^{\text{sys}}]^T\boldsymbol{M}^{\text{sys}}\left[\boldsymbol{J}^{\text{sys}}\ddot{\boldsymbol{q}} + \boldsymbol{L}^{\text{sys}}\boldsymbol{b}^{\text{sys}}\boldsymbol{V}^{\text{sys}} + \dot{\boldsymbol{V}}^{\text{sys}}_{\text{base}}\right] - [\boldsymbol{J}^{\text{sys}}]^T[\boldsymbol{b}^{\text{sys}}]^T\boldsymbol{M}^{\text{sys}}\boldsymbol{V}^{\text{sys}}\\ &\qquad + [\boldsymbol{J}^{\text{sys}}]^T\boldsymbol{W}^{\text{sys}}_{\text{ext}}\,.\end{aligned}$$

Substituting Equation 6.64 into the above equation while assuming a system with fixed base i.e. $\boldsymbol{V}^{\text{sys}}_{\text{base}} = \mathbf{0}$ and base acceleration as the gravity vector i.e. $\dot{\boldsymbol{V}}^{\text{sys}}_{\text{base}} = \dot{\boldsymbol{V}}^{\text{sys}}_{g} = [\dot{\boldsymbol{V}}_g, \mathbf{0}, \ldots, \mathbf{0}]^T$, one gets:

$$\begin{aligned}\boldsymbol{\tau} = \underbrace{[\boldsymbol{J}^{\text{sys}}]^T\boldsymbol{M}^{\text{sys}}\boldsymbol{J}^{\text{sys}}}_{\boldsymbol{M}(\boldsymbol{q})}\ddot{\boldsymbol{q}} + \underbrace{[\boldsymbol{J}^{\text{sys}}]^T\left(\boldsymbol{M}^{\text{sys}}\boldsymbol{L}^{\text{sys}}\boldsymbol{b}^{\text{sys}} - [\boldsymbol{b}^{\text{sys}}]^T\boldsymbol{M}^{\text{sys}}\right)\boldsymbol{J}^{\text{sys}}}_{\boldsymbol{C}(\boldsymbol{q},\dot{\boldsymbol{q}})}\dot{\boldsymbol{q}} +\\ \underbrace{[\boldsymbol{J}^{\text{sys}}]^T\boldsymbol{M}^{\text{sys}}\dot{\boldsymbol{V}}^{\text{sys}}_{\text{g}}}_{\boldsymbol{g}(\boldsymbol{q})} + \underbrace{[\boldsymbol{J}^{\text{sys}}]^T\boldsymbol{W}^{\text{sys}}_{\text{ext}}}_{\boldsymbol{\tau}_{\text{ext}}}\,.\end{aligned} \tag{6.69}$$

Hence, the expressions for individual terms in Equation 6.63 are given by:

$$\boldsymbol{M}(\boldsymbol{q}) = [\boldsymbol{J}^{\text{sys}}]^T\boldsymbol{M}^{\text{sys}}\boldsymbol{J}^{\text{sys}} \tag{6.70}$$

$$\boldsymbol{C}(\boldsymbol{q},\dot{\boldsymbol{q}}) = [\boldsymbol{J}^{\text{sys}}]^T\left(\boldsymbol{M}^{\text{sys}}\boldsymbol{L}^{\text{sys}}\boldsymbol{b}^{\text{sys}} - [\boldsymbol{b}^{\text{sys}}]^T\boldsymbol{M}^{\text{sys}}\right)\boldsymbol{J}^{\text{sys}} \tag{6.71}$$

$$\boldsymbol{g}(\boldsymbol{q}) = [\boldsymbol{J}^{\text{sys}}]^T\boldsymbol{M}^{\text{sys}}\dot{\boldsymbol{V}}^{\text{sys}}_{\text{g}} \tag{6.72}$$

$$\boldsymbol{\tau}_{\text{ext}} = [\boldsymbol{J}^{\text{sys}}]^T\boldsymbol{W}^{\text{sys}}_{\text{ext}} \tag{6.73}$$

where the system Jacobian can be further factorized and written as $\boldsymbol{J}^{\text{sys}} = \boldsymbol{A}^{\text{sys}}\boldsymbol{S}^{\text{sys}}$ as shown in Equation 6.17.

6.6 Conclusion

This chapter presents the screw theory and Lie group theory based approaches for the kinematic and dynamic modeling of serial or tree type robotic systems. Starting with a topological description of a kinematic chain using graph theoretic concepts, the recursive relations for computing kinematics are presented. Then equations of motion for a single rigid body dynamics are presented from both classical and modern viewpoints in order to highlight the advantage of the modern geometric setting. Finally, a fully recursive $O(n)$ algorithm for computing the inverse dynamics of a kinematic tree is presented and further utilized to develop EOM in closed form. This chapter hence provides the theoretical background needed to understand the kinematics and dynamics of more complex series-parallel hybrid robotic systems which will be discussed in the next chapter.

Chapter 7

Modular and Analytical Methods for Series-Parallel Hybrid Systems

This chapter presents the kinematics and dynamics of series-parallel hybrid robots based on the theoretical foundations presented in the previous chapter for tree type systems. The main contribution here is an analytical and modular methodology to compute the kinematics and dynamics of arbitrary hybrid robots that are a serial composition of serial or parallel submechanism modules. The approach allows modular composition of the analytical loop closure functions (LCF) and its associated derivatives and transfers them into computation of the analytical forward and inverse dynamics algorithms. The benefit of this approach is that efficient and recursive $O(n)$ dynamics algorithms for tree type systems can be directly used. The results are free of numerical errors resulting from loop closure and free of singularities arising from redundant constraints. This approach forms the basis of the Hybrid Robot Dynamics (HyRoDyn) software framework which will be described in the next chapter. The explanation in this chapter is aided with an example of a series-parallel hybrid humanoid robot developed at DFKI-RIC which employs several different closed loop and parallel mechanism based modules. The content presented here is based on [Kumar and Mueller, 2019] and was first presented in the form of a poster [Kumar et al., 2018d].

This chapter is organized as follows: Section 7.1 provides theoretical preliminaries for modeling robots with closed loops along with an introduction to the concept of loop closure functions. It derives the formulas for the forward and inverse dynamics for these mechanisms. Section 7.2 introduces the notion of modularity and guidelines for selecting submechanism modules in series-parallel hybrid robotic systems. Section 7.3 presents the modular graph based topological modeling of series-parallel

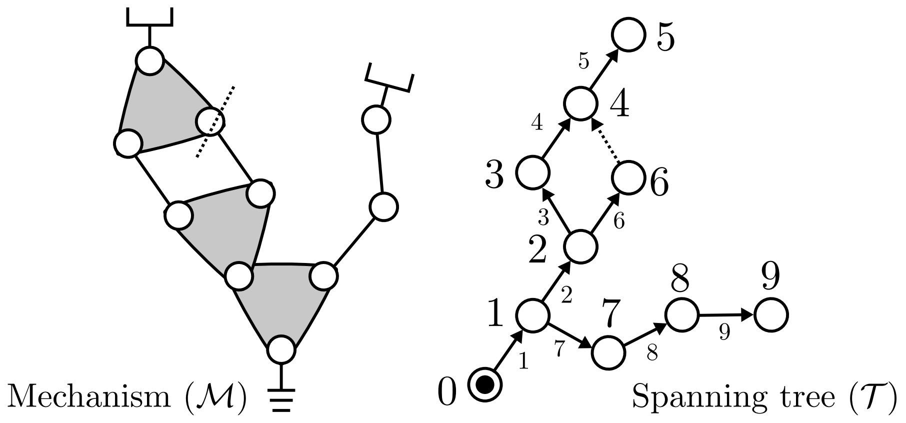


Figure 7.1: A closed loop robot and its associated spanning tree numbered using regular numbering scheme. Here, links and joints are denoted as vertices and edges respectively.

hybrid robots using the previously identified submechanism modules. Section 7.4 elaborates the analytical and modular approach for kinematic and dynamics modeling of such series-parallel hybrid systems. It further comments on their computational complexity. Section 7.5 draws the conclusions of this chapter and makes a bridge to the software implementation of this approach which is discussed in next part of this thesis.

7.1 Modeling Rigid Body Systems with Closed Loops

This section briefly introduces the theory of multi-body dynamics subjected to holonomic and sceleronomic constraints. It mostly adopts the notation and terminology introduced by Featherstone in [Featherstone, 2008]. Consider a rigid body system with N_B bodies, N_J joints, and $N_L = N_J - N_B$ kinematic loops. Assume that a spanning tree is defined and that the joints are enumerated using regular numbering scheme (see Fig. 7.1 for example). Let n denote the degree of freedom of the selected spanning tree, computed as $n = \sum_{i=1}^{N_B} n_i$, and let n_c denote the number of loop-closure constraints, computed as $n_c = \sum_{k=N_B+1}^{N_J} n_{ck}$ where n_{ck} denotes the number of loop constraints imposed by k^{th} cut joint. Further, let $\boldsymbol{q}$ indicate the vector of all joints of the spanning tree (of size n) and let $\boldsymbol{y}$ indicate the vector of all independent variables (of size $n - n_c$).

7.1.1 Loop Constraints

Loop constraints are non-linear constraints on the motion variables of a multi-body system. Loop constraints can be expressed in an implicit and in an explicit way, they are summarized in Table 7.1 at position, velocity, and acceleration levels. Here let $\boldsymbol{K} = \frac{\partial \phi}{\partial \boldsymbol{q}}$, $\boldsymbol{k} = -\dot{\boldsymbol{K}}\dot{\boldsymbol{q}}$, $\boldsymbol{G} = \frac{\partial \gamma}{\partial \boldsymbol{y}}$, and $\boldsymbol{g} = \dot{\boldsymbol{G}}\dot{\boldsymbol{y}}$. If both functions ϕ and γ describe the same constraint, $\phi \circ \gamma = \boldsymbol{0}$, $\boldsymbol{KG} = \boldsymbol{0}$, and $\boldsymbol{Kg} = \boldsymbol{k}$ can be deduced. Algorithms to compute variables in Table 7.1 from the spanning tree are provided in [Featherstone, 2008] and skipped here for brevity.

Table 7.1: Loop constraints [Featherstone, 2008]

Type	position	velocity	acceleration
implicit:	$\phi(\boldsymbol{q}) = \boldsymbol{0}$	$\boldsymbol{K}\dot{\boldsymbol{q}} = \boldsymbol{0}$	$\boldsymbol{K}\ddot{\boldsymbol{q}} = \boldsymbol{k}$
explicit:	$\boldsymbol{q} = \gamma(\boldsymbol{y})$	$\dot{\boldsymbol{q}} = \boldsymbol{G}\dot{\boldsymbol{y}}$	$\ddot{\boldsymbol{q}} = \boldsymbol{G}\ddot{\boldsymbol{y}} + \boldsymbol{g}$

7.1.2 Equations of Motion (EOM)

The equations of motion for a tree topology multi-body system can be written as

$$\boldsymbol{\tau} = \boldsymbol{M}(\boldsymbol{q})\ddot{\boldsymbol{q}} + \boldsymbol{C}(\boldsymbol{q}, \dot{\boldsymbol{q}}) \tag{7.1}$$

where $\boldsymbol{q}, \dot{\boldsymbol{q}}, \ddot{\boldsymbol{q}}$ are $(n \times 1)$ vectors of joint position, velocity and acceleration variables of the spanning tree, $\boldsymbol{M}(\boldsymbol{q})$ is the $(n \times n)$ generalized mass-inertia matrix, $\boldsymbol{C}(\boldsymbol{q}, \dot{\boldsymbol{q}})$ is a $(n \times 1)$ vector for Coriolis-centrifugal and gravity efforts, and $\boldsymbol{\tau}$ is the $(n \times 1)$ vector of force/torque variables. In case of robots with closed loops, the equivalent spanning tree of the robot system is subjected to loop constraint forces

$$\boldsymbol{M}(\boldsymbol{q})\ddot{\boldsymbol{q}} + \boldsymbol{C}(\boldsymbol{q}, \dot{\boldsymbol{q}}) = \boldsymbol{\tau} + \boldsymbol{\tau}_a + \boldsymbol{\tau}_c \tag{7.2}$$

where $\boldsymbol{\tau}_c$ and $\boldsymbol{\tau}_a$ are the constraint and active forces, respectively produced by the cut joints. If the selected cut joint is passive, $\boldsymbol{\tau}_a = 0$ can be substituted in Equation 7.2. The constraint force $\boldsymbol{\tau}_c$ is usually unknown but its value can either be calculated or eliminated from the equation following the Jourdain's principle [Piedboeuf, 1993] of virtual power, i.e., $\boldsymbol{\tau}_c\dot{\boldsymbol{q}} = 0$. Based on the (implicit or explicit) nature of the loop constraints, the equations of motion are developed for the entire system.

7.1.2.1 EOM with Implicit Loop Constraints

The cut joints impose a set of kinematic constraints on the spanning tree which are briefly introduced in Table 7.1. Assuming that the implicit position level constraints have been successfully differentiated twice, the acceleration level loop constraints can be collected in a single matrix equation of the form

$$\boldsymbol{K}\ddot{\boldsymbol{q}} = \boldsymbol{k} \tag{7.3}$$

where $\boldsymbol{K}$ is a $(n_c \times n)$ matrix. If the system is subjected to an implicit loop constraint then it can be shown that $\boldsymbol{\tau}_c$ takes the form

$$\boldsymbol{\tau}_c = \boldsymbol{K}^T \boldsymbol{\lambda} \tag{7.4}$$

where $\boldsymbol{\lambda}$ is the vector of Lagrangian multipliers. Combining Equations 7.2, 7.3 and 7.4, the equation of motion of the overall system taking into account implicit loop constraints can be written as

$$\begin{bmatrix} \boldsymbol{M} & \boldsymbol{K}^T \\ \boldsymbol{K} & \boldsymbol{0} \end{bmatrix} \begin{bmatrix} \ddot{q} \\ -\boldsymbol{\lambda} \end{bmatrix} = \begin{bmatrix} \boldsymbol{\tau} - \boldsymbol{C} + \boldsymbol{\tau}_a \\ \boldsymbol{k} \end{bmatrix} . \tag{7.5}$$

This is a system of $(n + n_c)$ equations in $(n + n_c)$ unknowns. The coefficient matrix of dimension $(n + n_c) \times (n + n_c)$ in Equation 7.5 is symmetric but not positive definite. If the rank of matrix $r = \text{rank}(\boldsymbol{K})$ is less than n_c, then the coefficient matrix becomes singular and the system is said to be over-constrained. Over-constrained systems are actually very common. For example, planar kinematic loops impose redundant constraints on the system – that either need to be removed manually [Featherstone, 2008] or demand numerical decomposition techniques which deteriorate the computational performance and numerical accuracy of the solution [Mueller, 2014].

7.1.2.2 EOM with Explicit Loop Constraints

Using explicit velocity level loop constraint (refer to Table 7.1) and Jourdain's principle of virtual power, one can establish that $\boldsymbol{\tau}_c$ will have the following property

$$\boldsymbol{G}^T \boldsymbol{\tau}_c = 0 . \tag{7.6}$$

Similarly, one can write the explicit motion constraints at an acceleration level

$$\ddot{q} = \boldsymbol{G}\ddot{y} + \boldsymbol{g} . \tag{7.7}$$

Combining Equations 7.2, 7.6 and 7.7, the equation of motion taking into account explicit loop constraints can be developed.

$$\begin{bmatrix} \boldsymbol{M} & -\boldsymbol{I} & \boldsymbol{0} \\ -\boldsymbol{I} & \boldsymbol{0} & \boldsymbol{G} \\ \boldsymbol{0} & \boldsymbol{G}^T & \boldsymbol{0} \end{bmatrix} \begin{bmatrix} \ddot{q} \\ \boldsymbol{\tau}_c \\ \ddot{y} \end{bmatrix} = \begin{bmatrix} \boldsymbol{\tau} - \boldsymbol{C} + \boldsymbol{\tau}_a \\ -\boldsymbol{g} \\ \boldsymbol{0} \end{bmatrix} \tag{7.8}$$

7.1.3 Loop Closure Functions

It is usually more complex to deal with robots involving closed loops. In contrast to a tree type robot, the mobility of closed loop system is dependent on r which can vary with the configuration. Also, the different assembly modes can lead to configuration ambiguities. Thus, it is useful to derive explicit functions for the modeling of closed loop systems whenever and wherever possible.

Let us define loop closure functions such that they provide a unique mapping between the independent position variables $\boldsymbol{y}$ and position variables in the spanning tree $\boldsymbol{q}$. The set of independent joint variables $\boldsymbol{y}$ may or may not be a subset of the position variables of the spanning tree $\boldsymbol{q}$. For this assumption to hold true, let us define a set $C \subseteq \mathbb{R}^{n-r}$ of acceptable values of $\boldsymbol{y}$, and assume $\boldsymbol{y} \in C$. For all $\boldsymbol{y} \in C$, there exists a function, γ such that

$$\begin{aligned} \boldsymbol{q} &= \gamma(\boldsymbol{y}) \\ \dot{\boldsymbol{q}} &= \boldsymbol{G}\dot{\boldsymbol{y}} \\ \ddot{\boldsymbol{q}} = \boldsymbol{G}\ddot{\boldsymbol{y}} + \dot{\boldsymbol{G}}\dot{\boldsymbol{y}} &= \boldsymbol{G}\ddot{\boldsymbol{y}} + \boldsymbol{g} \end{aligned} \tag{7.9}$$

It must be noted that the above relations are identical to the explicit loop constraint equations noted in Table 7.1 when the loop closure errors are zero. This method is less generic in nature than the ones described before because it is not always possible to find such a mapping analytically. However, the advantages of an analytical solution to the loop closure constraints outweighs the manual effort needed because numerical loop closure errors can not occur. Also, there is no need to introduce a constraint stabilization term unlike when practically dealing with Equation 7.3. Further, the spanning tree can be selected such that there are no active forces on the cut joints i.e. $\boldsymbol{\tau}_a = 0$.

7.1.4 Forward and Inverse Dynamics

The equations of motion presented above could be either solved for independent joint accelerations $\ddot{\boldsymbol{y}}$ under given actuator force conditions or for the actuator forces $\boldsymbol{u}$ required to generate given acceleration. The former is called the forward dynamics problem and the latter is called the inverse dynamics problem [Featherstone, 2008, Jain, 2011b]. In the following, the forward and inverse dynamics formula are derived such that a link between the loop closure functions and spanning tree dynamics is established.

7.1.4.1 Forward Dynamics Solution

By using Equation 7.6 and multiplying by G^T on both sides of Equation 7.2, the loop constraint forces $\boldsymbol{\tau}_c$ can be eliminated.

$$\boldsymbol{G}^T\boldsymbol{M}\ddot{\boldsymbol{q}} = \boldsymbol{G}^T(\boldsymbol{\tau} - \boldsymbol{C}) \tag{7.10}$$

Substituting Equation 7.7 in Equation 7.10 and simplifying one can arrive at the solution to the forward dynamics problem:

$$\ddot{\boldsymbol{y}} = (\boldsymbol{G}^T\boldsymbol{M}\boldsymbol{G})^{-1}(\boldsymbol{G}^T\boldsymbol{\tau} - \boldsymbol{G}^T(\boldsymbol{C} + \boldsymbol{M}\boldsymbol{g}))\ . \tag{7.11}$$

7.1.4.2 Inverse Dynamics Solution

Equation 7.10 could be rewritten as:

$$\boldsymbol{G}^T\boldsymbol{\tau} = \boldsymbol{G}^T(\boldsymbol{M}\ddot{\boldsymbol{q}} + \boldsymbol{C}) = \boldsymbol{G}^T\boldsymbol{\tau}_{ID} \tag{7.12}$$

where τ_{ID} is the inverse dynamics output of a spanning tree given by

$$\boldsymbol{\tau}_{ID} = \boldsymbol{M}(\gamma(\boldsymbol{y}))(\boldsymbol{G}\ddot{\boldsymbol{y}} + \boldsymbol{g}) + \boldsymbol{C}(\gamma(\boldsymbol{y}), \boldsymbol{G}\dot{\boldsymbol{y}})\ . \tag{7.13}$$

The solution to Equation 7.12 is not unique because G^T is an $(n-r) \times n$ matrix which imposes $(n-r)$ constraints on an n dimensional vector of unknowns, leaving r freedoms of choice. In other words, there are ∞^r different values of $\boldsymbol{\tau}$ which will produce the same acceleration. To arrive at a unique solution, the actuated degrees of freedom must be separated from the passive degrees of freedom. This can be done with the help of a matrix $\boldsymbol{G}_u$ which basically contain the rows of $\boldsymbol{G}$ corresponding to the actuated degrees of freedom. If the rank of matrix $\boldsymbol{G}_u$ is equal to $(n-r)$, then the system is properly actuated[1] and a unique solution to the inverse dynamics problem can be found which is given by:

$$\boldsymbol{\tau}_u = \boldsymbol{G}_u^{-T}\boldsymbol{G}^T\boldsymbol{\tau}_{ID} \tag{7.14}$$

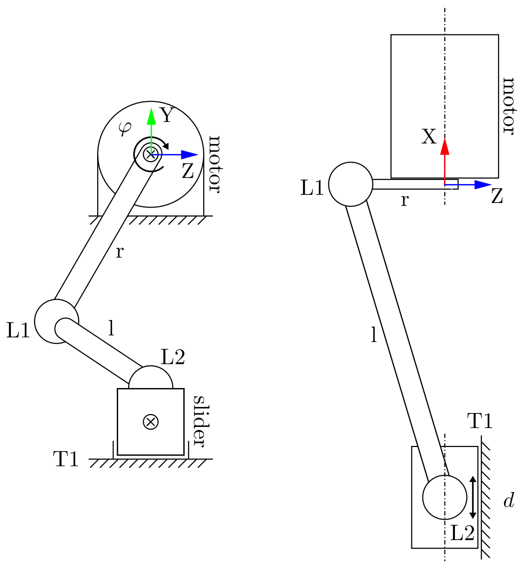
where $\boldsymbol{\tau}_u$ is a vector of actuator forces required to produce the given acceleration $\ddot{\boldsymbol{y}}$.

7.1.5 Comparison of Numerical and Analytical Resolution of Loop Constraints: Case Study of a 3D Slider Crank Mechanism

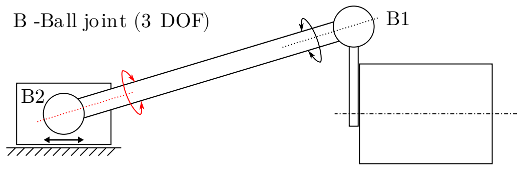
To make a comparison between numerical and analytical resolution of loop closure constraints, we present the case study of a 3D slider crank mechanism. This mecha-

[1]If $p > \text{rank}(\boldsymbol{G}_u)$, the system is redundantly actuated and $\boldsymbol{\tau}_u = \boldsymbol{G}_u^{\dagger T}\boldsymbol{G}^T\boldsymbol{\tau}_{ID}$

nism consists of a crank, a connecting rod and a slider. Fig. 7.2a shows the schematic of the 3D slider crank linkage. The crank is driven by a motor with its rotational axis parallel to global X-axis. The slider makes a sliding movement on the ground with its translational axis coinciding with the global X-axis. Those two parts are connected via the connecting rod that itself is connected to the two parts by use of two ZYX ball joints. Since, the system has two spherical joints (SS pair) on the connecting rod, the system has one redundant rotational DOF around the rod's axis (see Fig. 7.2b). The input motor angle is denoted with φ and the output slider movement is denoted with d. An input movement trajectory, $\varphi = \pi t^2$, is provided to the mechanism for a simulation time of 1 second in 100 time steps. The numerical solution is obtained using the multi-body simulation tool called ADAMS and the analytical solution is implemented in the HyRoDyn software framework. The analytical equations for the loop closure function are skipped here for brevity.



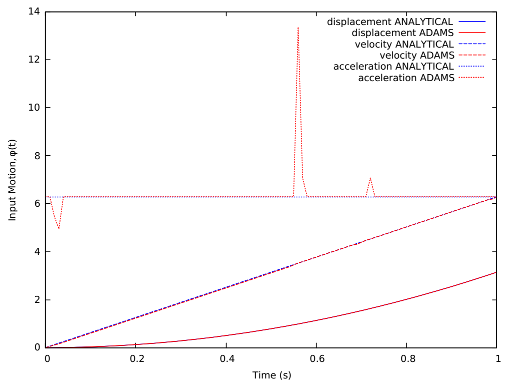
(a) Schematic



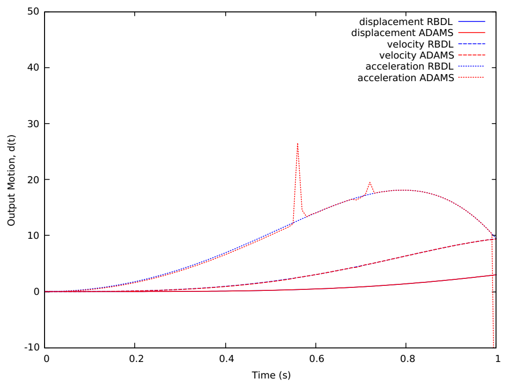
(b) Redundant constraint in SS pair

Figure 7.2: 3D slider crank mechanism

Fig. 7.3 shows the input and output motion trajectories plotted using HyRoDyn and ADAMS respectively. It can be noticed that the plots produced by ADAMS are subjected to some numerical jitters. Further, the results from the inverse dynamics analysis is plotted (see Fig. 7.4). To study the effect of redundant constraints on the quality of numerical solution, two different cases of this mechanism with same physical dimensions are studied: one with SU pair and the other SS pair on the connecting rod. As one can expect, the mechanism with SU pair should match better with the analytical solver. However, we found that the quality of the numerical solution also depends on whether the inertia term around the rod's redundant axis (highlighted in red in Fig. 7.2b) is defined. Fig. 7.4 (a) shows the best match between the analytical solution and numerical solution obtained from ADAMS for a model with SU pair on the rod and inertia around rod's axis is set to zero. The solution becomes unstable when there is a non-zero inertia defined for the rod around this axis (Fig. 7.4 (b)). Fig. 7.4 (c) and (d) demonstrate the case with SS pair on the rod with and without zero inertia. Both of these plots are subjected to numerical jitters. Further, it was found that the computational performance of the analytical solver was much better than the numerical solver. Thus, it can be concluded that analytical solutions to loop closure constraints, when available, always outperform their numerical counterparts.



(a) Input motion



(b) Output motion

Figure 7.3: Input and output motion of 3D slider crank mechanism in HyRoDyn and ADAMS

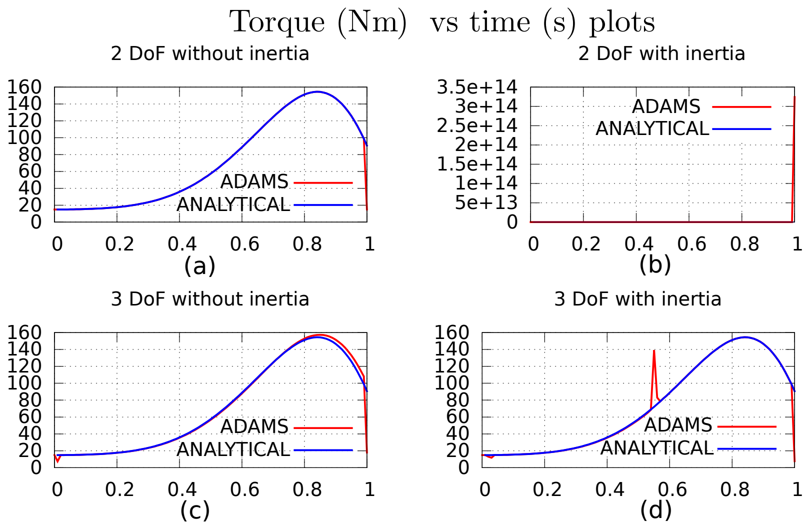


Figure 7.4: Analytical (HyRoDyn) and numerical (ADAMS) inverse dynamics analyses of 3D slider crank mechanism

7.2 Notion of Modularity

Hybrid robots are robots that can be seen as a serial composition of serial or parallel submechanisms modules. Table 2.1 and Table 2.2 of Chapter 2 present a survey on closed loop kinematics and parallel submechanism modules which have been utilized in series-parallel hybrid robot designs in the last decade. Such a design methodology is biologically inspired (see Fig. 7.5) as most joints found in the biological systems are actuated with muscle groups in a parallel architecture. This allows the exploitation of the non-linear transmission from the actuation space to the task space and provide better actuator placement possibilities which can optimise the mass-inertia properties of the limbs. Two observations can be made from this survey:

- submechanism modules are used as a mechanical generator of a motion subspace (revolute, universal, spherical, free joint etc.)
- the same type of submechanism with different physical parameters is utilized as a module to serve different purposes (ankle, wrist, torso joints etc.) in the same robot.

The analysis of closed loop mechanisms is difficult and hence they require special treatments in contrast to tree type systems. Numerical resolution of the loop closure constraints can lead to inaccuracy and poor performance as described in the previous section. Analytical resolution of loop closure constraints is worth the effort if it can be made reusable as this can help in dealing with a large number of similar loop constraints inside a hybrid robot. This inspires an analytical treatment of loop closure constraints for a submechanism module in a robot assembly so that the modularity in the hardware design is also reflected in the kinematic and dynamic modeling of the robot.

7.2.1 Definition of Submechanism Module

A submechanism module ($\mathcal{M}_i$) is defined as a set of links and joints which can produce any motion from m-dimensional motion subspace of $SE(3)$ while demonstrating properties of a *minimal loop cluster*. A loop cluster is any set of loops with the property that no loop within the cluster is coupled to any other loop outside it; and a minimal loop cluster is the one that can not be divided into two small loop clusters. In other words, these loop clusters do not share a common edge (joint). A serial combination of links and joints can be seen as a submechanism with no closed loops. This definition helps us in two different ways:

- it reveals the block diagonal structure in the constraint Jacobian matrix ($\boldsymbol{G}$) and its derivative ($\dot{\boldsymbol{G}}$) which can lead to efficiency in kinematics and dynamics

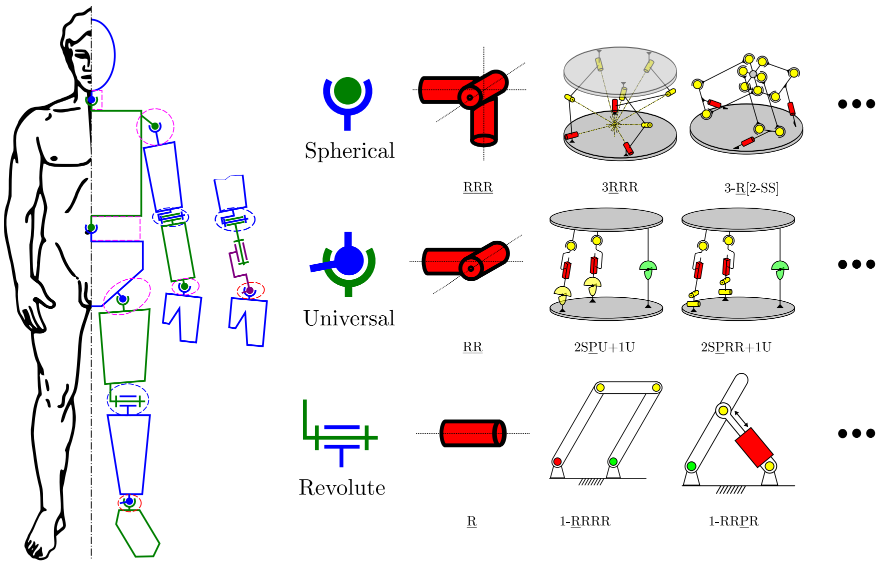


Figure 7.5: Biologically inspired series-parallel hybrid humanoid design

computations,

- and, it helps in generation of simpler abstract mechanisms which do not further contain any closed loops and have physical meaning.

To define a submechanism module, it is crucial to define three subsets of joints from the connectivity graph ($\mathcal{G}_i$) of a mechanism:

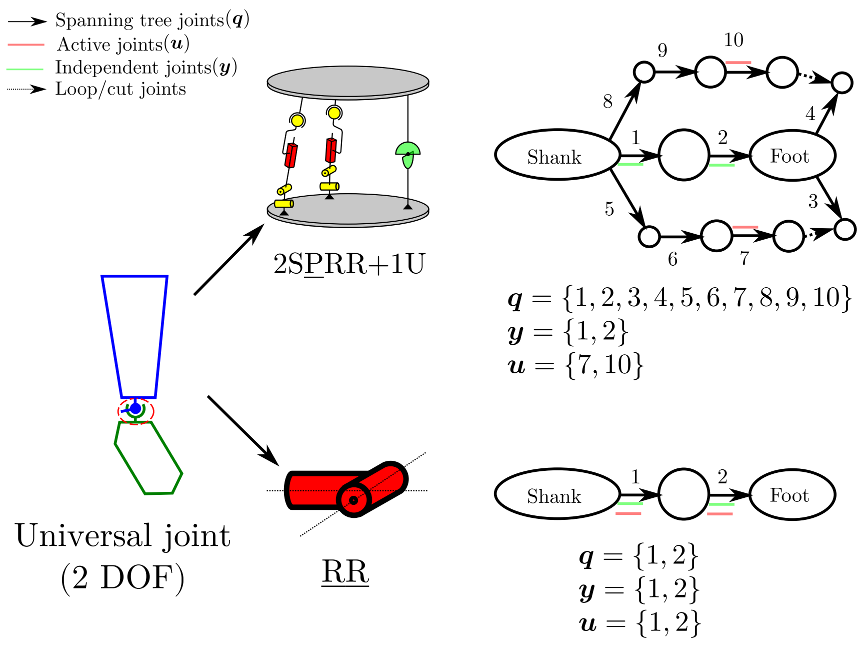
- spanning tree joints ($\boldsymbol{q}_i \in \mathbb{R}^{n_i}$): all the joints belonging to the spanning tree ($\mathcal{T}_i$) chosen by regular numbering scheme,
- independent joints ($\boldsymbol{y}_i \in \mathbb{R}^{m_i}$): a set of independent variables selected such that $\boldsymbol{y}_i$ defines $\boldsymbol{q}_i$ uniquely,
- active joints ($\boldsymbol{u}_i \in \mathbb{R}^{p_i}$): all the joints containing the actuators

Let us define a selection matrix, $\boldsymbol{Q}_i \in \mathbb{R}^{p_i \times n_i}$ such that $\boldsymbol{u}_i = \boldsymbol{Q}_i \boldsymbol{q}_i$. Also, in the scope of the current work, it is assumed the submechanism modules are *properly actuated* (i.e. $p_i = m_i$) and *properly constrained* (i.e. $m_i = n_i - n_{ci}$).

7.2.2 Guidelines for Selection

While the definition of a submechanism module is intentionally chosen not to be too restrictive, some guidelines can be followed while selecting the submechanism modules:

- It is evident that the choice of independent coordinates for a parallel submechanism module is non-unique. The independent joints in a submechanism module



Spanning tree joints($\boldsymbol{q}$)
Active joints($\boldsymbol{u}$)
Independent joints($\boldsymbol{y}$)
Loop/cut joints
2SPRR+1U
Universal joint (2 DOF)
RR
Shank
Foot
1
2
3
4
5
6
7
8
9
10
$\boldsymbol{q} = \{1,2,3,4,5,6,7,8,9,10\}$
$\boldsymbol{y} = \{1,2\}$
$\boldsymbol{u} = \{7,10\}$
$\boldsymbol{q} = \{1,2\}$
$\boldsymbol{y} = \{1,2\}$
$\boldsymbol{u} = \{1,2\}$

Figure 7.6: Example of submechanism definition

should be chosen such that they represent the operational space or platform coordinates of the submechanism module because it is easier to solve the inverse kinematics of parallel mechanisms analytically in comparison to forward kinematics. Further, it is possible to integrate sensors to measure these coordinates directly in many cases (see Table 2.1). For serial submechanism modules, the choice of independent coordinates must be the same as its configuration space coordinates.

- A careful choice of submechanism module can help in exploiting the hierarchy in the robot design through abstractions. The two categories of parallel submechanism modules used in the design of series-parallel hybrid robots (see Table 2.1 and Table 2.2) always encapsulate a known joint type (for e.g. revolute, universal, spherical etc.). Overall, the serial nature of hybrid robot design may simply be following well studied 6 or 7 DOF anthropomorphic limb designs such as URS, SRS etc. Hence, analytical inverse kinematics solutions for such limbs can be easily mapped to the series-parallel hybrid robot and therefore enable the analytical computation of kinematics and dynamics of the complete system.

Example 2 *The designer may construct the ankle joint of a humanoid robot either using a set of two serially connected actuators (e.g. 2R orthogonal arrangement) for the sake of simplicity or utilize a parallel mechanism (e.g. 2SPRR+1U [Kumar et al., 2018c]) for minimizing the lower leg inertia and exploit-*

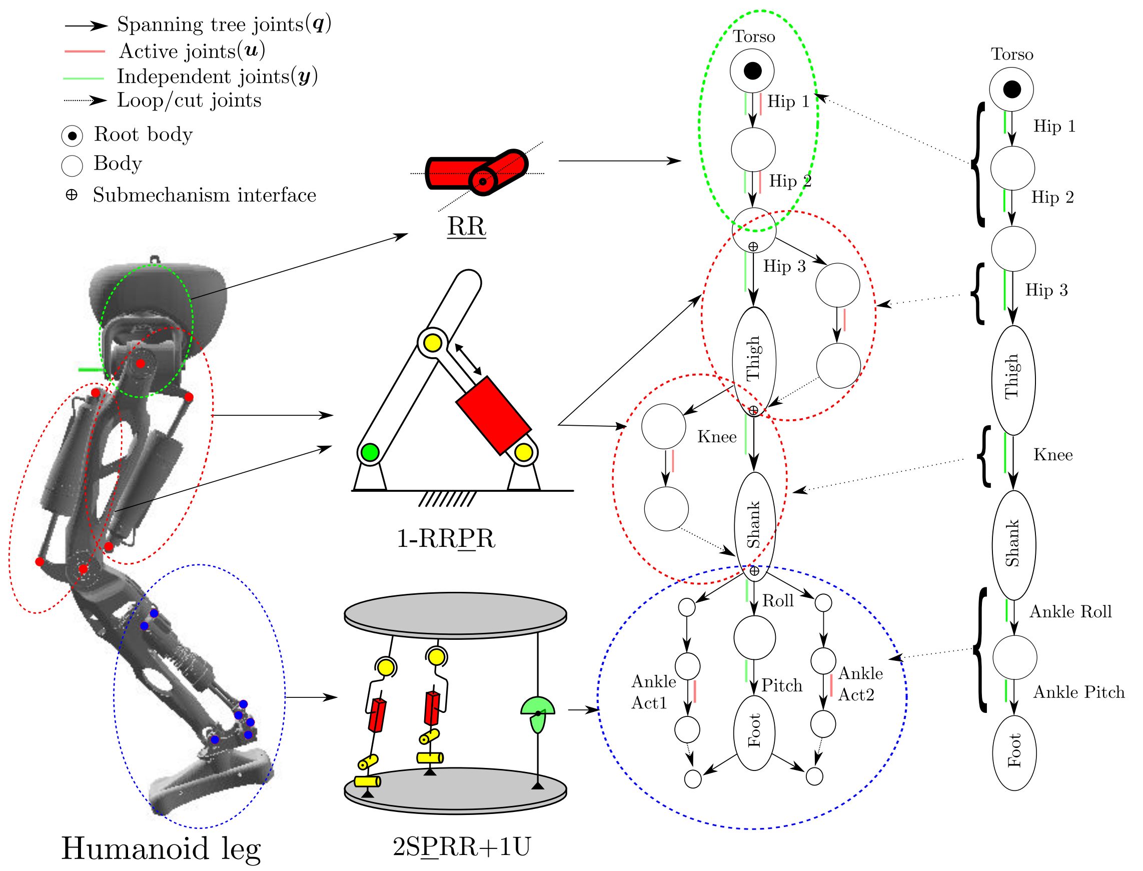


Figure 7.7: Example of series-parallel hybrid composition

ing the non-linear transmission. Fig. 7.6 shows the two design possibilities and the corresponding definitions of the submechanism modules.

7.3 Topological Modeling

A submechanism module is modeled using a two-terminal graph (TTG) which is defined as a graph with two distinguished vertices, called source and sink. The source and sink basically represent a fixed transformations to a submechanism interface point (for e.g. physical screws or nut-bolt pair) in the overall assembly of the hybrid robot. In addition to the source and sink links, base and end-effector links on the submechanism are identified on the TTG which is important for local resolution of loop closure constraints inside the submechanism module. Further, a spanning tree is deduced from this TTG, where all the nodes except for the ones connected using fixed joint are numbered using regular numbering scheme described in Section 6.1. It is additionally ensured that the main branch or trunk of the tree connecting the base and EE link is numbered first and then the additional branches are numbered. We call this extension of regular numbering scheme as *modular graph enumeration* scheme. Right top and bottom of Fig. 7.6 shows the associated submechanism graphs skipping the source and sink links.

The series composition $\mathcal{G} = \mathcal{G}^x \oplus \mathcal{G}^y$ of two TTGs $\mathcal{G}^x$ and $\mathcal{G}^y$ is a TTG created from the disjoint union of graphs $\mathcal{G}^x$ and $\mathcal{G}^y$ by merging the sink of $\mathcal{G}^x$ with the source of $\mathcal{G}^y$. The source of $\mathcal{G}^x$ becomes the source of $\mathcal{G}$ and the sink of $\mathcal{G}^y$ becomes the sink of $\mathcal{G}$ [Eppstein, 1992]. A series-parallel hybrid robot can be seen as series composition of various submechanism modules represented by their respective graphs. Let $\mathcal{T}^i$ denote the spanning tree of a submechanism module and s serial or parallel submechanism modules are joined in series to compose a series-parallel hybrid robot. The spanning tree of this hybrid robot ($\mathcal{T}$) will be given by:

$$\mathcal{T} = \bigoplus_{i=1}^{s} \mathcal{T}^i \tag{7.15}$$

The three joint variable sets namely spanning tree joints, independent joints and active joints set for the hybrid robot can be composed as: $\boldsymbol{q}^T = (\boldsymbol{q}_1^T, \dots, \boldsymbol{q}_s^T)$, $\boldsymbol{y}^T = (\boldsymbol{y}_1^T, \dots, \boldsymbol{y}_s^T)$, $\boldsymbol{u}^T = (\boldsymbol{u}_1^T, \dots, \boldsymbol{u}_s^T)$.

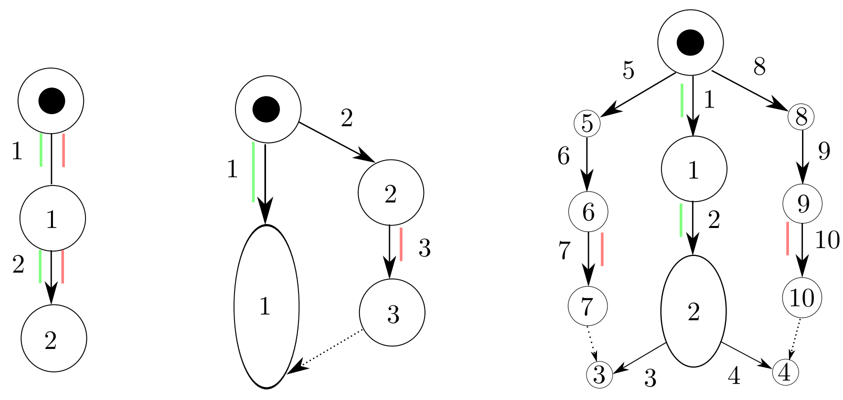
7.3.1 Bottom-Up Composition

The topological model of a complex series-parallel hybrid robot can be composed from topological models of its submechanism modules as shown in Equation 7.15. However, it must be noted that the composition operation is non-commutative i.e. $\mathcal{G}^1 \oplus \mathcal{G}^2 \neq \mathcal{G}^2 \oplus \mathcal{G}^1$. Once, the submechanism graphs have been combined, the fixed joints involved can be removed. Provided the graph descriptions of individual modules, the graph description of the composed mechanism can be easily deduced. Let (p_1, s_1) denote the predecessor array and successor arrays representing $\mathcal{G}^1$ and (p_2, s_2) denote the predecessor array and successor arrays representing $\mathcal{G}^2$. Let ι denote the submechanism interface link on $\mathcal{G}^1$, then the predecessor array of the composition p is given by:

$$p = \{p_1, p_2'\} \quad \text{where} \quad \forall i \in \{1, \dots, \mid p_2 \mid\}, p_2'(i) = \begin{cases} \iota : p_2(i) = 0 \\ p_2(i) + \max(s_1) : p_2(i) \neq 0 \end{cases} \tag{7.16}$$

and the successor array s of the composition is given by:

$$s = \{s_1, s_2'\} \quad \text{where} \quad \forall i \in \{1, \dots, \mid s_2 \mid\}, s_2'(i) = s_2(i) + \max(s_1)\ . \tag{7.17}$$



$\underline{\text{RR}}$

$p = \{0,1\}$
$s = \{1,2\}$
$\lambda = \{0,1\}$
$\mu = \{\{1\},\{2\}\}$

1-RR$\underline{\text{P}}$R

$p = \{0,0,2\}$
$s = \{1,2,3\}$
$\lambda = \{0,0,2\}$
$\mu = \{\{1,2\},\{\},\{3\},\{\}\}$

2S$\underline{\text{P}}$RR+1U

$p = \{0,1,2,2,0,5,6,0,8,9\}$
$s = \{1,2,3,4,5,6,7,8,9,10\}$
$\lambda = \{0,1,2,2,0,5,6,0,8,9\}$
$\mu = \{\{1,5,8\},\{2\},\{3,4\},$
$\{\},\{\},\{6\},\{7\},\{\},\{9\},\{10\},\{\}\}$

Figure 7.8: Graphs of submechanism modules

From predecessor and successor arrays, other properties such as parent array λ and child array μ can be easily calculated. This way of modeling is preferable when a complex model is being built up from scratch so that each submechanism model can be unit-tested beforehand or when there are well-defined attachment interfaces on all the modules involved during the composition.

7.3.2 Top-Down Decomposition

During topological modeling, there might sometimes be a need of decomposing a complex series-parallel hybrid model into models of its submechanism modules. This can also be done easily thanks to the modular graph enumeration scheme. If a topological graph $\mathcal{G}$ decomposes into $\mathcal{G}^1$ and $\mathcal{G}^2$, Let ι denote the submechanism interface link on $\mathcal{G}$, then the successor array splits into s_1 and s_2' such that $\min(\mu(\iota))$ is the first element of s_2'. Then, s_2 can be computed as $s_2(i) = s_2'(i) - \max(s_1) \quad \forall i \in \{1, \ldots, \mid s_2' \mid\}$. The predecessor array p will split into p_1 and p_2' such that ι is the first element of p_2'. And the predecessor array p_2 can be computed as:

$$\forall i \in \{1, \ldots, \mid p_2' \mid\}, p_2(i) = \begin{cases} 0 : p_2'(i) = \iota \\ p_2'(i) - \max(s_1) : p_2'(i) \neq \iota \end{cases} . \tag{7.18}$$

This kind of topological decomposition is needed when a complex system model is already at hand and the topological models of the submechanism modules are to be derived.

Example 3 *Fig. 7.7 shows the composition of a series-parallel hybrid humanoid leg with the help of four submechanism modules. The first module $\mathcal{T}^1$ is an RR module and consists of Hip1 and Hip2 joints, second module $\mathcal{T}^2$ is a 1-RRPR module used for Hip3 joint, third module $\mathcal{T}^3$ is again 1-RRPR module used for Knee joint and the fourth module $\mathcal{T}^4$ is 2-SPRR+1U mechanism used for the ankle joint. The respective graphs of the three distinct submechanism modules are provided in Fig. 7.8. $\mathcal{T}^2$ is attached to $\mathcal{T}^1$ at link 2, $\mathcal{T}^3$ is attached to the link 3 of $\mathcal{T}^1 \oplus \mathcal{T}^2$ and $\mathcal{T}^4$ is attached to the link 4 of $\mathcal{T}^1 \oplus \mathcal{T}^2 \oplus \mathcal{T}^3$. The submechanism interface links are collected in an submechanism interface array $\iota = \{0, 2, 3, 4\}$. The graph description of the overall series-parallel hybrid composition is given by:*

$$\begin{aligned}
p &= \{0,1,2,2,4,3,3,7,6,9,10,10,6,13,14,6,16,17\} \\
s &= \{1,2,3,4,5,6,7,8,9,10,11,12,13,14,15,16,17,18\} \\
\lambda &= \{0,1,2,2,4,3,3,7,6,9,10,10,6,13,14,6,16,17\} \\
\mu &= \{\{1\},\{2\},\{3,4\},\{6,7\},\{5\},\{\},\{9,13,16\},\{8\},\{\},\{10\}, \\
&\quad \{11,12\},\{\},\{\},\{14\},\{15\},\{\},\{17\},\{18\},\{\}\}
\end{aligned} \tag{7.19}$$

and the graph enumerated using the modular numbering scheme is shown in Fig. 7.9.

7.4 Kinematics and Dynamics

7.4.1 Submechanism Module

For serial submechanism modules, the independent variables in the spanning tree are the same as generalized coordinates of the spanning tree i.e. $\boldsymbol{q}_i = \boldsymbol{y}_i$, $\dot{\boldsymbol{q}}_i = \dot{\boldsymbol{y}}_i$ and $\ddot{\boldsymbol{q}}_i = \ddot{\boldsymbol{y}}_i$. The loop closure function for a serial submechanism module is given by:

$$\begin{aligned}
\boldsymbol{q}_i = \boldsymbol{\gamma}_i(\boldsymbol{y}_i) &= \boldsymbol{y}_i \\
\boldsymbol{G}_i &= \boldsymbol{I}_i \\
\boldsymbol{g}_i &= \boldsymbol{0}_i
\end{aligned} \tag{7.20}$$

where $\boldsymbol{I}_i$ is an identity matrix of size $(n_i \times n_i)$ and $\boldsymbol{g}_i$ is a zero vector of size $(n_i \times 1)$. Since, we assume that all the submechanism modules are properly actuated, the matrix $\boldsymbol{G}_{ui}$ is also an identity matrix of size $(n_i \times n_i)$.

Problems related to geometry or kinematics for parallel robots is usually easy to formulate but difficult to solve because they result in a set of non-linear algebraic equations which need careful analysis and treatment. The three most useful solution techniques to deal with such problems are polynomial continuation, Gröbner bases, and elimination method [Nielsen and Roth, 1999]. To the best knowledge of the author, there is no universally applicable way to solve kinematics problems in case of parallel robots. Hence, the geometer or kinematician may choose any formulation and solution method which works the best for a specific type of parallel robot and arrive at the loop closure function. The loop closure functions for parallel submechanism modules are defined using Equation 7.9. The foundations for deriving LCFs for 1-RRPR and 2-SPRR+1U mechanisms used in the humanoid leg example introduced in Fig. 7.7 can be found in Section 3.5 [Featherstone, 2008] and Section 4.3 [Kumar et al., 2018c] respectively and are skipped here for brevity.

The EOM for the equivalent spanning tree ($\mathcal{T}_i$) of a submechanism module subjected to analytical loop closure constraints ($\boldsymbol{G}_i^T \boldsymbol{\tau}_{ci} = 0$) is given by

$$\boldsymbol{G}_i^T \boldsymbol{M}_i \boldsymbol{G}_i \ddot{\boldsymbol{y}}_i + \boldsymbol{G}_i^T (\boldsymbol{C}_i + \boldsymbol{M}_i \boldsymbol{g}_i) = \boldsymbol{G}_i^T \boldsymbol{\tau}_i \tag{7.21}$$

It may be noted that Equation 7.21 has the same algebraic form as the equation of motion for the unconstrained system in Equation 7.1 and hence can be written in the form:

$$\hat{\boldsymbol{M}}_i \ddot{\boldsymbol{y}}_i + \hat{\boldsymbol{C}}_i = \boldsymbol{\tau}_{yi} \tag{7.22}$$

where $\hat{\boldsymbol{M}}_i = \boldsymbol{G}_i^T \boldsymbol{M}_i \boldsymbol{G}_i$ is module's $(m_i \times m_i)$ mass-inertia matrix which is symmetric and positive-definite and $\hat{\boldsymbol{C}}_i = \boldsymbol{G}_i^T (\boldsymbol{C}_i + \boldsymbol{M}_i \boldsymbol{g}_i)$ is its $(m_i \times 1)$ vector of bias forces and $\boldsymbol{\tau}_{yi} = \boldsymbol{G}_i^T \boldsymbol{\tau}_i$ is the $(m_i \times 1)$ vector of generalized forces for the submechanism module. The actuator forces $\boldsymbol{\tau}_{ui}$ for the submechanism module can be computed using Equation 7.14 where $\boldsymbol{G}_{ui} = \boldsymbol{Q}_i \boldsymbol{G}_i$.

With $\boldsymbol{q}_i, \dot{\boldsymbol{q}}_i, \ddot{\boldsymbol{q}}_i$ known by solving the loop constraints determined by Equation 7.9, only the equations (7.1) of the unconstrained system need to be evaluated. This is most efficiently carried out by a recursive $O(n)$ algorithm. Various of such have been proposed in the literature. The actual compuational effort depends on the representation of spatial twists and wrenches. The spatial representation [Jain, 2011b, Featherstone, 2008] is deemed as the most efficient representation. For the i^{th} submechanism module, denote with $\boldsymbol{V}_{i,k} = \left(\boldsymbol{\omega}^T, \boldsymbol{v}^T\right)^T \in se(3)$ the twist vector of body k, and with $\boldsymbol{J}_{i,k}$ the instantaneous screw coordinate vector of joint k, both in spatial representation. The recursive Newton-Euler algorithm (RNEA) [Featherstone, 2008, Müller, 2018] consists of a forward recursion ($k = 1, \ldots, n_i$), where the spatial state of the system is computed from the generalized coordinates, velocities, and accelerations ($\boldsymbol{q}_i, \dot{\boldsymbol{q}}_i, \ddot{\boldsymbol{q}}_i$).

$$\begin{aligned} \boldsymbol{V}_{i,k} &= \boldsymbol{V}_{i,\lambda_i(k)} + \boldsymbol{J}_{i,k} \dot{q}_{i,k} \\ \dot{\boldsymbol{V}}_{i,k} &= \dot{\boldsymbol{V}}_{i,\lambda_i(k)} + \boldsymbol{J}_{i,k} \ddot{q}_{i,k} + \mathbf{ad}_{\boldsymbol{V}_{i,\lambda_i(k)}} \boldsymbol{V}_{i,k} \end{aligned} \tag{7.23}$$

The backward recursion ($k = n_i, \dots, 1$) in RNEA computes the generalized forces $\boldsymbol{\tau}_i$ from the spatial state of the system. To this end, denote with $q_{i,k}$ and $\tau_{i,k}$ the kth element of $\boldsymbol{q}_i$ and $\boldsymbol{\tau}_i$ of submodule i.

$$\begin{aligned} \boldsymbol{W}_{i,k} &= \sum_{j \in \mu_i(k)} \boldsymbol{W}_{i,j} + \boldsymbol{M}_{i,k}\dot{\boldsymbol{V}}_{i,k} - \mathbf{ad}^T_{\boldsymbol{V}_{i,k}} \boldsymbol{M}_{i,k}\boldsymbol{V}_{i,k} + \boldsymbol{W}^{\text{app}}_{i,k} \\ \tau_{i,k} &= \boldsymbol{J}^T_{i,k}\boldsymbol{W}_{i,k} \end{aligned} \tag{7.24}$$

Here, $\boldsymbol{V}_{i,k}$, $\boldsymbol{M}_{i,k}$, $\boldsymbol{W}^{\text{app}}_{i,k}$ are the spatial twist, the inertia matrix, and the applied wrench of body k in submodule i, respectively. The 6×6 matrix

$$\mathbf{ad}_{\boldsymbol{V}} = \begin{pmatrix} \widetilde{\boldsymbol{\omega}} & \mathbf{0} \\ \widetilde{\boldsymbol{v}} & \widetilde{\boldsymbol{\omega}} \end{pmatrix} \tag{7.25}$$

is called the screw product or spatial cross product operator. Here, $\widetilde{\boldsymbol{\omega}}$ and $\widetilde{\boldsymbol{v}}$ represent the skew symmetric matrices associated with the vectors $\boldsymbol{\omega}$ and $\boldsymbol{v}$ respectively.

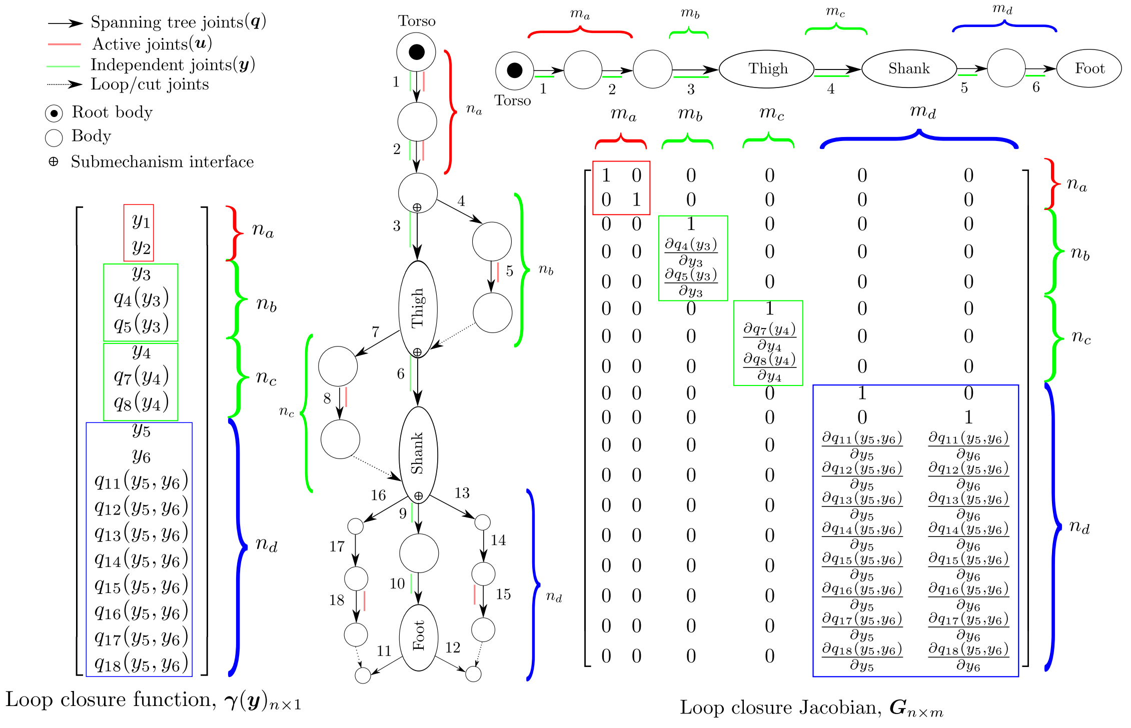

Spanning tree joints($\boldsymbol{q}$)
Active joints($\boldsymbol{u}$)
Independent joints($\boldsymbol{y}$)
Loop/cut joints
Root body
Body
Submechanism interface
Torso
Thigh
Shank
Foot
n_a
n_b
n_c
n_d
m_a
m_b
m_c
m_d
Loop closure function, $\boldsymbol{\gamma}(\boldsymbol{y})_{n\times 1}$
Loop closure Jacobian, $\boldsymbol{G}_{n\times m}$

Figure 7.9: Modular composition of loop closure function

Algorithm 10 Spanning tree state (SYSSTATE)

(in) Independent joint state $(\boldsymbol{y}, \dot{\boldsymbol{y}}, \ddot{\boldsymbol{y}})$

(out) Spanning tree joint state $(\boldsymbol{q}, \dot{\boldsymbol{q}}, \ddot{\boldsymbol{q}})$

```
1:  function SysState($\boldsymbol{y}, \dot{\boldsymbol{y}}, \ddot{\boldsymbol{y}}$)
2:      for $\mathcal{M}_i \in (\mathcal{M}_1, \dots, \mathcal{M}_s)$ do
3:          if $m_i \neq n_i$ then                          ▷ Check if module is parallel
4:              $\boldsymbol{q}_i \leftarrow \gamma_i(\boldsymbol{y}_i)$
5:              $\dot{\boldsymbol{q}}_i \leftarrow \boldsymbol{G}_i \dot{\boldsymbol{y}}_i$
6:              $\ddot{\boldsymbol{q}}_i \leftarrow \boldsymbol{G}_i \ddot{\boldsymbol{y}}_i + \boldsymbol{g}_i$
7:          else
8:              $\boldsymbol{q}_i \leftarrow \boldsymbol{y}_i$, $\dot{\boldsymbol{q}}_i \leftarrow \dot{\boldsymbol{y}}_i$, $\ddot{\boldsymbol{q}}_i \leftarrow \ddot{\boldsymbol{y}}_i$
9:          $\boldsymbol{q} \leftarrow \boldsymbol{y}_i$, $\dot{\boldsymbol{q}} \leftarrow \dot{\boldsymbol{y}}_i$, $\ddot{\boldsymbol{q}} \leftarrow \ddot{\boldsymbol{y}}_i$ $\forall i \in [1, \dots, s]$
10:     return $[\boldsymbol{q}, \dot{\boldsymbol{q}}, \ddot{\boldsymbol{q}}]$
```

7.4.2 Series-Parallel Hybrid Composition

The loop closure function for the series-parallel hybrid robot is composed as follows.

$$\begin{aligned}
\gamma &= \begin{bmatrix} \gamma_1^T & \dots & \gamma_i^T & \dots & \gamma_s^T \end{bmatrix}^T_{(n\times 1)} \\
\boldsymbol{G} &= \begin{bmatrix} \boldsymbol{G}_1 & \dots & \boldsymbol{0} & \dots & \boldsymbol{0} \\ \vdots & \vdots & \vdots & \vdots & \vdots \\ \boldsymbol{0} & \dots & \boldsymbol{G}_i & \dots & \boldsymbol{0} \\ \vdots & \vdots & \vdots & \vdots & \vdots \\ \boldsymbol{0} & \dots & \boldsymbol{0} & \dots & \boldsymbol{G}_s \end{bmatrix}_{(n\times m)} \\
\boldsymbol{g} &= \begin{bmatrix} \boldsymbol{g}_1^T & \dots & \boldsymbol{g}_i^T & \dots & \boldsymbol{g}_s^T \end{bmatrix}^T_{(n\times 1)}
\end{aligned} \tag{7.26}$$

From Equation 7.26, three observations about $\boldsymbol{G}$ can be made: it typically contains many zeros due to branch induced sparsity, it contains various identity matrix blocks corresponding to serial submechanism modules and it has a block diagonal nature due to modular choice of spanning tree. These properties of the matrix can be used to save some computational costs occuring in sparse matrix multiplications. Fig. 7.9 shows the block diagonal nature of the loop closure Jacobian for the series-parallel hybrid leg example introduced in Fig. 7.7. The actuator Jacobian matrix $\boldsymbol{G}_u$ also has a block diagonal nature which can be exploited while computing its inverse as shown in Equation 7.27.

$$
\begin{aligned}
\boldsymbol{G}_u = \boldsymbol{Q}\boldsymbol{G} &= \begin{bmatrix} \boldsymbol{G}_{u1} & \dots & \mathbf{0} & \dots & \mathbf{0} \\ \vdots & \ddots & \vdots & \ddots & \vdots \\ \mathbf{0} & \dots & \boldsymbol{G}_i & \dots & \mathbf{0} \\ \vdots & \ddots & \vdots & \ddots & \vdots \\ \mathbf{0} & \dots & \mathbf{0} & \dots & \boldsymbol{G}_{us} \end{bmatrix}_{(m \times m)} \\
\boldsymbol{G}_u^{-1} &= \begin{bmatrix} \boldsymbol{G}_{u1}^{-1} & \dots & \mathbf{0} & \dots & \mathbf{0} \\ \vdots & \ddots & \vdots & \ddots & \vdots \\ \mathbf{0} & \dots & \boldsymbol{G}_{ui}^{-1} & \dots & \mathbf{0} \\ \vdots & \ddots & \vdots & \ddots & \vdots \\ \mathbf{0} & \dots & \mathbf{0} & \dots & \boldsymbol{G}_{us}^{-1} \end{bmatrix}_{(m \times m)}
\end{aligned}
\tag{7.27}
$$

Algorithm 10 presents an algorithm to compute the position, velocity and acceleration state of the spanning tree from the loop closure function exploiting these properties. Once, the full system state of the equivalent spanning tree is known, it is straightforward to compute the pose of any point on the spanning tree as well as quantities like point Jacobian, spatial twists and acceleration etc.

Algorithm 11 Inverse Dynamics (IDYN)

(in) Independent joint state $(\boldsymbol{y}, \dot{\boldsymbol{y}}, \ddot{\boldsymbol{y}})$ and system model $(\mathcal{M})$

(out) Actuator forces $(\boldsymbol{\tau}_u)$

```
1:  function IDYN($\mathcal{M}, \boldsymbol{y}, \dot{\boldsymbol{y}}, \ddot{\boldsymbol{y}}$)
2:      $[\boldsymbol{q}, \dot{\boldsymbol{q}}, \ddot{q}] \leftarrow$ SYSSTATE($\boldsymbol{y}, \dot{\boldsymbol{y}}, \ddot{\boldsymbol{y}}$)        ▷ Algorithm 10
3:      $\boldsymbol{\tau} \leftarrow$ RNEA($\mathcal{M}, \boldsymbol{q}, \dot{\boldsymbol{q}}, \ddot{\boldsymbol{q}}$)        ▷ Inv. dyn. of tree, Algorithm 12
4:      for $\mathcal{M}_i \in (\mathcal{M}_1, \dots, \mathcal{M}_s)$ do
5:          if $m_i \neq n_i$ then        ▷ Check if module is parallel
6:              $\boldsymbol{\tau}_{ui} \leftarrow \boldsymbol{G}_{ui}^{-1} \boldsymbol{G}_i^T \boldsymbol{\tau}_i$
7:          else
8:              $\boldsymbol{\tau}_{ui} \leftarrow \boldsymbol{\tau}_i$
9:          $\boldsymbol{\tau}_u \leftarrow (\boldsymbol{\tau}_{ui} : 1 \leq i \leq s)$
10:     return $\boldsymbol{\tau}_u$
```

Algorithm 12 Modular Recursive Newton Euler Algorithm (RNEA)

(in) Spanning tree joint state $(\boldsymbol{q}, \dot{\boldsymbol{q}}, \ddot{\boldsymbol{q}})$, vector of external wrenches $\boldsymbol{W}^{\text{ext}}$ and system model $(\mathcal{M})$

(out) Tree joint forces $(\boldsymbol{\tau})$

```
1:  function RNEA($\mathcal{M}, \boldsymbol{q}, \dot{\boldsymbol{q}}, \ddot{\boldsymbol{q}}$)
2:      $\boldsymbol{V}_{0,0} = \boldsymbol{0}$                                    ▷ Velocity of root link
3:      $\dot{\boldsymbol{V}}_{0,0} = -\dot{\boldsymbol{V}}_g$                                 ▷ Gravity vector
4:      for $i \in (1, \dots, s)$ do                          ▷ Inter-modular F.R
5:          for $k \in (1, \dots, n_i)$ do                    ▷ Intra-modular F.R
6:              $\boldsymbol{V}_{i,k} = \boldsymbol{V}_{i,\lambda_i(k)} + \boldsymbol{J}_{i,k}\dot{q}_{i,k}$
7:              $\dot{\boldsymbol{V}}_{i,k} = \dot{\boldsymbol{V}}_{i,\lambda_i(k)} + \boldsymbol{J}_{i,k}\ddot{q}_{i,k} + \mathbf{ad}_{\boldsymbol{V}_{i,\lambda_i(k)}}\boldsymbol{V}_{i,k}$
8:      for $i \in (s, \dots, 1)$ do                          ▷ Inter-modular B.R
9:          for $k \in (n_i, \dots, 1)$ do                    ▷ Intra-modular B.R
10:             $\boldsymbol{W}_{i,k} = \sum_{j \in \mu_i(k)} \boldsymbol{W}_{i,j} + \boldsymbol{M}_{i,k}\dot{\boldsymbol{V}}_{i,k} - \mathbf{ad}^T_{\boldsymbol{V}_{i,k}}\boldsymbol{M}_{i,k}\boldsymbol{V}_{i,k} + \boldsymbol{W}^{\text{ext}}_{i,k}$
11:             $\tau_{i,k} = \boldsymbol{J}^T_{i,k}\boldsymbol{W}_{i,k}$
12:     return $\boldsymbol{\tau}$
```

An algorithm to solve the inverse dynamics of the series-parallel hybrid composition is provided in Algorithm 11 which takes as input the motion described in independent coordinates $(\boldsymbol{y}, \dot{\boldsymbol{y}}, \ddot{\boldsymbol{y}})$. First, the full state of the spanning tree is computed (Line 2) with the help of Algorithm 10. The next step is to compute the joint forces $(\boldsymbol{\tau})$ for the spanning tree which is done using a modular form of RNEA presented in Algorithm 12. In this algorithm, the forward and backward recursions are carried out first within the module (intra-modular) and then across the modules (inter-modular) to compute the spatial state of the spanning tree (using Equation 7.23) and tree joint forces (using Equation 7.24). And lastly, the tree joint forces are converted to the actuator forces of the series-parallel hybrid robot using an inter-modular recursion (see Line 6 and Line 8) by exploiting the block diagonal nature of loop closure Jacobian matrix.

7.4.3 Computational Effort

In general, it is more difficult to analyse the computational performance of dynamics algorithms for systems with closed loops because the closed form solutions to the loop closure constraints might not always exist and the number of floating point operations involved are highly dependent on the geometry of the parallel mechanism.

Nevertheless, the presented modular approach leads to some cost savings occuring in matrix-vector multiplication and inversion which will be discussed here. The tree joint forces ($\boldsymbol{\tau}$) are solved using $O(n)$ RNEA. The projection of these forces to the independent joint space of the robot i.e. $\boldsymbol{\tau}_y = \boldsymbol{G}^T\boldsymbol{\tau}$ can be computed with mn multiplications and $m(n-1)$ additions or $2m(n-1)$ floating point operations (FLOPs). Due to block diagonal structure in $\boldsymbol{G}$, this matrix-vector multiplication can be done in $2\sum_{i=1}^{s} m_i(n_i-1)$ FLOPs. It is trivial to show that $2\sum_{i=1}^{s} m_i(n_i-1) \le 2m(n-1)$ which demonstrates the cost savings involved in this step [2]. The projection of independent joint forces to actuator forces requires the inversion of actuator Jacobian matrix $\boldsymbol{G}_u$ which can be done with $O(m^3)$ complexity and its multiplication with $\boldsymbol{\tau}_y$ vector which requires $2m(m-1)$ FLOPs. Due to the modular formulation, actuator Jacobian matrix $\boldsymbol{G}_u$ also has a block diagonal structure and instead of a full inversion, its inverse can be computed simply by computing the inverse of its submatrix blocks along the diagonal. Hence, compared to $O((\sum_{i=1}^{s} m_i)^3)$ complexity involved in this process, the block diagonal nature leads to a reduced inversion complexity of $O(\sum_{i=1}^{s} m_i^3)$ as $\sum_{i=1}^{s} m_i^3 \le (\sum_{i=1}^{s} m_i)^3 \forall m_i \in \mathbb{N}$. Lastly, the matrix-vector multiplication $\boldsymbol{\tau}_u = \boldsymbol{G}_u^{-1}\boldsymbol{\tau}_y$ leads to a reduced cost of $2\sum_{i=1}^{s} m_i(m_i-1)$ FLOPs as compared to $2m(m-1)$ FLOPs. These cost savings demonstrate the efficiency of the inverse dynamics algorithm due to the adopted notion of modularity.

7.5 Conclusion

This chapter presents an analytical and modular approach for kinematic and dynamic modeling of series-parallel hybrid robots. The key idea behind this approach is to see a complex hybrid mechanism as a serial composition of serial or parallel submechanism module. Once, the submechanisms in the hybrid kinematic chain are identified, the topological model for each submechanim module is derived and used to compose the topological model of the complete hybrid chain. Thanks to the modular graph enumeration during topological modeling, the associated loop closure functions can be easily composed and have a block diagonal structure. This can be exploited in various kinematics and dynamics algorithms and leads to efficient and user-friendly models. The approach presented here forms the basis of the analytical and modular software workbench called HyRoDyn which will be presented in the next part of this thesis.

[2] $\sum_{i=1}^{s} m_i n_i \le \sum_{i=1}^{s} m_i \sum_{i=1}^{s} n_i \; \forall \; m_i,\; n_i \in \mathbb{N}$

Part IV

HyRoDyn: Design and Applications

Chapter 8

HyRoDyn Software Architecture

This chapter presents the various aspects of the HyRoDyn software architecture which is built on the modular and analytical methods for solving kinematics and dynamics of complex hybrid robotic systems presented in the previous chapter. The content presented here is based on [Kumar and Mueller, 2019] and was first presented in the form of a poster [Kumar et al., 2018d]. The chapter is organized as the following: Section 8.2 presents the robot description format for HyRoDyn and Section 8.3 describes a visual editing tool Phobos for generating such description files. Section 8.4 presents some software implementation details of HyRoDyn and Section 8.5 presents the integration of HyRoDyn in a robot middleware operating system called RoCK.

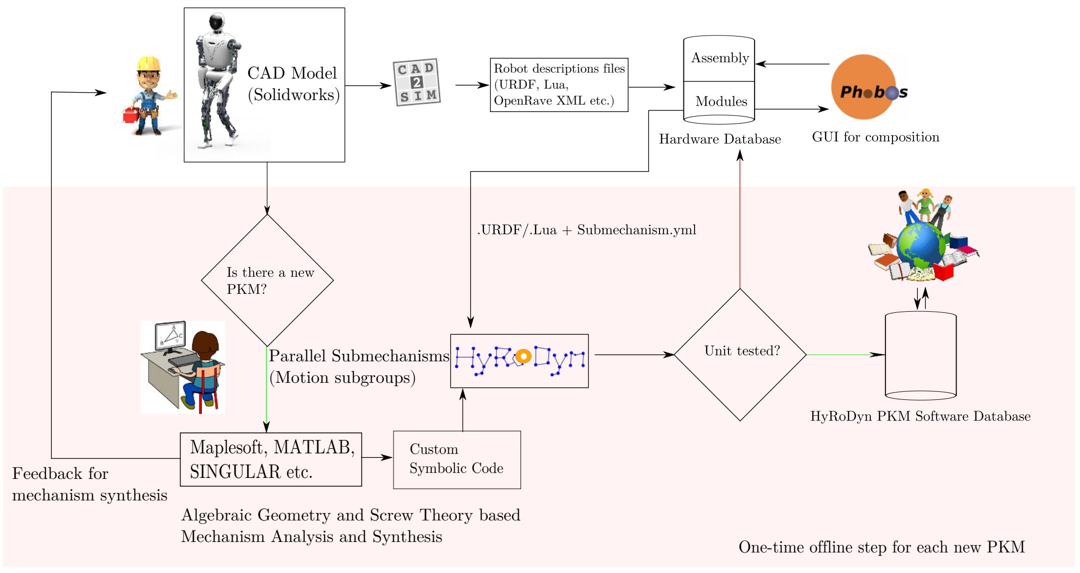

CAD Model (Solidworks)
CAD 2 SIM
Robot descriptions files (URDF, Lua, OpenRave XML etc.)
Assembly
Modules
Hardware Database
Phobos
GUI for composition
.URDF/.Lua + Submechanism.yml
Is there a new PKM?
Parallel Submechanisms (Motion subgroups)
HyRoDyn
Unit tested?
HyRoDyn PKM Software Database
Maplesoft, MATLAB, SINGULAR etc.
Custom Symbolic Code
Feedback for mechanism synthesis
Algebraic Geometry and Screw Theory based Mechanism Analysis and Synthesis
One-time offline step for each new PKM

Figure 8.1: HyRoDyn Developer Workflow

8.1 Motivation

Building a robot is a highly complex process which requires cross-domain expertise. Robots are becoming so complex that it is kind of impossible for a single person to manage the process of robot development. We restrict the scope of robot development to its conceptualization, construction, kinematic, dynamic modeling and position-velocity-torque control. These are the absolute minimum steps that are needed in the development process of most robots before more complex behaviors are realized with the system. To aid the further explanation, three roles are identified:

- **Designer**: Designer is the person responsible from initial design conceptualization to implementing the mechatronic aspects in the final prototype. A designer is an expert in CAD software and may also know some CAE tools (e.g. ADAMS MSC, RecurDyn) for immediate analysis and simulation of their design.

- **Kinematician**: Kinematician or geometer is a domain expert in the field of geometric and kinematic analysis of mechanisms. Such a person is expected to have strong foundations in mathematics in particular in the area of geometry and mechanics.

- **Control Engineer**: Control Engineer is the person responsible for implementing low, mid or high level controllers inside a robot. They are considered to be expert in control theory and have a good understanding of applied mechanics.

The main motivation behind the HyRoDyn software architecture is to provide a holistic treatment for kinematic and dynamic modeling of series-parallel hybrid robotic systems while involving the three above defined roles into the process. In Chapter 2, a systematic survey on the state of the art in series-parallel hybrid robots is presented. Designers find it increasingly useful to utilize parallel kinematics based submechanism modules in their design to achieve a good payload to weight ratio, stiffness properties and dynamic properties. While they can perform a basic simulation of their design in CAD or CAE software, they often turn to domain level experts to optimise their design further. The presence of closed loops in robots significantly increases the complexity of the kinematics and dynamics problems associated with multi-body systems (MBS) (see Section 2.3 for detailed perspectives). Hence, most multi-body dynamics libraries or software packages support only serial or tree type mechanisms and provide analytical formulations for solving forward kinematics, forward and inverse dynamics. Inverse kinematics is usually solved through numerical techniques. Further, it has been noted that a limited number of tools that do provide the possibility of modeling closed loop systems deal with loop closure constraints numerically. This allows for a general treatment of mechanisms but only leads to a limited *local* kinematics and dynamics analysis of these systems without deeper insights that are typically required in the design and analysis of complex mechanisms. Further, the numerical approaches may suffer from inaccuracies and computational inefficiency. Hence, a big portion of research done by the kinematics community is to provide comprehensive geometric/kinematic analysis of mechanisms of specific class or type (for e.g. see Chapter 4 and Chapter 5). This kind of analysis is usually a one time effort and very useful in the design optimisation for e.g. removing singularities from the feasible workspace, optimising force and velocity transmission of the parallel joint modules in different applications (e.g. wrist, ankle or torso design). Typically, this kind of feedback is given by the geometer/kinematician to the mechanical designer to optimise their design. This information is also useful for the control engineers for modeling the kinematics and dynamics to realize a position, velocity or torque control in the robot. However, there is no effort yet to make this kinematic analysis reusable in the context of design and modeling closed loop systems and the mechanisms that can be derived from their compositions. There is, however, a very practical need to do so because of the growing popularity of series-parallel hybrid designs in robotics. It is becoming increasingly desirable to analyse and control these robots accurately and efficiently. In Chapter 7, analytical and modular methods for solving the kinematics and dynamics of series-parallel hybrid robots are presented based on the concept of loop closure functions. The main idea behind HyRoDyn is to store these LCFs in configurable submechanism libraries to form a PKM software database. This leads to an analytical and modular software workbench which optimises the above workflow and make the process efficient and reusable. It fits into the overall picture of x-RoCK project series [D-RoCK, 2018] which attempts to streamline and simplify the robot development process. In the following, HyRoDyn developer and user workflows are discussed.

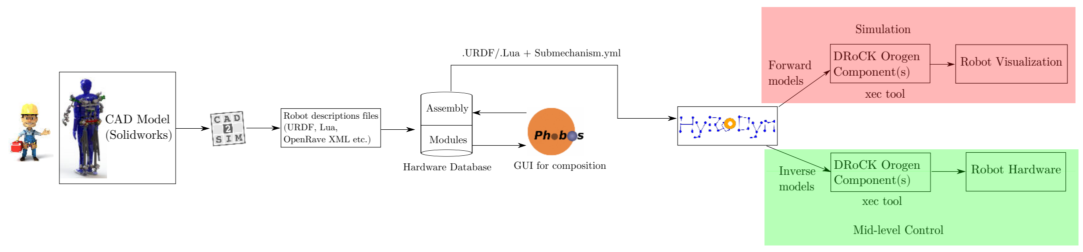


Figure 8.2: HyRoDyn User Workflow

8.1.1 Developer Workflow

An overview of HyRoDyn developer workflow is demonstrated in Fig. 8.1. The designer usually starts with a high level design specification which is often overly simplistic in nature. An example can be to design a 32 DOF humanoid robot with height 1.60-1.80 m and weight 60-70 kg which can carry 5 kg payload in each arm. The designer performs a simulation study to deduce further design criteria which includes required range of motion in each joint, torque requirements. They may start with a walking simulation of a simple stick figure humanoid (where links are simplified as point masses) to get an estimate of torque requirements in the system for the desired height and weight ranges. Once a rough estimate of range of motion, torque requirements is at hand, the designer starts with the mechanism as well as actuator design. The designer often takes inspiration from nature and tries to abstract the biological system with well studied kinematic joint types (see Fig. 7.5). For example, a humanoid leg might be constructed as a serial mechanism of type spherical-revolute-universal (SRU) or spherical-revolute-spherical (SRS). The kinematic joints in this design may further be realized using already existing closed loop mechanisms or PKM modules in order to optimise mass-inertia properties of links, dynamic performance, stiffness properties etc. In the process of doing so, they may invent a new parallel mechanism, the preliminary analysis of which can be conducted in typical CAD software. However, in order to further optimise the design, a complete kinematic analysis is desirable.

At this step, the designer communicates with the kinematician to perform a detailed analysis. Based on the topological description of this new mechanism, the kinematician formulates the constraint equations of the mechanism and analyses the geometric conditions under which these equations are valid [1]. The kinematician should try to keep these constraint equations as generic as possible by maintaining a balanced trade-off between the principle solvability of the equations and computational efficiency in solving them. They may use modern computer algebra tools (for e.g. MATLAB, Maple, Mathematica, SINGULAR etc) for analysing and solving these constraint equations. Various key insights like maximum number of solutions to forward/inverse kinematics, singularity curves of the mechanism, velocity/force transmission can be derived. This feedback can be provided to the designer for improving the mechanism design. The symbolic solution of these constraint equations can be exported in the form of efficient C-code and transferred into loop closure function of the submechanism library in HyRoDyn's PKM software database. More details about writing a submechanism library will be provided later in this chapter. The PKM software database in HyRoDyn will grow as developers around the world can contribute to it and benefit from the existing submechanism libraries.

The control engineer working with the designer and kinematician would export

[1]While the topological description carries indicative information about the mobility of the mechanism, the real mobility of the mechanism depends on the configuration and choice of geometric parameters (For further perspectives, read Chapter 3).

the CAD model of the submechanism into robot description formats (for e.g. URDF) using tools like CAD2SIM or Solidworks2URDF exporter. This hardware model can be unit-tested with the HyRoDyn library. At this stage, they can already realize a physical prototype of the mechanism and test the model for control purposes. Due to the modular approach here, it becomes easier to debug the software and improve the mechanism's model. On successful testing, the model can be stored in the x-RoCK hardware database.

8.1.2 User Workflow

Once, the submechanism design has been optimised, the designer repeats the same process for designing other submechanism modules and composes the overall model from it. For e.g. a humanoid leg of type SRU can be realized with a composition of RR module, 1-RRPR for hip flexion-extension, 1-RRPR for knee joint and 2-SPRR+1U module for the ankle joint (see Fig. 7.7). Preliminary simulation of the overall CAD model can be performed by the designer and the design can be optimised.

The kinematician at this stage checks if its feasible to a derive closed form solution for the abstracted serial SRU chain. If yes, such a model can be provided to the control engineer.

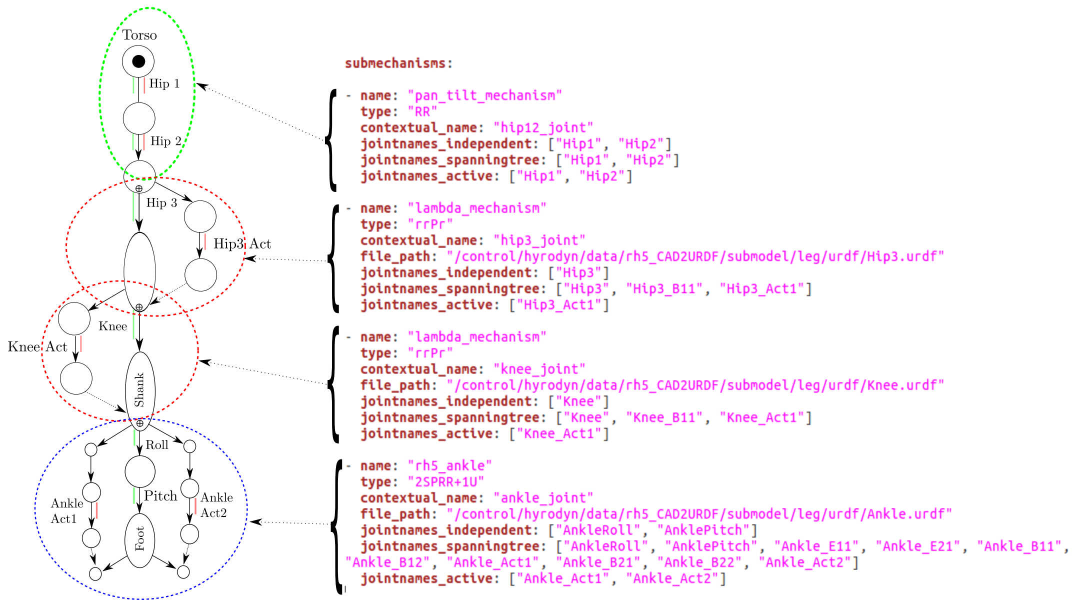
The control engineer exports the robot description of all the submechanism models (including source and sink transforms described in Section 7.3) and test it using HyRoDyn. Then, a Blender based visual editor called Phobos is used to compose the complete robot model and its description is exported. This provides a user-friendly way to compose highly complex models. Once, the overall model has been tested, it is added to the xRoCK hardware database. These models can be used to implement position, velocity or torque control in the robot and can be used in conjugation with more complex control architectures like whole body control. The forward algorithms in HyRoDyn can be used for simulation purposes and the inverse algorithms can be used for analysis and control. The overall user workflow is depicted in Fig. 8.2.

8.2 SMURF: Robot Description for HyRoDyn

There are a number of (open source) model formats commonly used in robotics research (we will not discuss proprietary formats used in industry here). Among them are the widespread Unified Robot Description Format (URDF[2]) used in ROS and the Simulation Description Format (SDF) developed for the simulation software Gazebo[3]. Other formats used in some applications are more general ones such as the 3D asset

[2]http://wiki.ros.org/urdf/XML

[3]http://sdformat.org/



```yaml
submechanisms:

- name: "pan_tilt_mechanism"
  type: "RR"
  contextual_name: "hip12_joint"
  jointnames_independent: ["Hip1", "Hip2"]
  jointnames_spanningtree: ["Hip1", "Hip2"]
  jointnames_active: ["Hip1", "Hip2"]

- name: "lambda_mechanism"
  type: "rrPr"
  contextual_name: "hip3_joint"
  file_path: "/control/hyrodyn/data/rh5_CAD2URDF/submodel/leg/urdf/Hip3.urdf"
  jointnames_independent: ["Hip3"]
  jointnames_spanningtree: ["Hip3", "Hip3_B11", "Hip3_Act1"]
  jointnames_active: ["Hip3_Act1"]

- name: "lambda_mechanism"
  type: "rrPr"
  contextual_name: "knee_joint"
  file_path: "/control/hyrodyn/data/rh5_CAD2URDF/submodel/leg/urdf/Knee.urdf"
  jointnames_independent: ["Knee"]
  jointnames_spanningtree: ["Knee", "Knee_B11", "Knee_Act1"]
  jointnames_active: ["Knee_Act1"]

- name: "rh5_ankle"
  type: "2SPRR+1U"
  contextual_name: "ankle_joint"
  file_path: "/control/hyrodyn/data/rh5_CAD2URDF/submodel/leg/urdf/Ankle.urdf"
  jointnames_independent: ["AnkleRoll", "AnklePitch"]
  jointnames_spanningtree: ["AnkleRoll", "AnklePitch", "Ankle_E11", "Ankle_E21", "Ankle_B11",
"Ankle_B12", "Ankle_Act1", "Ankle_B21", "Ankle_B22", "Ankle_Act2"]
  jointnames_active: ["Ankle_Act1", "Ankle_Act2"]
```

Figure 8.3: Example of YAML based submechanism description

exchange format Collada[4] and more specialized ones such as the custom XML format used by the motion planning library OpenRAVE[5]. All of these models were designed with specific purposes in mind and thus have some drawbacks or limitations. For instance, the rather generic, complicated syntax of Collada makes is less readable for humans, a problem since more often than not the task of deriving a robot description from CAD data still involves manual editing in text editors and is thus notoriously error-prone to begin with. On the other hand, URDF, which was specifically designed for robotics, misses such key features as proper definition of closed loops and modularity, problems that often lead to complicated work-arounds when working with URDF descriptions of contemporary, complex robots. Other formats, such as SDF, allow the definition of parallel linkages, but do not further provide the functionality to explicitly define a spanning tree of a looped graph, allowing a standardized tree representation of a model. This is however a desirable feature, for instance to maintain determinacy in planning and handling of joint state uncertainties.

To overcome the limitations of URDF, which mainly deals with geometric and physical parameters of a robot, but is arguably the most widely-used format in the scientific robotics community, the Supplementable Mostly Universal Robot description Format (SMURF) format has been developed at DFKI-RIC, augmenting URDF by annotating data with reference to its links and joints [von Szadkowski and Langosz, 2015]. To accomodate the definition of the kind of modular mechanisms described in this thesis, SMURF was updated to allow the speci-

[4]https://www.khronos.org/collada/
[5]http://openrave.programmingvision.com/wiki/index.php/Format:XML

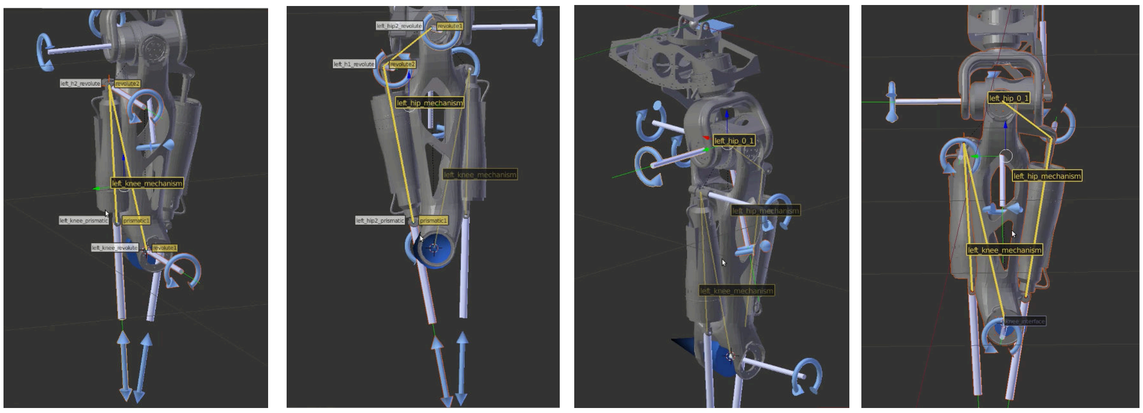
fication of sub-mechanisms mapped on the spanning tree defined in a URDF file, thus preserving the possibility to define an explicit spanning tree on a looped graph, while still adding the information relevant for identifying the nature of loop closure constraints in the mechanical system. The submechanisms definition is exported in the form of a YAML Ain't Markup Language (YAML) file as:

```
...
- name: <SUBMECHANISM_NAME>
  type: <TYPE>
  file_path: <PATH_TO_SUBMECHANISM_URDF>
  jointnames_independent: [<J1>,..,<JM>]
  jointnames_spanningtree: [<J1>,..,<JN>]
  jointnames_active: [<J1>,..,<JP>]
...
```

This YAML file basically contains the list of submechanisms that constitute the overall spanning tree of the robot (see Section 7.2.1 for the formal description). The parallel mechanisms are identified by a type (e.g. 1-RRPR, 2-SPRR+1U, 2SPU+1U, 6-UPS etc.) and vectors of names of independent joints, active joints and spanning tree joints are defined. The spanning tree joint names are listed respecting the modular graph enumeration scheme described in Section 7.3. Further, it is required to provide a file path to the submechanim's URDF here for the parallel submechanism modules. Overall, to define a series-parallel hybrid robot completely, one needs to have the full URDF file of the robot itself, along with a YAML based submechanism description file which contains information about the modular composition (see Fig. 8.3 for an example).

8.3 Phobos: Visual Editor for HyRoDyn

Phobos [von Szadkowski and Langosz, 2015] is an open source visual editor for robots based on Blender developed by DFKI-RIC and is capable of exporting both URDF and the accompanying SMURF models. It is based on what-you-see-is-what-you-get (WYSIWYG) philosophy. Phobos was extended for the purpose of the methods presented here, allowing the export of sub-mechanism definitions as described above as part of the SMURF representation of a robot. The sub-mechanisms recognized by HyRoDyn can be defined in a generic fashion in YAML files, which are parsed by Phobos and fed into an interactive operator that allows the assignment of the different joints defined for a mechanism to joints in the visual model, visualizing the components of the mechanism in the process. The relevant information for the mechanism definition is



(a) Knee module (b) Hip3 module (c) Hip12 module (d) All modules

Figure 8.4: Top-Down modeling using Phobos (Credits: Kai Alexander von Szadkowski)

stored in the objects representing the mechanical joints and exported to `SMURF` together with the rest of the robot data, i.e. the kinematic model as a `URDF`, motor and sensor information, etc. This data can then be processed by `HyRoDyn`.

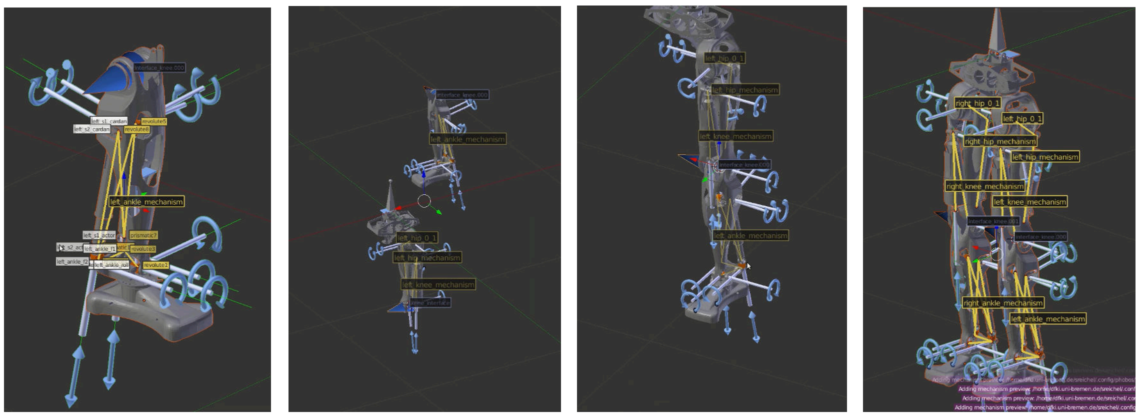
Due to the handling of objects in Blender in a similar spanning tree as `URDF`, the use of Phobos enables both a top-down or bottom-up approach to design and describe such modular series-parallel hybrid robotic systems. Since both approaches contain a number of hurdles for human designers, the use of a visual editor makes this process more reliable and simple.

8.3.1 Top-Down Modeling

In the top-down approach, an already existing `URDF` representation of a robot can be annotated with sub-mechanism definitions and the submechansim `URDF`s can be exported. Fig. 8.4 shows an example of top-down modeling using Phobos. A URDF of the RH5 humanoid upper leg is imported into Phobos and three submechanisms namely Knee, Hip3 and Hip12 are annotated on the model. A SMURF export at this stage provides the `YAML` based submechanism description and individual `URDF`s for each annotated submechanism. For the example shown in Fig. 8.4, the submechanism description will include the first three blocks of the description provided in Fig. 8.3. The blue cone at the end of the kinematic chain in Fig. 8.4 demonstrates the sink link.

8.3.2 Bottom-Up Modeling

In the bottom-up approach, `URDF` representations of pre-defined sub-mechanisms are assembled into one complex model. This is a powerful way of modeling complex systems as symmetry in the mechanical design can be exploited to replicate complex substructures without having to remodel them. This is demonstrated with the help of an example composition in Fig. 8.5. Fig. 8.5a shows the already annotated ankle submechanism of the humanoid leg. It is imported in the Phobos environment already containing the upper leg assembly (see Fig. 8.5b for an exploded view). Using the source and sink links on these submechanisms, they are connected to form the single leg assembly of RH5 humanoid as shown in Fig. 8.5c. Since, the two legs in the RH5 humanoid are symmetric in design, the leg assembly in replicated twice in Phobos. Then, the batch rename feature is used to rename left leg attributes to right leg attributes. After this step, the second leg joint in the right leg is adjusted to regain the symmetry which prepares lower body model of RH5 humanoid as shown in Fig. 8.5d.



(a) Ankle (b) Ankle & upperleg (c) RH5 single leg (d) RH5 both legs

Figure 8.5: Bottom-Up modeling using Phobos (Credits: Kai Alexander von Szadkowski)

8.4 HyRoDyn Software Library

Hybrid Robot Dynamics (HyRoDyn) is a software library that implements the modular and analytical formulations for series-parallel hybrid robotic systems presented in Chapter 7. It is implemented in C++ and utilizes the implementation of multi-body dynamics algorithms for tree type systems (based on Featherstone [Featherstone, 2008]) from the Rigid Body Dynamics Library (RBDL) [Felis, 2017]. Additionally, it depends upon the linear algebra package Eigen 3 [Guennebaud et al., 2010] and yaml-cpp[6] for parsing the modular submechanism definitions provided in SMURF model. The functions implemented in HyRoDyn have a

[6]https://github.com/jbeder/yaml-cpp

model-based interface which separates the models from the algorithms. The library is completely independent of any middleware software used in robotics community.

8.4.1 Submechanism Libraries

The most crucial aspect of HyRoDyn is its submechanism libraries which contains the symbolic expressions for the loop closure function of a particular type of the submechanism. Each submechanism library is a C++ class that inherits from an *abstract* class called AnalyticalLoopConstraintSet. It is a way of enforcing a contract between the class designer (i.e. HyRoDyn main developer) and the users of that class (i.e. Kinematician). The abstract class defines the matching member variables and functions that the derived submechanism class must have. Fig. 8.6 shows the the relationship between the abstract class AnalyticalLoopConstraintSet and child submechanism classes with the help of a class diagram. Hence, an object of the abstract class AnalyticalLoopConstraintSet is basically an abstract mechanism which defaults to a serial mechanism because the loop closure function for a serial chain defaults to identity function. A submechanism class is specific to a mechanism type which must have a member variables as geometric parameters of the mechanism, member functions as loop closure functions and a constructor which loads the robot model and extracts the geometric parameters of the robot. The class structure is completely agnostic to any specific method for solving the loop closure constraints inside the class. While the general guideline is to derive analytical expressions for LCFs, one may also use numerical methods inside them.

Example 4 *The C++ code snippet presented in Listing 8.1 shows the ease of use of HyRoDyn library. First, an object rh5 of the RobotModel_HyRoDyn class is created (Line 7). By defining the file paths to the URDF and submechanism description of the RH5 humanoid leg (Lines 9 and 10), a robot model is loaded (Line 12). The Hip3 and Knee joint angles are set to* 0.1 *radians and* 0.2 *radians respectively (Lines 14 and 15) which constitutes a configuration in independent joint space* y *of the robot. One could set the independent joint velocities and accelerations by setting the variables yd and ydd (they default to zero). Three different member functions of this class are demonstrated in this example.*

- *The complete system state of the robot can be computed by calling the member function calculate_system_state(). The output of this function is stored in the member variables Q, QDot, QDDot (see Line 18-19 for the position output of the spanning tree).*

- *The inverse dynamics of the robot can be computed by calling the member function calculate_inverse_dynamics(). The output of this function is stored in the*

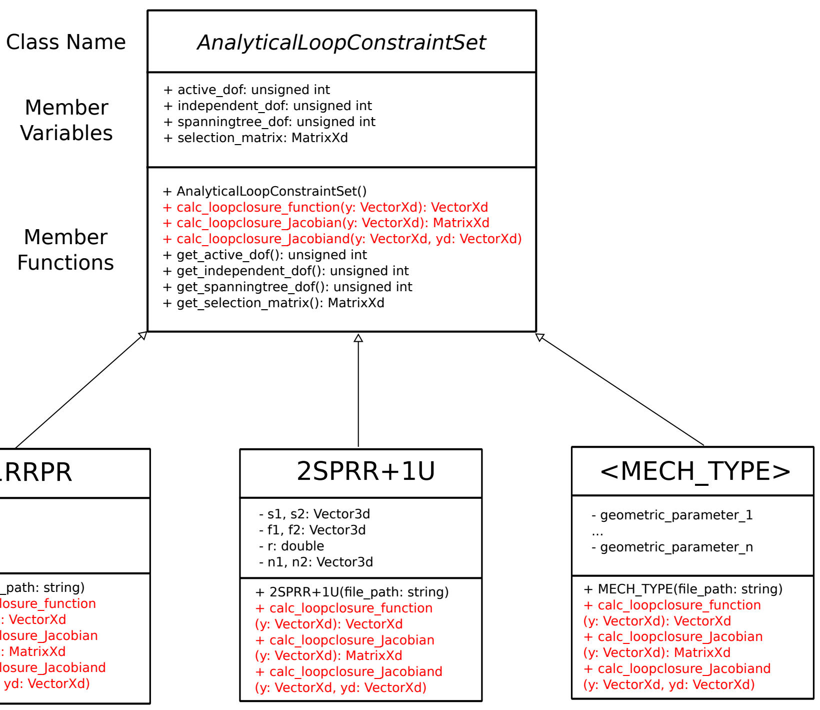

Class Name
Member Variables
Member Functions
AnalyticalLoopConstraintSet
+ active_dof: unsigned int
+ independent_dof: unsigned int
+ spanningtree_dof: unsigned int
+ selection_matrix: MatrixXd
+ AnalyticalLoopConstraintSet()
+ calc_loopclosure_function(y: VectorXd): VectorXd
+ calc_loopclosure_Jacobian(y: VectorXd): MatrixXd
+ calc_loopclosure_Jacobiand(y: VectorXd, yd: VectorXd)
+ get_active_dof(): unsigned int
+ get_independent_dof(): unsigned int
+ get_spanningtree_dof(): unsigned int
+ get_selection_matrix(): MatrixXd
1RRPR
- l1: double
- l2: double
- d: double
+ 1RRPR(file_path: string)
+ calc_loopclosure_function (y: VectorXd): VectorXd
+ calc_loopclosure_Jacobian (y: VectorXd): MatrixXd
+ calc_loopclosure_Jacobiand (y: VectorXd, yd: VectorXd)
2SPRR+1U
- s1, s2: Vector3d
- f1, f2: Vector3d
- r: double
- n1, n2: Vector3d
+ 2SPRR+1U(file_path: string)
+ calc_loopclosure_function (y: VectorXd): VectorXd
+ calc_loopclosure_Jacobian (y: VectorXd): MatrixXd
+ calc_loopclosure_Jacobiand (y: VectorXd, yd: VectorXd)
<MECH_TYPE>
- geometric_parameter_1
...
- geometric_parameter_n
+ MECH_TYPE(file_path: string)
+ calc_loopclosure_function (y: VectorXd): VectorXd
+ calc_loopclosure_Jacobian (y: VectorXd): MatrixXd
+ calc_loopclosure_Jacobiand (y: VectorXd, yd: VectorXd)

Figure 8.6: Class diagram demonstrating the relationship between the abstract class AnalyticalLoopConstraintSet and submechanism classes

member variable Tau_actuated (see Line 21-22 for the output actuator torques).

- *The forward dynamics of the robot can be computed by calling the member function calculate_forward_dynamics(). The output of this function is stored in the member variable Tau_actuated (see Line 25-26 for the output independent joint accelerations).*

The output of this simple program can be found in Listing 8.2. Notice in Line 2 that the position state of the spanning tree is zero at all indices except for indices belonging to the Hip3 and Knee submechanism modules. The velocity and acceleration state of the spanning tree are zero vectors because there was no input velocities and accelerations to the model and hence not shown here. The inverse dynamics output is shown in Line 4 which corresponds to the gravity torques in this configuration since the velocity and acceleration are set to zero. The forward dynamics output shown in Line 6 also corresponds to an almost zero vector since the output of inverse dynamics is the input to forward dynamics function.

Listing 8.1: C++ code for solving system state, inverse and forward dynamics using HyRoDyn

```
1 #include <iostream>
2 #include <hyrodyn/robot_model_hyrodyn.hpp>
```

```cpp

int main(int argc, char** argv)
{
    // Create an object of RobotModel_HyRoDyn class
    hyrodyn::RobotModel_HyRoDyn rh5;
    // Define robot description file paths
    string filepath_urdf = "leg.urdf";
    string filepath_submechanisms = "submechanisms_leg.yml";
    // Load robot model
    rh5.load_robotmodel(filepath_urdf, filepath_submechanisms);
    // Set independent joint angles
    rh5.y(2) = 0.1; // Hip3 joint
    rh5.y(3) = 0.2; // Knee joint
    // Compute the system state of the robot
    rh5.calculate_system_state();
    cout << "System State Output: " << rh5.Q.transpose() << endl;
    // Compute inverse dynamics of the robot
    rh5.calculate_inverse_dynamics();
    cout << "Inverse Dynamics Output: " << endl <<
    rh5.Tau_actuated.transpose() << endl;
    // Compute forward dynamics of the robot
    rh5.calculate_forward_dynamics();
    cout<<"Forward Dynamics Output: " << endl <<
    rh5.ydd.transpose() << endl;
    return 0;
}
```

Listing 8.2: Output of C++ code in Listing 8.1

```
System State Output:
0    0    0.1    0.120279    −0.00973191    0.2    −0.0393719    0.0129114    0
0    0    0    0    0    0    0    0    0
Inverse Dynamics Output:
−1.2033 −0.0178554 −49.4034 36.2226 −1.48666 −1.48665
Forward Dynamics output:
−9.3213e−17 2.2155e−16 6.3693e−16 −7.4643e−16 −5.0238e−16 1.7903e−14
```

8.4.2 Computational Performance

The computational performance of HyRoDyn is evaluated by testing it with different robot models (e.g. UR5, Recupera Wheelchair system, RH5 humanoid) of varying complexity for different algorithms (SYSSTATE, IDYN, FDYN) presented in this thesis. Active, independent and spanning tree DOF for these systems as well as number

Table 8.1: Robotic systems tested with HyRoDyn

Robotic system	Spanning tree DOF (n)	Independent DOF (m)	Active DOF (p)	Independent loops (c)
UR5	6	6	6	0
Recupera	18	5	5	2
RH5 leg	18	6	6	4
RH5 both legs	36	12	12	8
RH5 full	71	29	29	14

of independent closed loops present in them are specified in Table 8.1. The computational performance is measured in terms of CPU time for $100,000$ calls of these methods for randomized input in independent joint space respecting the joint limits. Fig. 8.7 shows the average CPU time needed for using these methods for the different robotic systems. These tests are performed on a standard laptop with Intel Core i7 CPU @ 2.8 GHz.

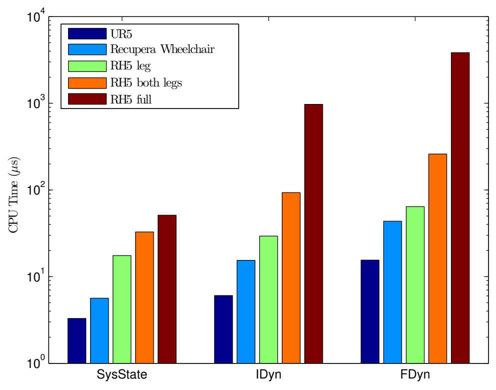


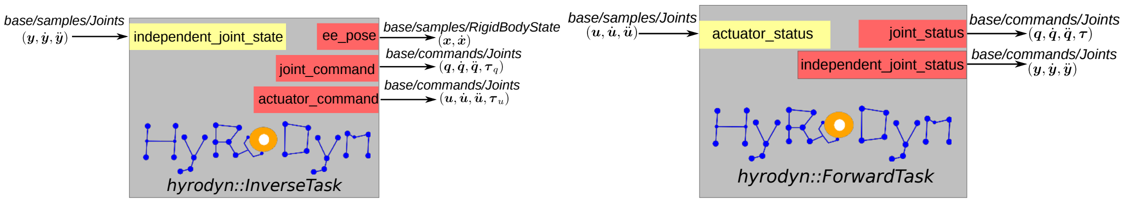
Figure 8.7: Computational performance of HyRoDyn for solving the system state, inverse dynamics and forward dynamics for different robotic systems described in Table 8.1

8.5 Integration in Middleware

A middleware is a software framework which enables distributed processes to exchange data on heterogeneous platforms. This software bus uses an object map to offer a simple and coherent interface to access objects and to guarantee data transmission. Probably, the most frequently classical used robot middlewares are the Robot

Operating System (ROS), Yet Another Robot Platform (YARP), Robot Construction Kit (RoCK), OpenRTM etc.

HyRoDyn is implemented as an Orogen component in the Robot Construction Kit (RoCK)[Joyeux, 2010] which is based on the component model of the Orocos Real Time Toolkit (RTT) and the object request broker (ORB) implementation, omniORB. Components encapsulate different functionalities or tasks, run independently and provide input and output for other components. The configuration can be applied to each component individually. `HyRoDyn-orogen` component implements both inverse and forward tasks. The inverse task as shown in Fig. 8.8a takes as input the motion trajectories defined in the independent joint space $(\boldsymbol{y}, \dot{\boldsymbol{y}}, \ddot{\boldsymbol{y}})$ and computes the actuator trajectories $(\boldsymbol{u}, \dot{\boldsymbol{u}}, \ddot{\boldsymbol{u}}, \boldsymbol{\tau}_u)$. This is crucial for abstraction of the robot in independent joint space so that higher level of control can be done by treating the robot as a tree type system. The forward task as shown in Fig. 8.8b takes as input the actuator status $(\boldsymbol{u}, \dot{\boldsymbol{u}}, \ddot{\boldsymbol{u}})$ and computes the independent joint status $(\boldsymbol{y}, \dot{\boldsymbol{y}}, \ddot{\boldsymbol{y}})$ of the robot. Both tasks compute the full system state of the spanning tree $(\boldsymbol{q}, \dot{\boldsymbol{q}}, \ddot{\boldsymbol{q}})$ and make it available on an output port which can be used for robot visualization. The input-output interface of `HyRoDyn-orogen` component is shown in Fig. 8.8. The component can be configured by a `SMURF` file (`URDF` and `YAML` based submechanism file) and provides a bi-directional mapping between the independent joint space and actuator space of the robot. These tasks can be used for both real time simulation and control of series-parallel hybrid robots. The component based architecture of RoCK framework makes it easy to reuse the same component in different robotics applications.


base/samples/Joints
$(\boldsymbol{y}, \dot{\boldsymbol{y}}, \ddot{\boldsymbol{y}})$
independent_joint_state
ee_pose
base/samples/RigidBodyState
$(\boldsymbol{x}, \dot{\boldsymbol{x}})$
joint_command
base/commands/Joints
$(\boldsymbol{q}, \dot{\boldsymbol{q}}, \ddot{\boldsymbol{q}}, \boldsymbol{\tau}_q)$
actuator_command
base/commands/Joints
$(\boldsymbol{u}, \dot{\boldsymbol{u}}, \ddot{\boldsymbol{u}}, \boldsymbol{\tau}_u)$
HyRoDyn
hyrodyn::InverseTask
base/samples/Joints
$(\boldsymbol{u}, \dot{\boldsymbol{u}}, \ddot{\boldsymbol{u}})$
actuator_status
joint_status
base/commands/Joints
$(\boldsymbol{q}, \dot{\boldsymbol{q}}, \ddot{\boldsymbol{q}}, \boldsymbol{\tau})$
independent_joint_status
base/commands/Joints
$(\boldsymbol{y}, \dot{\boldsymbol{y}}, \ddot{\boldsymbol{y}})$
HyRoDyn
hyrodyn::ForwardTask

(a) Inverse Task (b) Forward Task

Figure 8.8: HyRoDyn orogen component in RoCK

8.5.1 Interfacing with Other Components

An advantage of component based software architecture is that it allows interfacing with other components to exchange data so that complex applications can be designed in a modular way. HyRoDyn orogen component can be interfaced with other components by following the guidelines below.

- To keep the configuration requirement of the component minimal, the output of the component is decided based on its input.

  - If only position is provided, HyRoDyn only computes the position and static forces/torques (gravity terms).
  - If position and velocity are provided as inputs, HyRoDyn computes output position+velocity with required forces/torques (coriolis-centrifugal + gravity terms).
  - If position, velocity and acceleration are provided as inputs, HyRoDyn computes output position, velocity, acceleration with required forces/torques (mass-inertial + coriolis-centrifugal + gravity terms).
  - If effort is provided on the input port, it will be assumed that inverse dynamics on the abstracted mechanism is being solved outside the component and hence, the component will simply transform these torques into the actuation space. This feature can be used if model order reduction is desired for inverse dynamics computations.

- HyRoDyn component takes a very strict approach towards plant modeling in mechanics domain. The input and output ports should strictly correspond to the joints available in the spanning tree. It does not bypass any joint status/-commands if they are not available in the urdf file and submechanism file.

- Floating base robots such as humanoids are modeled using a six DOF free flyer joint which should be defined explicitly in both YAML based submechanism description file and the robot's URDF. A combination of six one DOF joints are added to the urdf in the following sequence: X, Y, Z, RX, RY, RZ. This becomes the part of both independent joint space input and joint space command output. The actuator space remains the same. HyRoDyn expects the floating base coordinates (from other components or Inertial Measurement Unit) as the input as it does not solve the inverse dynamics problem in a true hybrid sense.

- The input and output ports operate on SI units i.e. for rotary joints: position is measured in rad, velocity in rad/s, torque in N m etc. and for linear joints: position is measured in m, velocity in m/s, force in N etc. Torque (N m) to current (A) conversion or position (rad/m) to motor ticks conversion must be done outside this component.

8.5.2 Examples

HyRoDyn component can be used in various application designs. In the following two examples, we discuss its application in gravity component control and task space control of a robot.

Example 5 *HyRoDyn orogen component can be used to put a robot in gravity compensation mode. Fig.* 8.9 *shows the application of HyRoDyn orogen component in the gravity compensation control of the Recupera Exoskeleton system using the RoCK framework. HyRoDyn component takes as input the independent joint state from NDLCOM joint driver, solves the inverse dynamics and generates the actuator space commands which is fed to the NDLCOM driver. These are then sent to the low level actuation space controllers configured in current control mode implemented on FPGA stacks in each joint.*

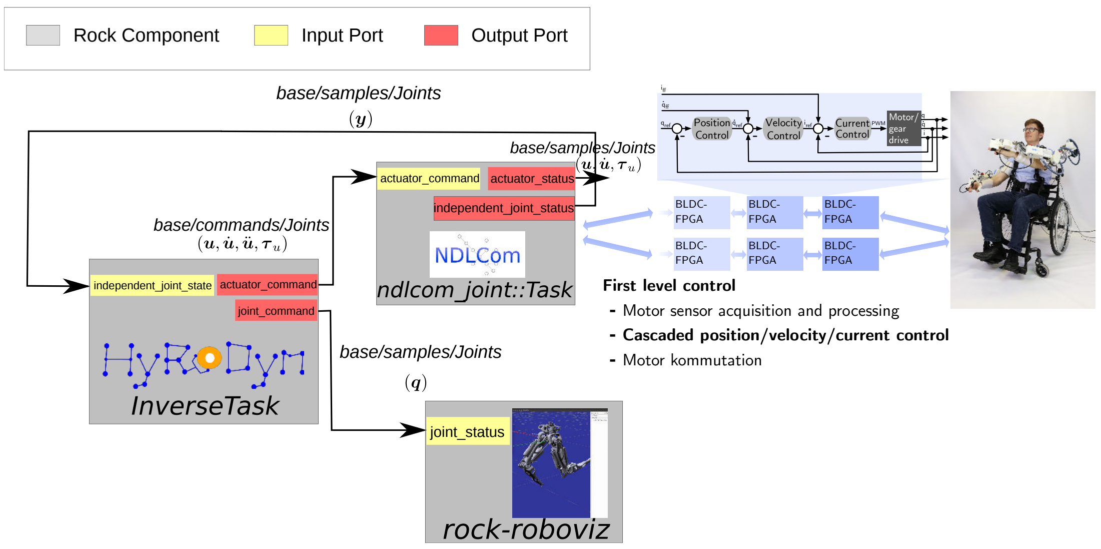


Figure 8.9: Application of HyRoDyn orogen component in the gravity compensation control of the Recupera Exoskeleton

Example 6 *HyRoDyn orogen component can be used to in whole body control applications where the chosen solver only has to work for a tree type system. Fig.* 8.10 *shows its application in whole body control (WBC) of the RH5 humanoid using the RoCK framework. WBC component takes as input a list of task space trajectories and outputs the independent joint space velocities for the robot by exploiting the redundancy of the Jacobian. It is then fed to an interpolator componnet which generates the position, velocity and acceleration command for the robot. HyRoDyn component solves the joint-space inverse dynamics and generates the actuator space commands which is then fed to the NDLCOM device drivers and further sent to the low level actuation space controllers implemented on FPGA stacks in each joint.*

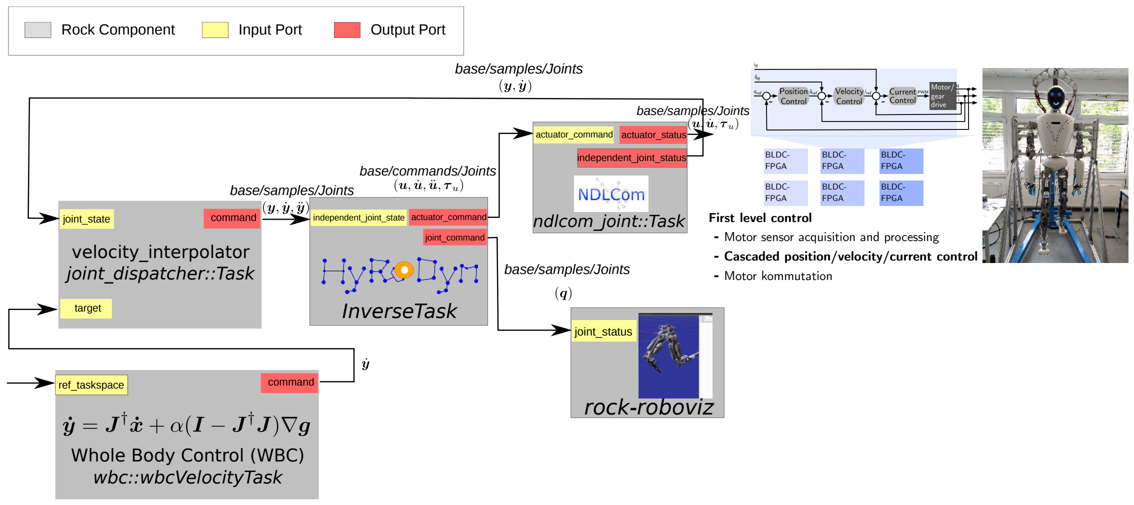


Figure 8.10: Application of HyRoDyn orogen component in the whole body control of RH5 humanoid

8.6 Conclusion

This chapter presents the architecture of HyRoDyn software framework for the simulation and control of the series-parallel hybrid robots. Different aspects of this framework such as robot description format, visual editing of the robot models, HyRoDyn software library and its integration in RoCK middleware are discussed. In the next chapter, various results from the applications of this software framework in the simulation, control and analysis of the two hybrid systems namely Recupera-Reha exoskeleton and RH5 humanoid will be presented.

Chapter 9

Results and Applications

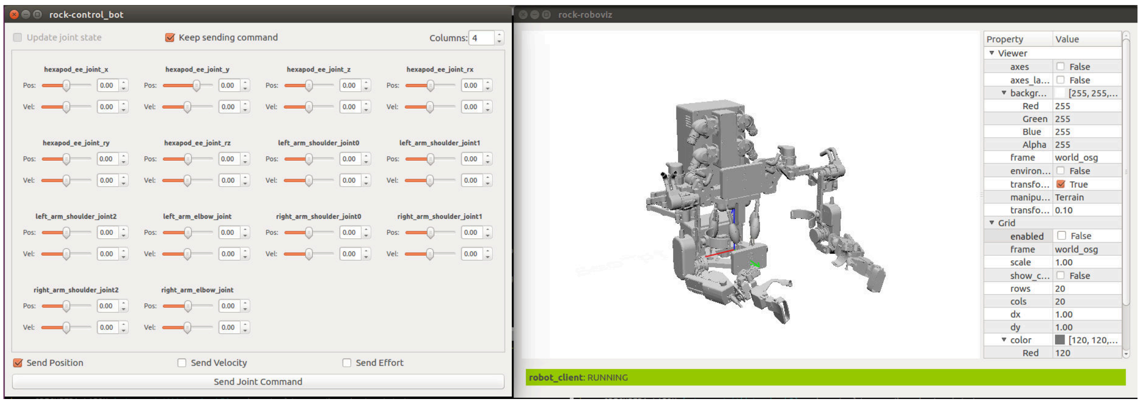
This chapter presents the results from simulation, analysis and real time control of the series-parallel hybrid robots, in particular, Recupera-Reha exoskeleton and RH5 humanoid, using HyRoDyn. As an outlook, the application of this tool in model order reduction of the dynamic model is presented. Finally, the application of this tool in the area of robotic rehabilitation is presented. The content presented here is based on [Kumar and Mueller, 2019, Kumar et al., 2019a] and [Kumar et al., 2019b]. The chapter is organized as the following: Section 9.1 presents the simulation results on the two systems. Section 9.2 presents the results from real time control of these systems exploiting the inverse dynamic model. Section 9.3 presents the applications of this tool in robot analysis and as an outlook Section 9.4 presents some insights into model order reduction. Finally, Section 9.5 presents the application of this tool in robotic rehabilitation.

9.1 Simulation

HyRoDyn can be used for both kinematic and dynamic simulation of series-parallel hybrid robots. In the following, two different simulation examples of the Recupera-Reha exoskeleton and RH5 humanoid are presented.

9.1.1 Recupera-Reha Exoskeleton

Let us consider a subsystem from Recupera-Reha exoskeleton which includes the upper body (which includes two arms) and torso submechanism (6-UPS Stewart platform) for the purpose of a purely kinematic simulation. This system has a total of 4 (left arm) + 4 (right arm) + 6 (torso) = 14 DOF, 1 (double parallelogram in left shoulder) + 1 (double parallelogram in right shoulder) + 5 (6-UPS Stewart platform) = 7 independent closed loops, and 8 (left arm) + 8 (right arm) + 24 (6-UPS Stewart platform) = 40 DOF in the spanning tree. Fig. 9.1 shows the kinematics simulation environment in RoCK. On the left side of the figure, a GUI to send joint commands in the independent joint space is shown and on the right side the robot can be visualized in its zero configuration.



Update joint state [✓ Keep sending command] Columns: 4

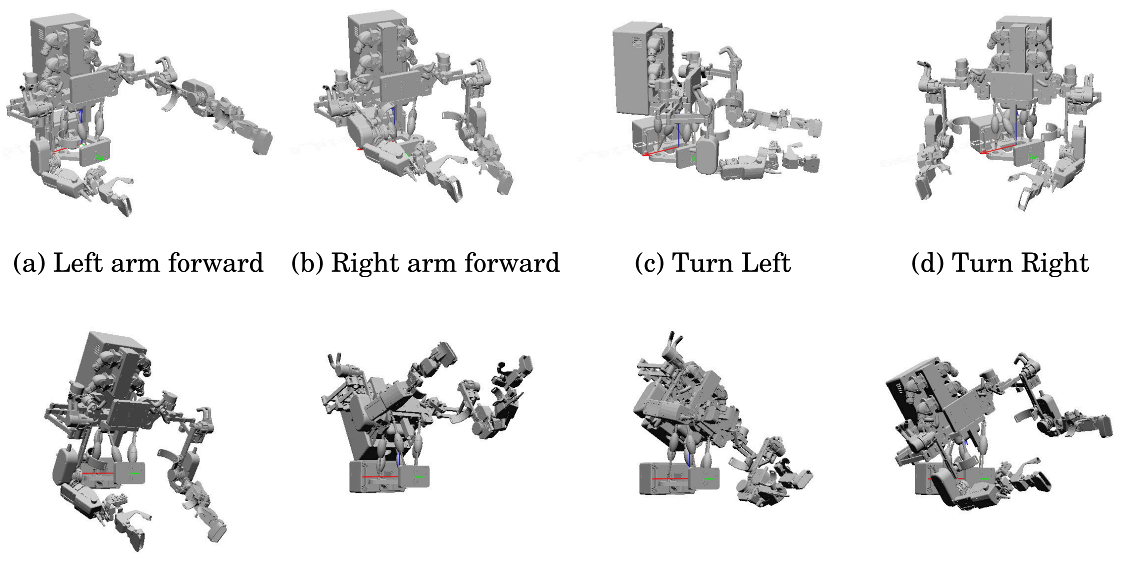
hexapod_ee_joint_x	hexapod_ee_joint_y	hexapod_ee_joint_z	hexapod_ee_joint_rx
Pos: 0.00	Pos: 0.00	Pos: 0.00	Pos: 0.00
Vel: 0.00	Vel: 0.00	Vel: 0.00	Vel: 0.00
hexapod_ee_joint_ry	**hexapod_ee_joint_rz**	**left_arm_shoulder_joint0**	**left_arm_shoulder_joint1**
Pos: 0.00	Pos: 0.00	Pos: 0.00	Pos: 0.00
Vel: 0.00	Vel: 0.00	Vel: 0.00	Vel: 0.00
left_arm_shoulder_joint2	**left_arm_elbow_joint**	**right_arm_shoulder_joint0**	**right_arm_shoulder_joint1**
Pos: 0.00	Pos: 0.00	Pos: 0.00	Pos: 0.00
Vel: 0.00	Vel: 0.00	Vel: 0.00	Vel: 0.00
right_arm_shoulder_joint2	**right_arm_elbow_joint**		
Pos: 0.00	Pos: 0.00		
Vel: 0.00	Vel: 0.00		

[✓ Send Position] Send Velocity Send Effort

Send Joint Command

Property	Value
▾ Viewer	
axes	☐ False
axes_la...	☐ False
▾ backgr...	[255, 255,...
Red	255
Green	255
Blue	255
Alpha	255
frame	world_osg
environ...	☐ False
transfo...	✓ True
manipu...	Terrain
transfo...	0.10
▾ Grid	
enabled	☐ False
frame	world_osg
scale	1.00
show_c...	☐ False
rows	20
cols	20
dx	1.00
dy	1.00
▾ color	▨ [120, 120,...
Red	120

Figure 9.1: Kinematics simulation environment for Recupera Exoskeleton in RoCK



(a) Left arm forward (b) Right arm forward (c) Turn Left (d) Turn Right

(e) Bend forward (f) Bend backward (g) Bend Left (h) Bend Right

Figure 9.2: Different poses in kinematic simulation of Recupera exoskeleton

Fig. 9.2 shows the different poses during the kinematic simulation using HyRoDyn. Fig. 9.2a and Fig. 9.2b show the left and right arms pointing forward respectively which is achieved by performing shoulder flexion and elbow extension movements. Fig. 9.2c and Fig. 9.2d show the turn left and right movements respectively which is achieved by setting the yaw angle (about z-axis) of the Stewart mechanism in independent joint space. Fig. 9.2e and Fig. 9.2f show the bend forward and back-

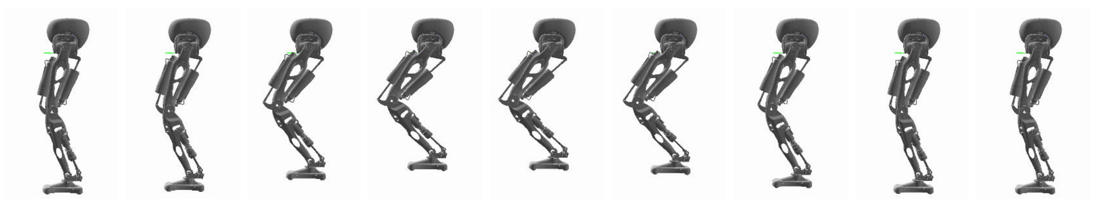
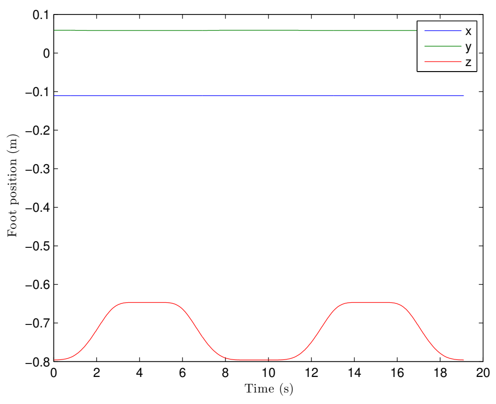
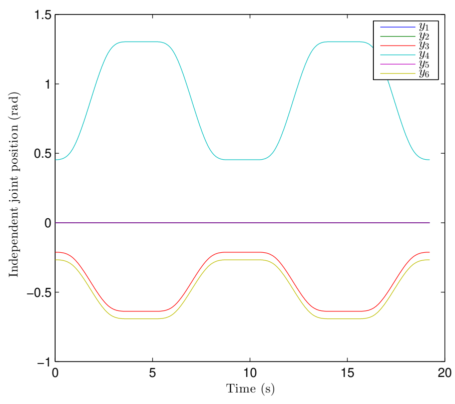


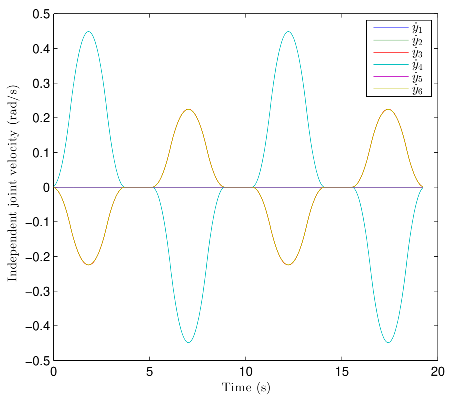
Figure 9.3: Animation of up-down movement with RH5 leg



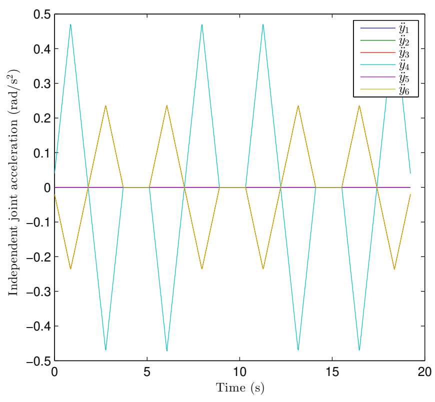
(a) Foot position



(b) Independent joint position



(c) Independent joint velocity



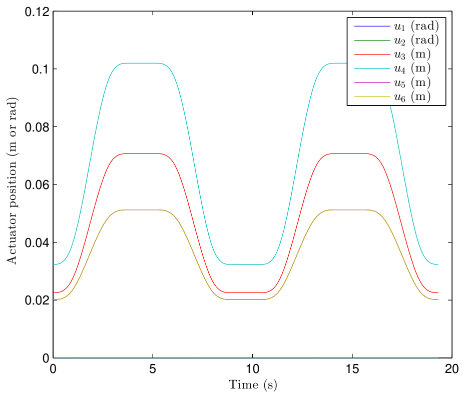
(d) Independent joint acceleration

Figure 9.4: Task space and independent joint space trajectories of RH5 leg for up-down movement

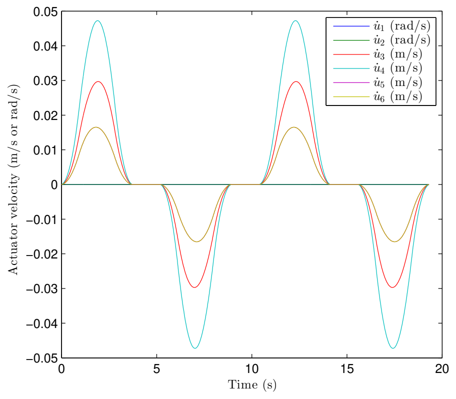
ward movements respectively which is achieved by setting the roll angle in the torso mechanism. Similarly, Fig. 9.2g and Fig. 9.2h show the bend left and backward right movements respectively which is achieved by setting the pitch angle in the torso mechanism. During the kinematic simulation, the full position state of the spanning tree is computed analytically and hence an accurate visualization of the complex robotic system can be achieved.

9.1.2 RH5 Humanoid Leg

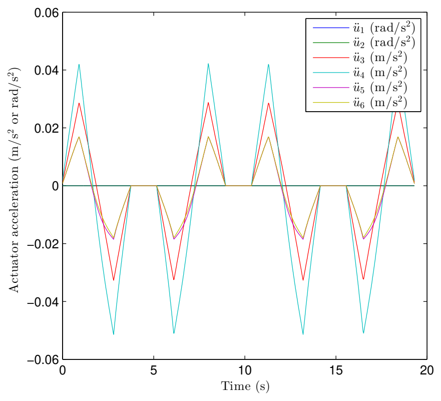
In Fig. 8.10, the application of HyRoDyn in whole body control of a humanoid robot is shown. Here, we present the task space control of RH5 humanoid leg previously introduced in Fig. 7.7. We assume a perfect model and hence, no controller action is required. Two set points i.e. foot up position and foot down position, are chosen in the task space of the robot. Using whole body control, the independent joint positions needed to reach these points are computed for the abstracted serial robot. Then, these waypoints in independent joint space are fed to an interpolator which provides smooth trajectories $(\boldsymbol{y}, \dot{\boldsymbol{y}}, \ddot{\boldsymbol{y}})$ for up and down movement of the leg. Fig. 9.4 shows the task space and independent joint space trajectories for the RH5 leg to produce this movement. These trajectories are then used to compute the actutaor trajectories $(\boldsymbol{u}, \dot{\boldsymbol{u}}, \ddot{\boldsymbol{u}}, \boldsymbol{\tau}_u)$ using inverse kinematics and inverse dynamics algorithms as presented earlier. Fig. 9.5 shows the actuator position, velocity, acceleration and torque profile of the robot. Further, the full spanning tree state $(\boldsymbol{q}, \dot{\boldsymbol{q}}, \ddot{\boldsymbol{q}})$ is computed during this process which can be used for robot visualization (see Fig. 9.3).



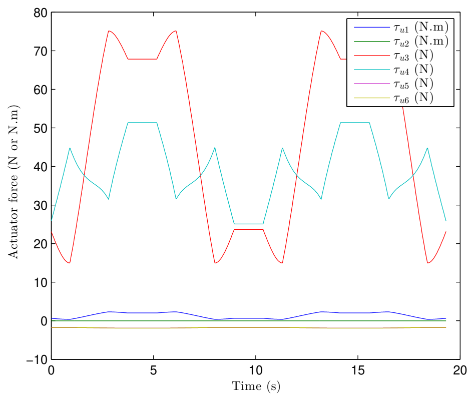
(a) Actuator position



(b) Actuator velocity

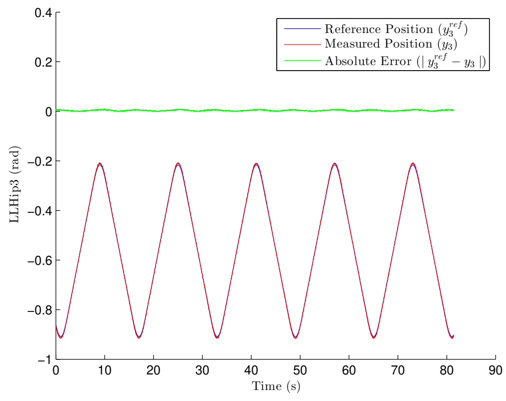


(c) Actuator acceleration

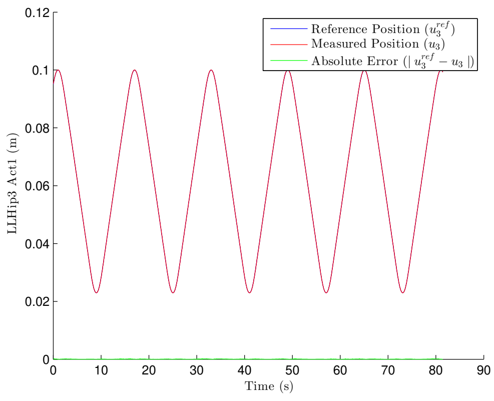


(d) Actuator forces

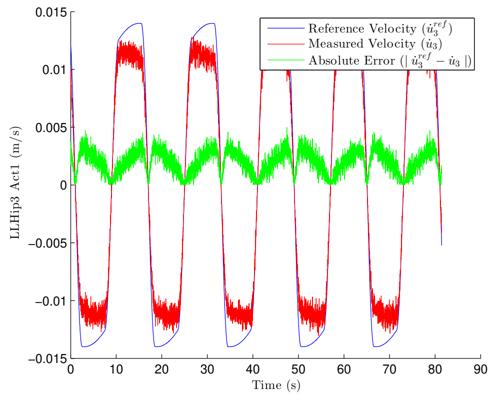
Figure 9.5: Actuator state of RH5 leg for up-down movement



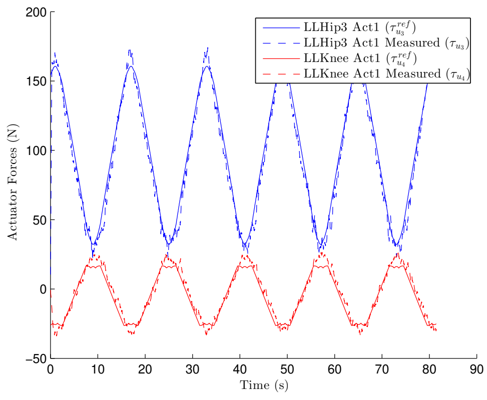
(a) Input motion tracking



(b) Actuator position tracking



(c) Actuator velocity tracking



(d) Actuator force tracking

Figure 9.6: Feed-forward control of RH5 humanoid using HyRoDyn

9.2 Real Time Control

HyRoDyn can be used for both kinematic and dynamic control of series-parallel hybrid robots. Two different examples of real-time control are presented here which takes into account the inverse dynamic model of the concerned robotic systems.

9.2.1 Feed-Forward Control of RH5 Humanoid

Feed-forward control scheme consists of feed-forward of the nonlinear dynamic model of robot and a linear servo feedback. It is the simplest form of the non-linear controller which can be employed for motion control of a robot exploiting its dynamic model. The control law can be formulated by the following equation:

$$\boldsymbol{\tau} = \boldsymbol{\tau}_u + \boldsymbol{K}_p(\boldsymbol{u}^{ref} - \boldsymbol{u}) + \boldsymbol{K}_d(\dot{\boldsymbol{u}}^{ref} - \dot{\boldsymbol{u}}) \tag{9.1}$$

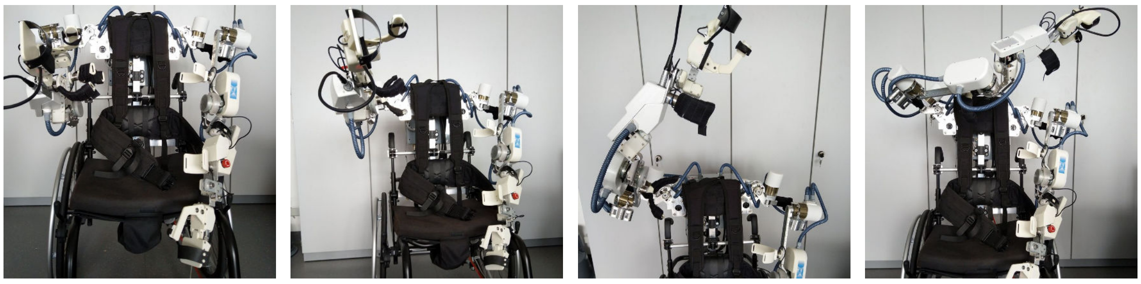

(a) Pose 1 (b) Pose 2 (c) Pose 3 (d) Pose 4

Figure 9.7: Gravity compensation for four different poses

Table 9.1: Commanded torque, measured torque and MAE for 4 different poses shown in Fig. 9.7

Pose	Commanded Torque (Nm)	Measured Torque (Nm)	$\|MAE\|$
1	$(-3.72, 0.85, -5.02, -2.99)^T$	$(-3.78, 0.86, -4.84, -2.99)^T$	0.2631
2	$(-4.70, 1.42, -9.62, -3.24)^T$	$(-4.59, 1.42, -9.51, -3.24)^T$	0.1536
3	$(-1.59, -1.91, -5.37, -0.41)^T$	$(-1.56, -1.90, -5.48, -0.42)^T$	0.1162
4	$(5.40, -3.86, -8.24, -2.09)^T$	$(5.27, -3.85, -8.31, -2.09)^T$	0.1435

where $\boldsymbol{\tau}_u$ is the vector of actuator forces computed by inverse dynamic model i.e. Equation 7.14, $\boldsymbol{K}_p = \text{diag}(K_p^1, K_p^2, \ldots, K_p^p)$ and $\boldsymbol{K}_d = \text{diag}(K_d^1, K_d^2, \ldots, K_d^p)$ are the diagonal matrices for proportional and derivative gains respectively. Fig. 8.10 shows the application of HyRoDyn in feed-forward motion control of the RH5 humanoid leg presented in Fig. 7.7. The input motion is specified in the independent joint space using an position interpolator between two desired set points. For the sake of simplicity, only one joint i.e. LLHip3 is moved from a position of -0.91 rad to -0.22 rad. The interpolator produces a smooth reference trajectory ($\boldsymbol{y}_3^{ref}$, $\dot{\boldsymbol{y}}_3^{ref}$, $\ddot{\boldsymbol{y}}_3^{ref}$) between these two set points back and forth. Inverse kinematics and dynamics problems are solved using HyRoDyn and the reference commands for the actuators ($\boldsymbol{u}^{ref}$, $\dot{\boldsymbol{u}}^{ref}$, $\boldsymbol{\tau}_u^{ref}$) are generated. These actuator commands are then fed to the feed-forward controller implemented on the FPGA of the BLDC joints in the humanoid platform. This controller utilizes the commanded actuator forces as a feed-forward term and commanded actuator position and velocity (along with the measured position and velocity i.e. $\boldsymbol{u}$ & $\dot{\boldsymbol{u}}$) for computing the linear servo feedback. Fig. 9.6 shows the trajectory tracking in both actuation and independent joint space. In particular, Fig. 9.6a shows the motion tracking in the independent joint space i.e. it compares the reference position values ($\boldsymbol{y}_3^{ref}$) against absolute position encoder measurements in the concerned joint ($\boldsymbol{y}_3$). Similarly, Fig. 9.6b and Fig. 9.6c show the actuator position and velocity tracking. Finally, Fig. 9.6d compares the commanded actuator force in LLHip3 and LLKnee joints against the direct force measurements available from the actuators.

9.2.2 Gravity Compensation Control of Recupera Exoskeleton

A good gravity compensation model is crucial to ensure transparency of the exoskeleton to the human subject. Fig. 8.9 shows the application of HyRoDyn in the gravity compensation control of the Recupera Exoskeleton system. The dynamic model in HyRoDyn takes as input the independent joint state (y) of the robotic system and outputs the actuator torques (τ_u) to compensate the gravity effect. The actutaor forces are then converted into reference current values for the FPGA based current controller implemented on the BLDC joints in the system. Table 9.1 shows average commanded torque (predicted with inverse dynamic model), average measured torque (through motor current), mean absolute error (MAE) in joint space and norm of MAE for a duration of $10\mathrm{s}$ sampled at $100\mathrm{Hz}$ for four balanced poses (guided by hand) of the right arm as shown in Fig. 9.7. It can be observed that for all poses, norm of MAE is between 0.12 Nm & 0.26 Nm which demonstrates the good quality of the model. Fig. 9.7 shows four different poses during the dual arm gravity compensation mode with the human subject inside the exoskeleton. Due to the good quality of the gravity compensation model, the human subject can move its arms freely within the system. Optionally, human arm weight compensation can also be included in the model.

9.3 Robot Analysis

HyRoDyn provides various tools for robot analysis as demonstrated in the following.

9.3.1 Workspace, Configuration Space, Actuation Space

HyRoDyn can be used for various offline applications like computation of the robot's workspace, configuration space, actuation space etc. Fig. 9.8 shows the workspace of Recupera Exoskeleton arm computed by homogenously discretizing the independent joint space and visualized with Blender based software tool Phobos. For example, Fig. 4.6 shows the slice of configuration space showing the actuation space of the 2-SPRR+1U parallel mechanism in the ankle joint of the RH5 leg. Such data can also be used to generate Look Up Tables (LUTs) or data driven approaches for learning the kinematic model.

9.3.2 Quality of Velocity and Force Transmission

The quality of velocity or force transmission of a robot can be measured by plotting the inverse of condition number of the kinematic Jacobian matrix ($\boldsymbol{J}$) over the robot's workspace. The inverse of condition number of the Jacobian is calculated with

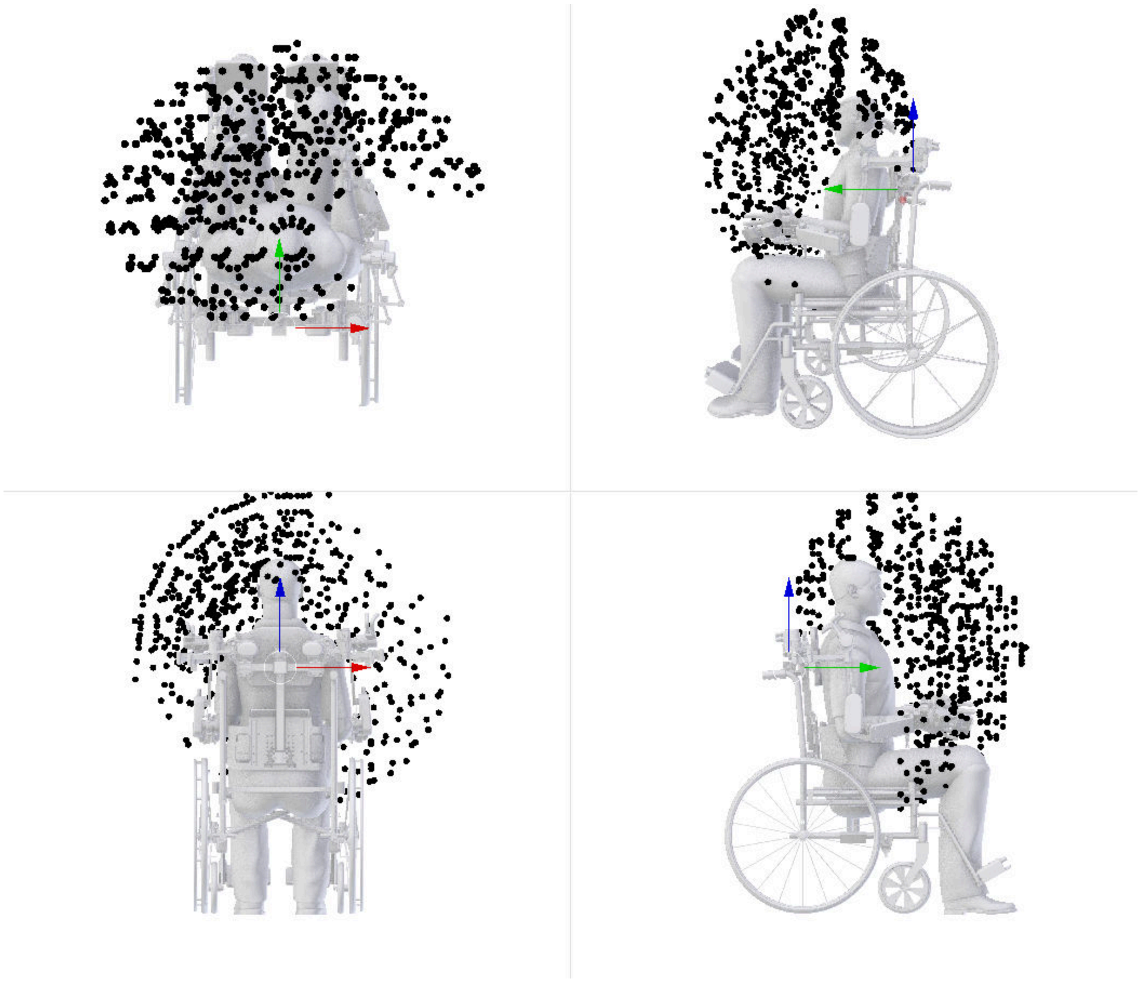
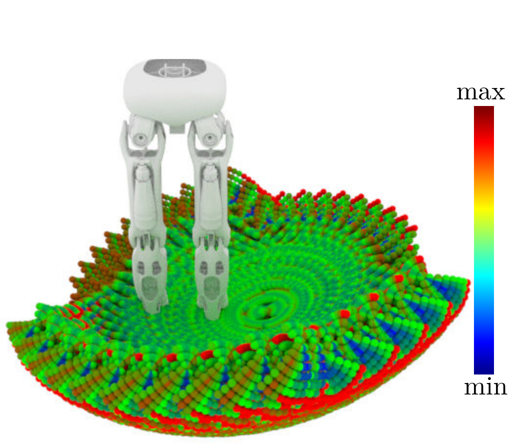


Figure 9.8: Workspace of Recupera Exoskeleton arm



(a) RH5 leg

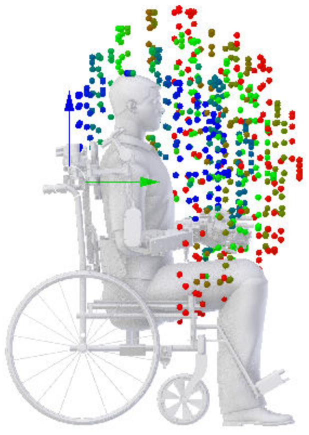

(b) Recupera Right Arm

Figure 9.9: Workspace quality measure using HyRoDyn

$c(\boldsymbol{J}) = \frac{1}{\|\boldsymbol{J}\|\|\boldsymbol{J}^{-1}\|}$ where $\|.\|$ represents the Euclidean norm of the matrix. Its value is bounded between 0 and 1 which signify the worst and best conditioning. Fig. 9.9 shows the plot of inverse of condition number of Jacobian over the workspace of RH5 leg and Recupera right arm systems. It is also possible to compute various slices of configuration space and equip them with such quality measures (for e.g. see Fig. 4.8). This is very relevant for the analysis of various parallel submechanism modules in a robot.

9.4 Model Order Reduction

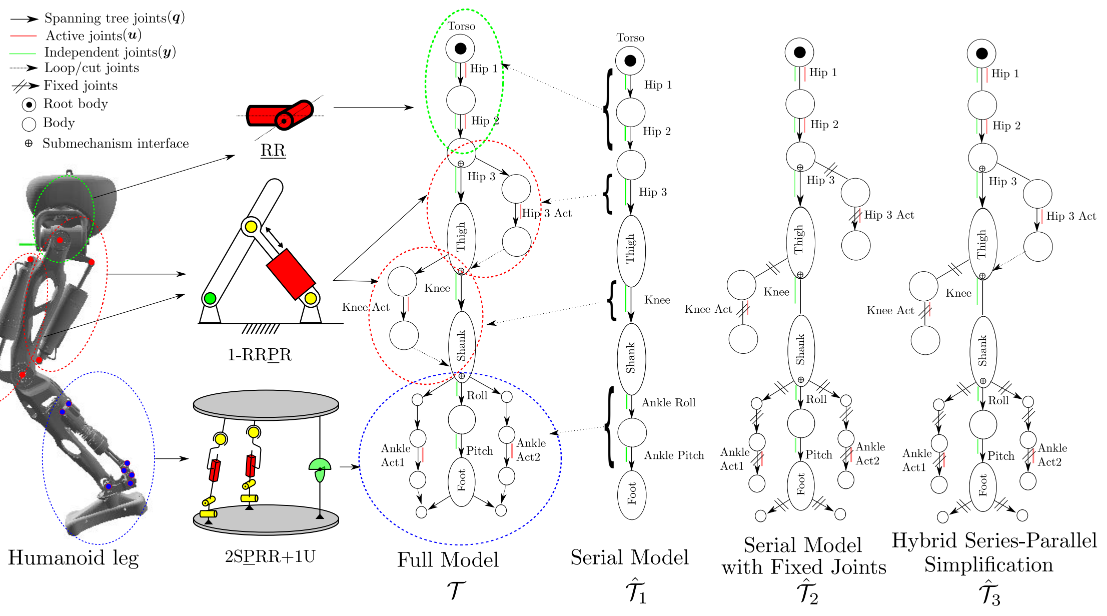


Figure 9.10: Serial-Parallel hybrid composition of the RH5 leg and assumed simplifications

Model order reduction (MOR) techniques aim to reduce the computational complexity of large scale mathematical models in computer simulations. Traditionally, this technique has been used for problems in fluid mechanics [Lassila et al., 2014] and structural mechanics [Wu and Tiso, 2016] to reduce the simulation time while making little compromises on the accuracy. As the complexity of rigid multi-body systems used in robotics is increasing, the need of MOR becomes indispensable for simulation and real-time control of modern contemporary robots. While this term is not classically used in the area of rigid body dynamics, the use of model simplification approach is common nevertheless. However, the trade off between the complete dynamic model and simplified dynamic model has not been reported in the literature. In that vein, we first briefly discuss the model simplification used in the state of the art. Then, we present a model simplification study on RH5 humanoid leg using HyRoDyn using a computational performance metric and relative power error metric for modeling simplification error.

9.4.1 Model Simplification in SOTA

Series-parallel hybrid robots are highly complex mechatronic systems and generic treatment of such robots remains an open problem. However, modularity in robot design allows for certain abstractions which simplifies their modeling and control. Such abstractions are shown in Fig. 2.6. While Fig. 2.6(a) captures the true complexity of the robot, due to absence of generic methods to model and control such systems, three different abstractions namely actuation space, independent joint space and task space, are adopted to simplify the modeling and control (for details, see Section 2.3). As pointed out before, the computation of full inverse dynamic model for hybrid robots can be computationally expensive due to the large size of their spanning trees and the large number of loop closure constraints to be resolved. For example, RH5 humanoid [Peters et al., 2017] which only contains relatively simple parallel mechanism modules (with less than 3 DOF) has 32 DOF ($m = p = 32$), $c = 15$ independent closed loops and $n = 76$ DOF in its spanning tree. The moving parts inside a parallel submechanism module may have relatively small contribution to the overall dynamics of the system which is essentially due to dynamics of major link segments lying on the trunk of the spanning tree and joint friction etc [Buschmann et al., 2013]. Hence, they may be left unmodeled or their mass-inertia properties can be merged to the larger link segments which are relevant to the independent joint space of the robot. Assuming actuators are the ideal torque source in the system, a simplified inverse dynamic model (see Fig. 2.6(c)) in independent joint space is often combined with an inverse static model in actuation space (see Fig. 2.6(b)) to compute the actuator forces [Hopkins et al., 2015], [Vonwirth, 2017] using

$$\boldsymbol{\tau}_u = \boldsymbol{G}_u^{-T} \boldsymbol{\tau}_y \tag{9.2}$$

This approach is used in model based torque controlled series-parallel hybrid humanoids such as THOR [Hopkins et al., 2015], Valkyrie [Paine et al., 2015], Lola [Buschmann et al., 2013] etc.

9.4.2 Model Simplification Study on RH5 Leg

The subject of interest within this study is the leg of the RH5 humanoid [Peters et al., 2017], currently being developed at the DFKI-RIC. Fig. 9.10 shows the serial-parallel hybrid mechanism consisting of 4 individual submechanism of types $\underline{R}R$, 1-RR$\underline{P}$R and 2S$\underline{P}$RR+1U. The full model of the leg ($\mathcal{T}$) has $m = 6$ independent degrees of freedom, $p = 6$ active joints and $n = 18$ spanning tree joints. Further, we introduce three different model simplifications from the full description of the robot:

1. $\hat{\mathcal{T}}_1$ is a subgraph of $\mathcal{T}$ containing only the independent joints neglecting all other branches,
2. $\hat{\mathcal{T}}_2$ is a subgraph of $\mathcal{T}$ containing only the independent joints where bodies separated by the cut joints in $\mathcal{T}$ are merged to parent and child links of each parallel submechanism module with the help of fixed joints [1],
3. $\hat{\mathcal{T}}_3$ is a subgraph of $\mathcal{T}$ which includes the Hip3 submechanism but the rest of the kinematic chain is kept serial like $\hat{\mathcal{T}}_2$. For the simplified models, it is assumed that actuators are still the ideal sources of forces or power.

Further, the bi-directional kinematic mapping between the independent joint space and the actuator space is preserved. In the following, we present a task space trajectory input to the models to study the effect of neglected dynamics.

9.4.2.1 Metrics

The following metrics are used in the model simplification study.

1. **Computation Time**: A first metric is given by the raw computation time of the inverse dynamics problem, $t_{ID} \in \mathbb{R}^+$. It relates to the computational effort of the algorithm. All computations have been performed on a standard laptop with Intel Core i7 CPU @ 2.8 GHz using HyRoDyn software tool.
2. **Power Error Metric**: The power $P \in \mathbb{R}^+$ is invariant to coordinate transformation and can therefore be used to describe the input-output behaviour of a system.

$$P = \boldsymbol{\tau}_q^T \dot{\boldsymbol{q}} = \boldsymbol{\tau}_u^T \dot{\boldsymbol{u}} = \boldsymbol{\tau}_y^T \dot{\boldsymbol{y}} \tag{9.3}$$

The total power can be expressed as a sum of power of each individual submechanism $i \in \{1, \dots, s\}$.

$$\Delta P = \left\| \sum_{i=1}^{s} \Delta\boldsymbol{\tau}_{ui}^T \dot{\boldsymbol{u}}_i \right\|_1 \tag{9.4}$$

The relative power error can then be defined as the ratio $\frac{\Delta P}{P}$.

[1]It should be noted that the total mass of $\hat{\mathcal{T}}_1$ is less than $\mathcal{T}$ and $\hat{\mathcal{T}}_2$ and $\mathcal{T}$ have equal masses.

9.4.2.2 Task Space Trajectory

We present a comparison between the full model ($\mathcal{T}$) and simplified models ($\hat{\mathcal{T}}_i, i \in \{1,2,3\}$) in task space control of the humanoid leg. Two set points i.e. foot up position & foot down position, separated by 15 cm in z-direction, are chosen in the task space of the robot. Using the inverse kinematics, the independent joint positions are computed which can be done either numerically or analytically. Then, these way-points in independent joint space are fed to an interpolator which provides smooth trajectories ($\boldsymbol{y}, \dot{\boldsymbol{y}}, \ddot{\boldsymbol{y}}$) for up and down movement of the leg. Fig. 9.4 shows the task space and independent joint space trajectories for the RH5 leg. These trajectories are then used to compute the actuator trajectories ($\boldsymbol{u}, \dot{\boldsymbol{u}}, \ddot{\boldsymbol{u}}, \boldsymbol{\tau}_u$) using inverse kinematics and inverse dynamics algorithms as presented earlier. Fig. 9.5 shows the position, velocity, acceleration required in different actuators of the robot to produce this movement for all the models as they share the same inverse kinematics mapping. The effect of neglected dynamics due to model simplification can be observed in Fig. 9.11 which compares the acuator forces between the two models ($\mathcal{T}$ & $\hat{\mathcal{T}}_2$). It can be noticed that the highest error is in Hip 3 Act joint.

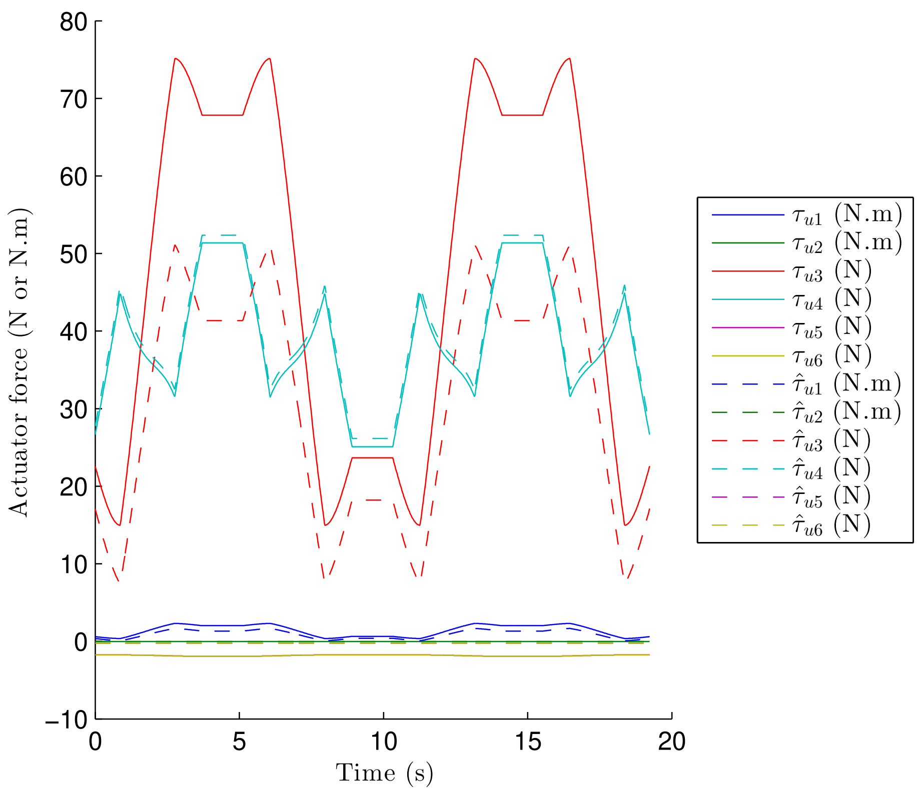


Figure 9.11: Actuator forces comparison between the full model $\mathcal{T}$ and simplified model $\hat{\mathcal{T}}_2$

Table 9.2: Comparison of CPU time and relative power error between the full and simplified models

Model (n)	Rel. Power Error ($\frac{\Delta P}{P}$%)	t_{ID} (µs)	Speedup in t_{ID} w.r.t. $\mathcal{T}$ (%)
$\mathcal{T}$ (18)	-	32.03	-
$\hat{\mathcal{T}}_1$ (6)	53.24	12.08	62.28
$\hat{\mathcal{T}}_2$ (6)	12.43	12.08	62.28
$\hat{\mathcal{T}}_3$ (8)	3.05	14.32	55.41

9.4.2.3 Discussion

In general, it is observed in the study that $\hat{\mathcal{T}}_1$ performs worse than $\hat{\mathcal{T}}_2$ due to the missing masses and the results have been skipped here for brevity. The branches in Hip 3 and Knee submechanisms have relatively higher masses and inertias than the branches in the Ankle submechanism and hence are more prone to influencing the error in overall dynamics of the system. The comparison of average CPU time for solving inverse dynamics (per call) and relative error in power between the full and simplified models is shown in Table 9.2 for the workspace trajectory following study. It can be noticed that model simplifications definitely provide a computational advantage in solving inverse dynamics for real time control. In the state of the art, it has been considered as a rule of thumb that a simplified model neglecting all the closed loops is sufficient for dynamic control. However, this is not true as certain closed loops may affect the error dynamics more than others and simplified models respecting those loop closure constraints outperform these models in terms of accuracy without compromising much on the CPU time. For example, the simplified model $\hat{\mathcal{T}}_3$ which respects the loop closure constraints in Hip3 submechanism (see Fig. 9.10) adds only 2 extra DOF to the spanning tree in comparison to $\hat{\mathcal{T}}_2$ brings down the relative power error to ≈ 3 % while taking only ≈ 2µs extra in computation of inverse dynamics.

To find a reduced order model which decreases the computational effort without compromising much on the accuracy is an interesting problem. This could also provide some insights on how to select the appropriate set of cut joints and generalized coordinates to properly describe the system. If these are known subjected to some offline analyses, HyRoDyn can also be used for computing the simplified inverse dynamic model.

9.5 Application in Robotic Rehabilitation

The inital need of the development of HyRoDyn software tool came from Recupera-Reha project [Recupera-Reha, 2018] which involved the development of a light weight

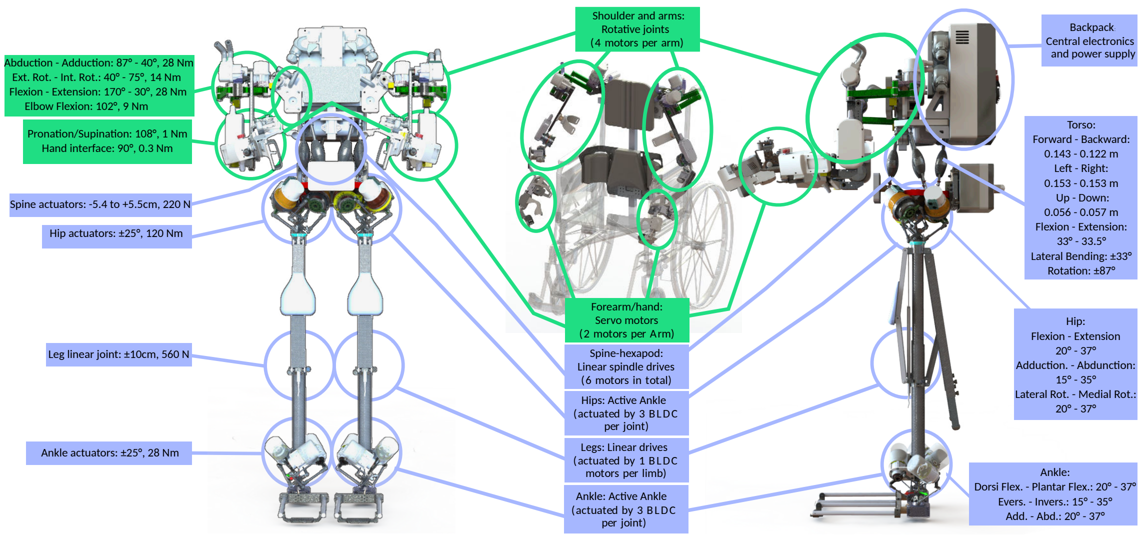

Abduction - Adduction: 87° - 40°, 28 Nm
Ext. Rot. - Int. Rot.: 40° - 75°, 14 Nm
Flexion - Extension: 170° - 30°, 28 Nm
Elbow Flexion: 102°, 9 Nm
Pronation/Supination: 108°, 1 Nm
Hand interface: 90°, 0.3 Nm
Spine actuators: -5.4 to +5.5cm, 220 N
Hip actuators: ±25°, 120 Nm
Leg linear joint: ±10cm, 560 N
Ankle actuators: ±25°, 28 Nm
Shoulder and arms: Rotative joints (4 motors per arm)
Forearm/hand: Servo motors (2 motors per Arm)
Spine-hexapod: Linear spindle drives (6 motors in total)
Hips: Active Ankle (actuated by 3 BLDC per joint)
Legs: Linear drives (actuated by 1 BLDC motors per limb)
Ankle: Active Ankle (actuated by 3 BLDC per joint)
Backpack: Central electronics and power supply
Torso: Forward - Backward: 0.143 - 0.122 m Left - Right: 0.153 - 0.153 m Up - Down: 0.056 - 0.057 m Flexion - Extension: 33° - 33.5° Lateral Bending: ±33° Rotation: ±87°
Hip: Flexion - Extension 20° - 37° Adduction. - Abdunction: 15° - 35° Lateral Rot. - Medial Rot.: 20° - 37°
Ankle: Dorsi Flex. - Plantar Flex.: 20° - 37° Evers. - Invers.: 15° - 35° Add. - Abd.: 20° - 37°

Figure 9.12: RECUPERA exoskeleton configurations (left and right: full-body, middle: wheelchair) with its ROM and force/torque capabilities. Modules highlighted in green are used in both systems, mechanisms highlighted in blue exist only in the fullbody system. Full-body system (weight = $42\,\mathrm{kg}$, height = $1.70 - 1.90\,\mathrm{m}$) is driven by 28 brushless direct current (BLDC) and 4 servo motors. Wheelchair system (weight = $8.6\,\mathrm{kg}$, $29.2\,\mathrm{kg}$ including commercial wheelchair) is driven by 8 BLDC and 4 servo motors.

and modular full body exoskeleton for the purpose of robot-assisted rehabilitation of neurological diseases. To make the design highly modular and lightweight, various combinations of serial and parallel actuated submechanisms are used to obtain a series-parallel hybrid design. Fig. 9.12 shows the overview of the developed system. Its lightweight design and modularity allows to use the system in different configurations, e.g., (1) as a 10 DOF upper body wheelchair-mounted part system or (2) as a 30 DOF full-body system with legs to support sitting and standing postures. The wheelchair system is designed to support the hemiparetic stroke patients who are usually bounded to a wheelchair in their early days after stroke. Once, they start to recover, they can continue training with the full body exoskeleton system which provides them the possibility to perform upper body exercises while both sitting and standing. Both systems are designed such that the patient does not have to carry the weight of the exoskeleton as the weight of the system is compensated by the controller and transferred to the ground. Further, the mobility of the patient is not compromised. Due to its bilateral design, the system can be used with both left and right sided stroke patients and can provide extended training possibilities e.g. dual arm tasks, and mirror therapy [Kumar et al., 2019b]. In this section, the control architecture of the exoskeleton, which uses HyRoDyn as a central component, is described along with some experimental results.

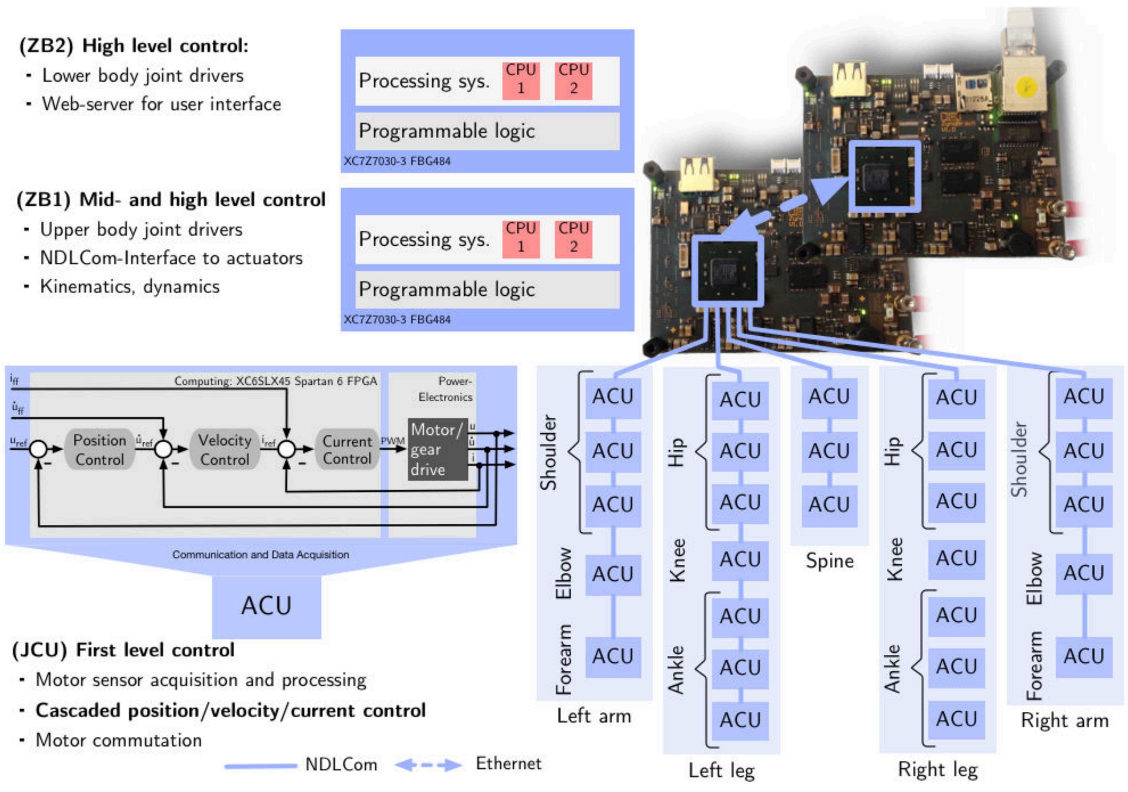


Figure 9.13: Hybrid centralized-decentralized control and network architecture. The exoskeleton is controlled by two central systems, called ZynqBrain (ZB), and a network of decentralized Actuator Control Units (ACU) for BLDC motor control. Each motor is controlled locally by an adjacent ACU. The distributed ACUs are interconnected via an NDLCom network. ACUs that belong to a specific limb form a subsystem. Bottom left: cascaded actuator-level control architecture implemented on the FPGA of each ACU; u is the angular position, $\dot{u}$ is the angular velocity and i is the motor current. The reference values are provided by the mid-level control on the ZBs. All ACUs continuously send telemetry status data including u, $\dot{u}$ and i to ZB1 with a frequency of $100\,\mathrm{Hz}$ using the NDLCom network.

9.5.1 Exoskeleton Control

9.5.1.1 First-Level Control

Each actuator is driven by an FPGA that implements the first level control architecture for the joint using a cascaded position, a velocity, and a current control loop. Each of the control cascades can be directly selected for control. Fig. 9.13 shows a block diagram illustrating the actuator level control architecture. The motors can be torque controlled with the help of motor current measurements. The actuator level modularity permits the implementation of decoupled safety checks from the mid-level controllers. For example, the position, velocity and current are limited via maximal values, and the controller is stopped in case the sensors fail at some point. This low level control architecture meets the requirements for the therapy concepts to be implemented in the system and constitutes a solid foundation for both kinematic and dynamic control which is implemented at mid-level.

9.5.1.2 Mid-Level Control

Mid-level control architecture implements the kinematic and dynamic model of the system and associated control approaches for rehabilitation therapies using HyRoDyn. In the (1) *Gravity Compensation* (GC) mode, the weight of the system is compensated with the help of an inverse dynamic model of the exoskeleton arms. One can also use GC mode to take into account the dynamics of human arms. The input to this model is the actuator positions read from the position encoders and the output is the reference torque values which is then converted into motor current and sent to the current controller implemented in the ACU. The GC mode is used to implement a transparent behavior of the system and represents the *basic operation mode* of the system, on which most of the other modes are based. To support repetitive movement therapies, (2) *Teach & Replay* (TR) can be used. This mode consists of two phases. First, the affected arm is put in the gravity compensation mode so that a therapist can easily move the arm. The forearm is equipped with a touch sensor which recognizes the intention of the therapist to teach a trajectory and stores the trajectory (position and velocity readings from the involved ACUs) in the system's storage device. Subsequently, the trajectory can be replayed according to a trigger by the patient or therapist. During the replay, the exoskeleton executes the trajectory movement in the cascaded position-velocity control mode in the ACU. Mirror therapy can be supported using the (3) *Master-Slave* (MS) mode, where movements from the healthy arm can be transferred to the unhealthy arm in a mirrored fashion. In this mode, the healthy arm is kept in the gravity compensation mode and the actuator positions read from the healthy arm are mirrored and sent to the ACUs of the un-

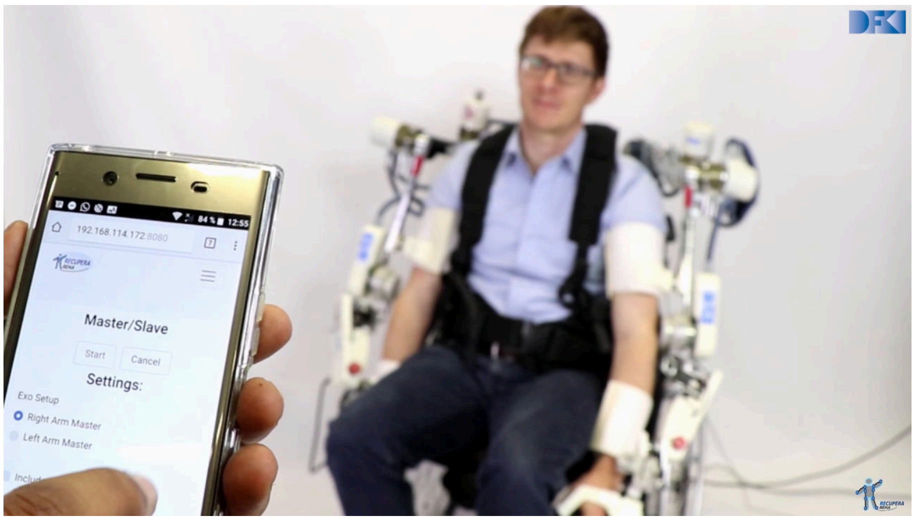


Figure 9.14: Web GUI for high level control

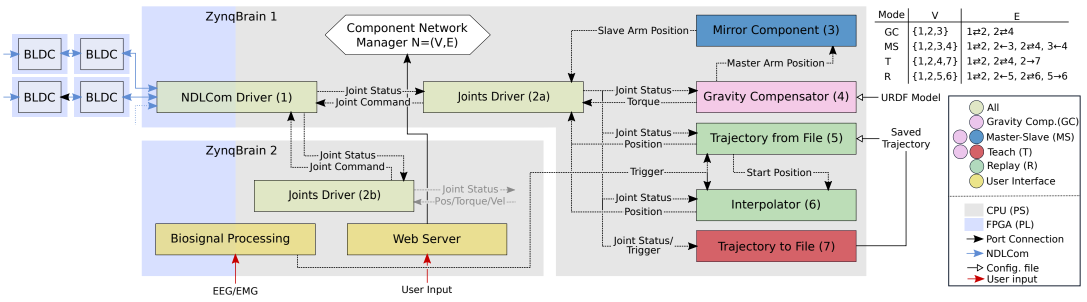
healthy arm, which is in cascaded position-velocity control mode. Further, sitting and standing features for the lower part of full body exoskeleton are implemented at this level.

9.5.1.3 High-Level Control

The high level control of the exoskeleton can be managed by a web-based GUI using a mobile phone or tablet (see Fig. 9.14). The Python Flask framework [Ronacher, 2017] was used to implement the web application. The Flask web server hosted on ZynqBrain2 was used as the server application. The GUI allows the therapist to select the therapy mode (for e.g. GC mode, MS mode, TR mode etc) or to manage different patient profiles, e.g., patient specific data and movements. Moreover, it allows the therapist to use the exoskeleton in different settings: single arm, dual arm, full body etc. Both left and right sided stroke patients can be supported.

9.5.1.4 Software Implementation

The mid and high level control is implemented using the Robot Construction Kit (Rock)[Joyeux, 2010] which is based on the component model of the Orocos Real Time Toolkit (RTT) and the object request broker (ORB) implementation, omniORB. Components (or tasks) encapsulate different functionalities, run independently and provide input and output for other components (see Fig. 9.15). A central component in the software architecture is the HyRoDyn orogen component described in Section 8.5 which is used for accessing the kinematic and dynamic models of the exoskeleton system (for e.g. Gravity Compensator in Fig. 9.15 is simply an instance of it). The configuration can be applied to each component individually. This allows adjustments to



Mode	V	E
GC	{1,2,3}	1⇄2, 2⇄4
MS	{1,2,3,4}	1⇄2, 2←3, 2⇄4, 3←4
T	{1,2,4,7}	1⇄2, 2⇄4, 2→7
R	{1,2,5,6}	1⇄2, 2←5, 2⇄6, 5→6

Figure 9.15: Software architecure overview: A Component Network Manager configures, connects and starts the subset of components (V) required for a specific mode. The corresponding directed connections (E) are described in the table in the top right corner. The components required for each mode are also represented by colors. A web server application hosted on ZynqBrain 2 is used as user interface.

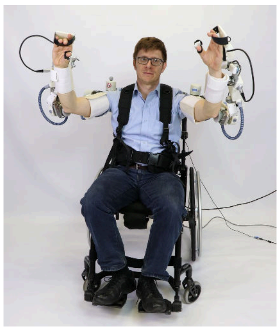
the system in a very flexible way and to distribute computational demanding components among the two ZynqBrains. Additionally, a web server is running on the second ZynqBrain providing access to a web application written in JavaScript and provides a user interface for the therapist or patient, e.g., to select a mode, adjust settings or observe the current state of the exoskeleton (see Sec. 9.5.1.3). Furthermore, some functionalities can be triggered by biosignals like EEG or EMG as the biosignal processing can be done on the embedded processors [Kirchner et al., 2016, Wöhrle et al., 2017].

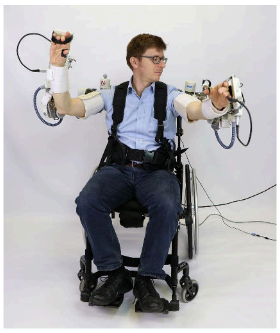
9.5.2 Therapy Modes

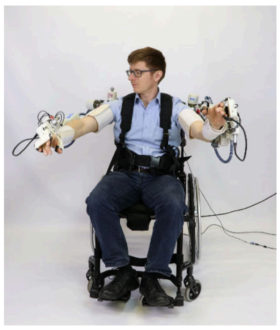
This section presents the experimental results of the rehabilitation therapies implemented on the wheelchair configuration. Since the upper body design is identical for both configurations, the results are equally valid between them. All the therapy modes can be used in both sitting and standing modes with the Recupera full body exoskeleton (see Fig. 9.23).

9.5.2.1 Gravity Compensation Mode

Fig. 9.16 shows four different poses during the dual arm gravity compensation mode with the human subject inside the exoskeleton. Due to the good quality of the gravity compensation model, the human subject can move its arms freely within the system. Optionally, human arm weight compensation can also be included in the model using the concept of mimic joints [Kumar et al., 2017b]. Another application of this functionality is the get in helper mode (see Fig. 9.17) which allows easy entry of the human subject inside the exoskeleton system. Thanks to the transparent behaviour of the exoskeleton arms in this mode, the human subject can wear them in a safe and user-friendly manner. Healthy subjects can enter the dual arm exoskeleton in less


(a) Pose 1


(b) Pose 2


(c) Pose 3

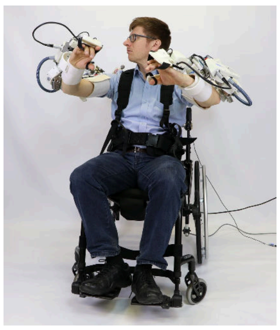
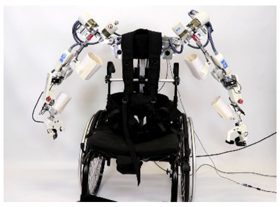
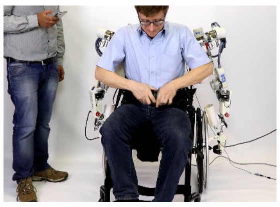
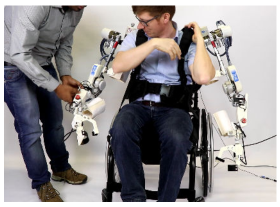
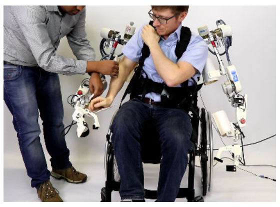

(d) Pose 4

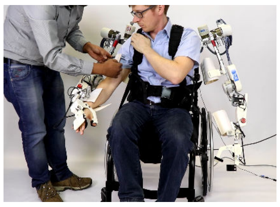
Figure 9.16: Gravity compensation mode with human subject

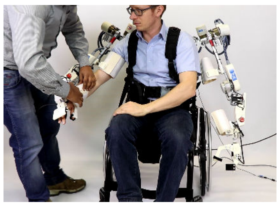

(a) Initial position

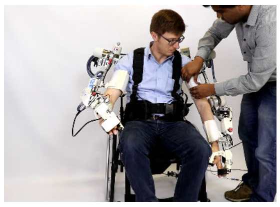

(b) Waist strap


(c) Shoulder straps


(d) Insert right arm


(e) Attach upper arm


(f) Attach forearm


(g) Insert left arm

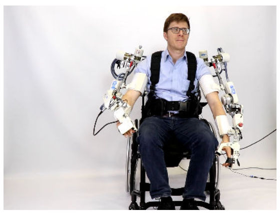
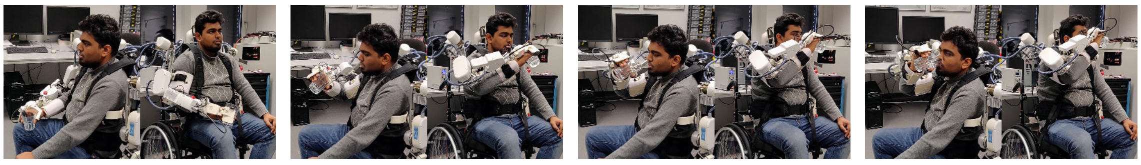

(h) System ready

Figure 9.17: Get in helper mode: To enable easy entry of the human subject inside the exoskeleton system, this mode sends the two exoskeleton arms into a special initial position as shown in (a) and switches to GC mode. When human subject sits in the wheelchair, the (b) waist straps, and the (c) shoulder straps are secured. Then, the subject (d) inserts its right arm in the system, and the (e) upper arm and (f) forearm are attached to the exoskeleton system. Similarly, (g) the left arm is inserted and secured to the exoskeleton system. The system is ready to be used in (h) dual-arm setup with the human subject.

than a minute on average provided no passive adjustments in the system are to be done.

9.5.2.2 Teach & Replay Mode

To demonstrate Teach & Replay therapy, drinking and pointing trajectories were taught to the system and were replayed ten times with and without human subject. Fig. 9.19 and Fig. 9.20 show the error band plots between the commanded trajectories


(a) Pose 1 (b) Pose 2 (c) Pose 3 (d) Pose 4

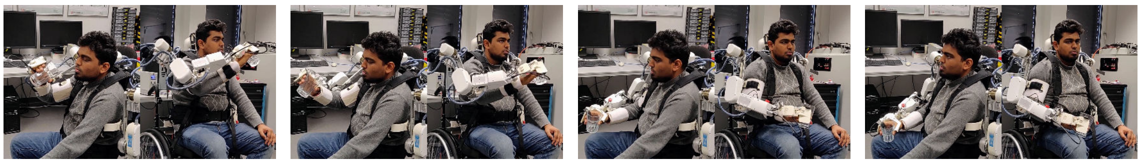
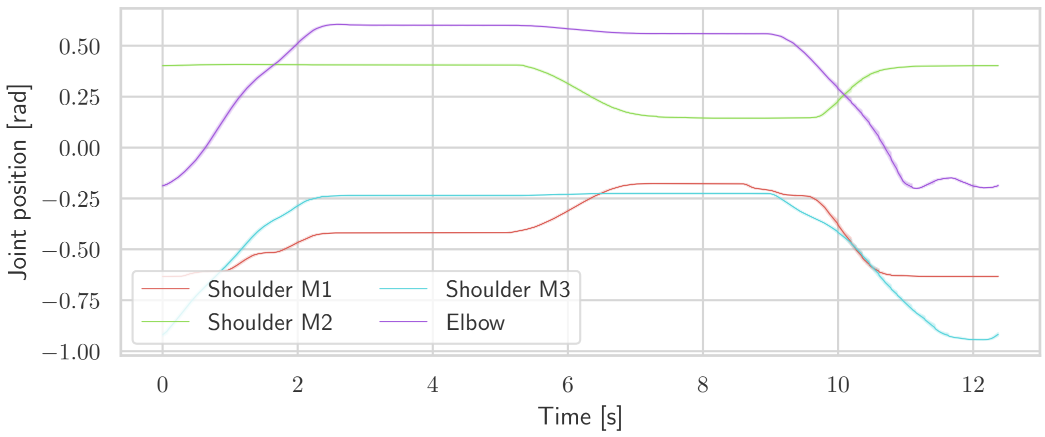
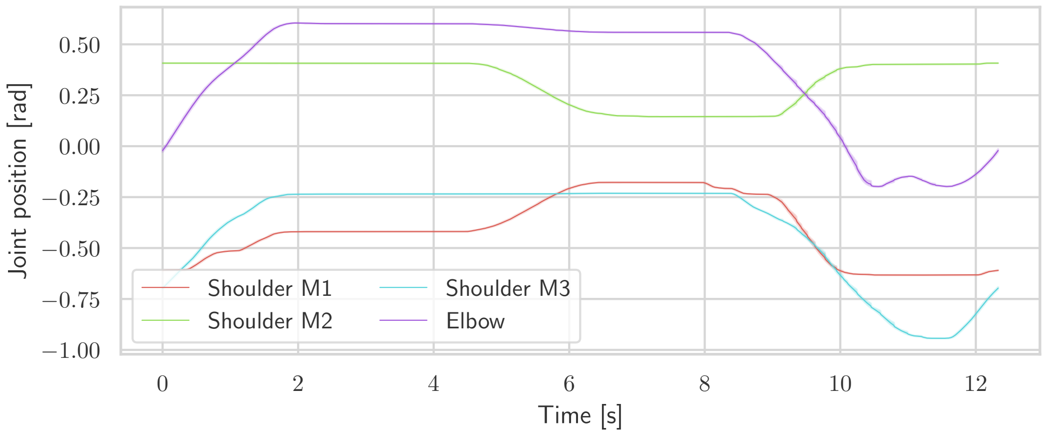

(e) Pose 5 (f) Pose 6 (g) Pose 7 (h) Pose 8

Figure 9.18: Teach and Replay mode with human subject for drinking movement

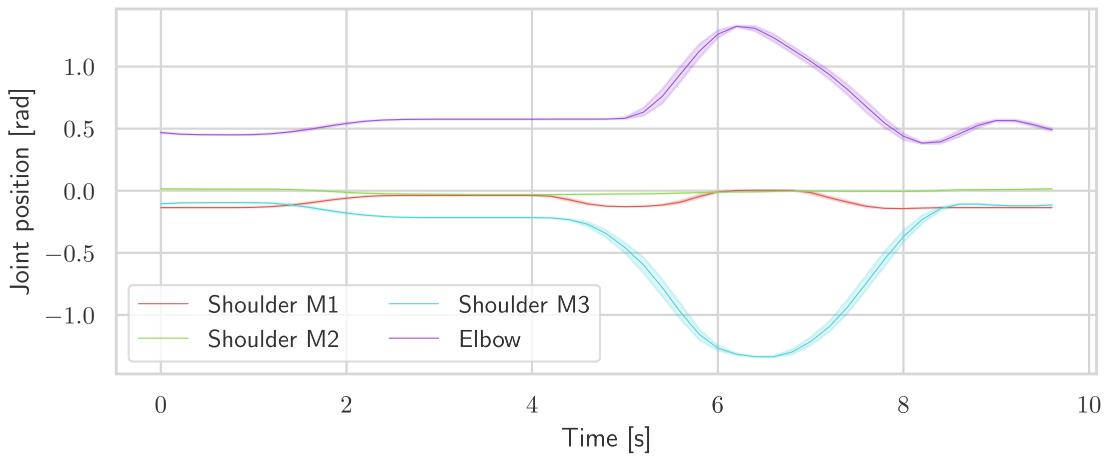


(a) Drinking: with human (Overall RMSE = 0.02243)

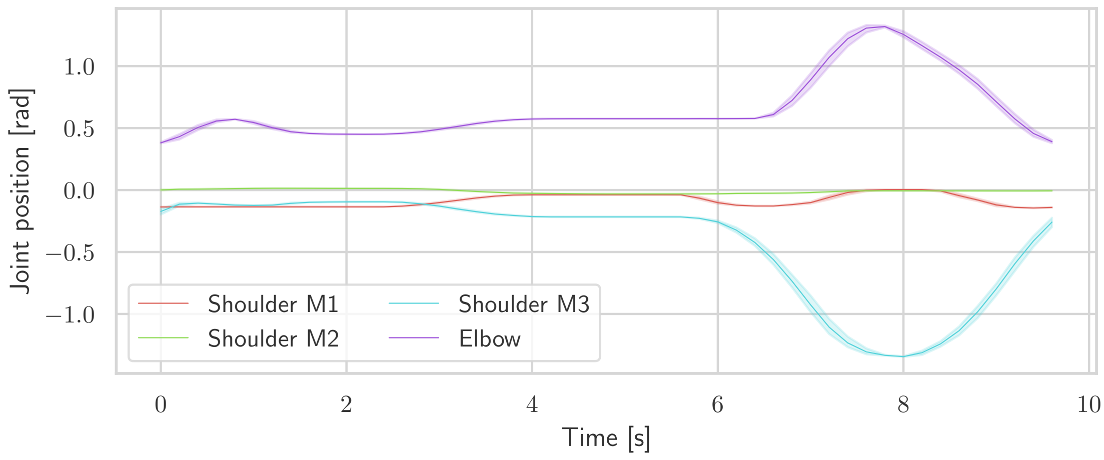


(b) Drinking: without human (Overall RMSE = 0.03030)

Figure 9.19: Teach & Replay therapy for drinking movement



(a) Pointing: with human (Overall RMSE = 0.05215)



(b) Pointing: without human (Overall RMSE = 0.05287)

Figure 9.20: Teach & Replay therapy for pointing movement

and joint status recorded from the exoskeleton and also notes the root mean square error (RMSE) for the two taught movements. It can be observed that the system performs better in the presence of a human subject as the human body acts as a damper to the small vibrations occuring due to structural flexibilities. Fig. 9.18 shows the motion sequence during the drinking movement with a human subject inside the exoskeleton system.

9.5.2.3 Master-Slave Mode

Master-Slave mode can be used in the Mirror therapy which involves mirroring the movements from the healthy arm (in GC mode) to the position controlled affected arm. Fig. 9.21 shows the joint position data recorded from the master and slave arms

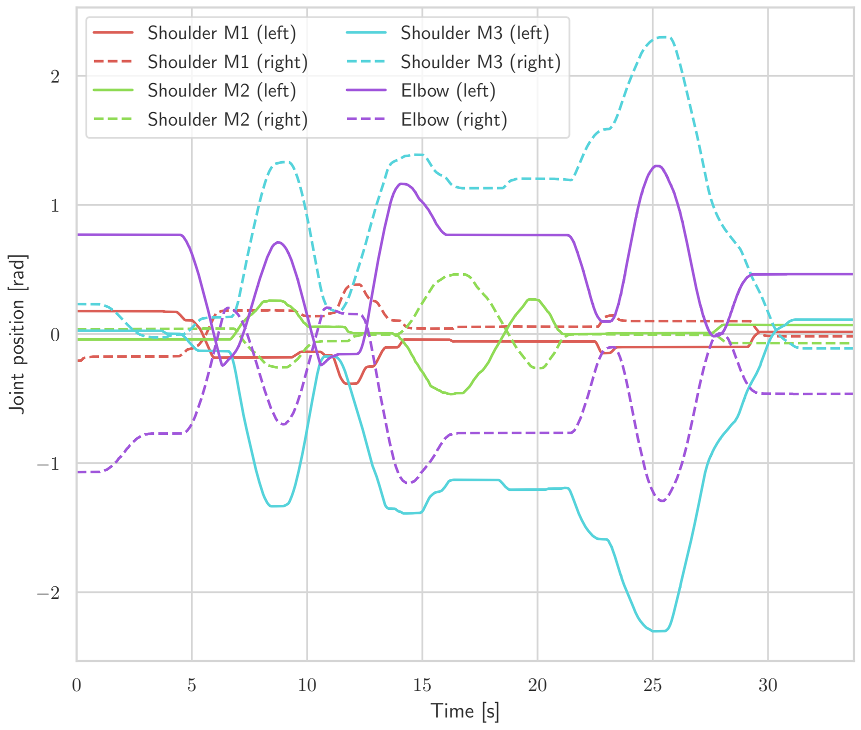
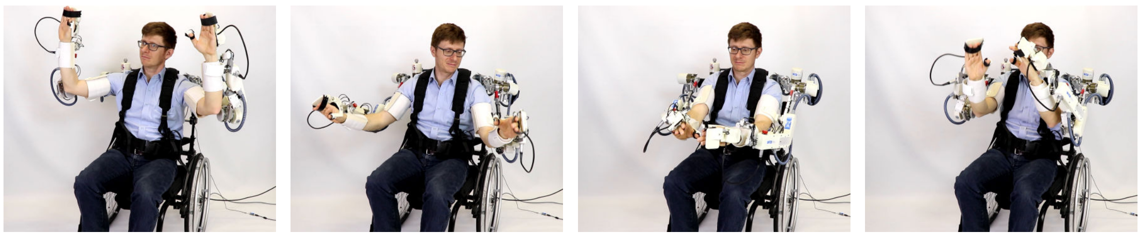


Figure 9.21: Mirror therapy



(a) Pose 1 (b) Pose 2 (c) Pose 3 (d) Pose 4

Figure 9.22: Master slave mode with human subject

during a mirror therapy session. This mode is ideal for increasing the self-training time of the stroke patients as they can train themselves by naturally copying the movements from their healthy arm to the affected arm in the minimal supervision of the therapists. Fig. 9.22 shows four different poses from the master slave mode with the system with a human subject inside.

9.6 Conclusion

This chapter presents the results of application of HyRoDyn software tool in simulation, analysis and real time control of highly complex series-parallel hybrid robotic systems such as the Recupera-Reha exoskeleton and RH5 humanoid. It enables ef-

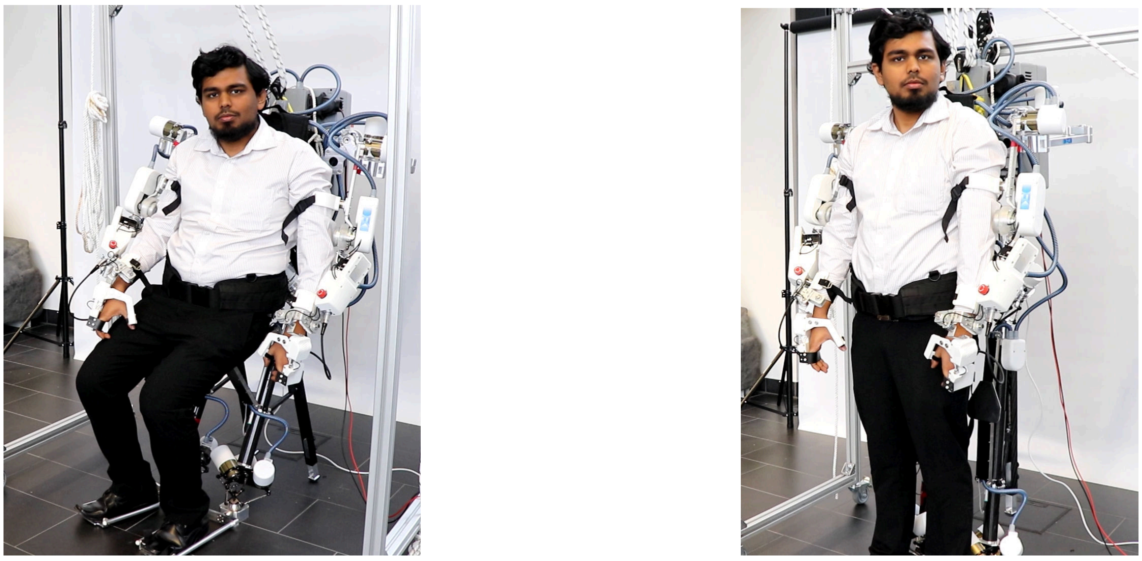

(a) Sitting posture (b) Standing posture

Figure 9.23: Sitting and Standing with RECUPERA full body exoskeleton

ficient and error free computation of kinematics and dynamics of such systems and forms the basis of their real time control. Further, it provides insights into model simplification of complex multi-body systems. Finally, the application of this software tool in a real application has been demonstrated. Overall, it can be concluded that HyRoDyn is a powerful software for modeling very complex robotic systems and provide a variety of tools to deal with different aspects of series-parallel hybrid robots thereby helping the designers and control engineers alike in developing the future robotic systems.

Part V

Conclusion

Chapter 10

Conclusion and Outlook

This chapter is the synopsis of this thesis and presents the main scientific contributions of the work. As an outlook, some insights for future work are described at the end.

10.1 Thesis Summary

The general motivation of this thesis stems from the growing popularity of series-parallel hybrid architectures in robotics. These robots combine the advantages of serial and parallel architectures and have significant mechanical advantages in terms of better payload-to-weight ratio, better stiffness property, good dynamic characteristics etc. Hence, this design trend is clearly reflected in various robot designs not only at DFKI-RIC (for e.g., AILA, Mantis, Charlie, Recupera Exoskeleton, RH5 humanoid etc.) but also around the world, for e.g. Lola (TUM), Valkyrie (NASA), THOR (Virginia Tech.) etc. On the downside, they also inherit the kinematic complexities of serial and parallel designs. Hence, the overall kinematic complexity of robot architectures is steadily increasing as the advantages of a series-parallel hybrid designs are becoming evident. In order to fully exploit their potential, it becomes very important to have their systematic analysis and accurate kinematic and dynamic modeling. Notwithstanding this hurdle, it is becoming equally important to develop robot software capable of handling this complexity for designers and control engineers so that they can develop and optimize high performance robotic systems. Modularity and model-based software development are keys to handle this complexity.

To lay a strong foundation, an extensive survey on various series-parallel hybrid robots was performed in order to study their design aspects in mechanics, electronics and software domains. The key insight from this survey was that these robots utilize parallel mechanisms as an abstraction of different kinematic joints and it is quite common to use different variants of the same parallel mechanism in order to build different kinematic joints. In most cases, it can be observed that the designer utilize these mechanisms as a higher DOF actuator module with additional mechanical advantages. This approach was also followed in two different series-parallel hybrid robotic systems namely Recupera-Reha exoskeleton and the RH5 humanoid being developed at DFKI-RIC during the thesis period.

Two new parallel mechanisms namely Active Ankle (used for constructing Hip and Ankle joints in Recupera-Reha exoskeleton) and 2SPRR+1U (used for ankle joint and its variant 2SPU+1U for torso and wrist joints in RH5 humanoid) were invented. To get a complete understanding of their geometric behavior, a rigorous and comprehensive kinematic analysis of these mechanisms were performed and the solutions to their forward and inverse kinematics problems were derived. Additionally, tools from computational algebraic geometry were used to get some global insights into their geometry. This analysis is not only useful for the purpose of kinematic control but also helps designers in improving their design for tailored application cases.

The next natural question to ask was how one could use the insights from a comprehensive kinematic analysis of these mechanisms for solving the kinematics and dynamics of the overall series-parallel hybrid robotic system. Instead of resorting to numerical resolution of loop closure constraints, analytical loop closure functions for these mechanisms were derived. Then a modular approach for the topological modeling of these systems was conceptualized based on which the loop closure functions of the overall hybrid system is composed in a modular fashion. The modularly composed loop closure function demonstrates block diagonal structure which can be exploited in various kinematics and dynamics algorithms.

This approach is implemented in a software framework called HyRoDyn written in C++. The robot description for HyRoDyn can be generated from a visual editing tool called Phobos which allows both bottom-up and top-down modeling approaches. Presently, closed form solutions to mechanisms such as 1-RRPR, 2-SPU+1U, 2-SPRR+1U, 6-RUS, 6-UPS, parallelogram chains are available in its submechanism libraries and the software can be used to analytically solve the kinematics and dynamics of arbitrary series-parallel hybrid robots composed of these submechanism modules. Actuation of the robot can be arbitrarily selected. The software has been successfully used in the analysis and control of some complex series parallel hybrid robots such as the Recupera-Reha exoskeleton and the RH5 humanoid.

10.2 Scientific Contributions

Following is the list of concrete scientific contributions (linked with respective publications) arising out of this work.

- **Survey on series-parallel hybrid robots:** A systematic survey of various series-parallel hybrid robotic systems developed in the last decades in the field of legged robotics, exoskeletons and industrial automation has been performed [Kumar et al., 2019c]. Their designs are studied in mechanics, electronics and software domains to develop a thorough understanding of the state of the art. It has been found that most of these hybrid robots utilize parallel submechanisms as an abstraction to a higher degree of freedom active joint (for e.g. universal, spherical, six dof) leading to an inherently modular design. This lets the designers exploit the non-linear transmission, enhance the stiffness and dynamic capabilities of the robot and produce a light weight design. Further, this review also highlighted the approaches that have been adopted in their modeling and control including the underlying assumptions and their limitations. The insights from this review forms the basis of this work i.e. submechanism based modularity should be reflected in the kinematic and dynamic modeling of these robots.

- **Kinematic Analysis of 2SPRR+1U mechanism:** The novel 2SPRR+1U mechanism and its existing variant 2SPU+1U mechanism for the abstraction of a universal joint has been studied extensively. This study, published in [Kumar et al., 2018c], involves exhaustive kinematic analyses which provide solutions to forward and inverse kinematic problems for these mechanism types. The use of computational algebraic geometry provides some global insights into the mechanism geometry for example, an upper bound to the maximum number of assembly modes, global description of its singularity curve etc. Further, workspace analysis and its force and velocity transmission capabilities are presented. These results are of both theoretical and practical interest for roboticists interested in utilizing the 2SPRR+1U mechanism or 2SPU+1U mechanism for an integrated 2 DOF universal joint unit.

- **Kinematic Analysis of Active Ankle mechanism:** A comprehensive study of design, analysis and control of the novel 3R-[2SS] mechanism also called as Active Ankle [Simnofske et al., 2016] developed for the abstraction of a spherical joint has been performed. This mechanism demonstrates an almost spherical motion in $SE(3)$ i.e. the primary motion of this device is spherical but is coupled with small translations that can be neglected for most practical applications. Due to this behavior, its kinematic analysis is very interesting. The solution to inverse kinematics problem is not sufficient for its kinematic control and hence, it is important to solve a rotative inverse kinematics problem which asks for a pose in $SO(3)$ instead of $SE(3)$ and provides the joint angles needed to achieve the pose along with an estimation of end effector shift. These results are published in [Kumar et al., 2018a]. Further the results from its kinematic control in task space is available in [Kumar et al., 2016].

Further, tools from computational algebraic geometry are used for solving the forward kinematics problems which provide several global insights into the mechanism's behavior. It is established that the upper bound to the number of unique solutions to forward kinematic problem is 40 which supports our observation that once the actuator angles are fixed in the three legs, ACTIVE ANKLE behaves as a special instance of $6-6$ Stewart platform. In practice, a maximum of 16 real solutions of the forward kinematic problem were found. In addition, the results of the torsional motion analysis which is of practical interest is presented and some singularities of the mechanism are highlighted. Moreover, the assembly modes where the mechanism behaves as an *almost-spherical* device are identified. These results are published in [Kumar et al., 2018b].

- **Modular and Analytical Approach for Kinematic and Dynamic Modeling of Hybrid robots:** The main contribution of this thesis is the modular and analytical approach for the kinematic and dynamic modeling of series-parallel hybrid robots. The notion of modularity is derived from an extensive survey on series-parallel hybrid robots which showed that parallel mechanisms are used as an abstraction to certain kinematic joints. Since, different variants of the same mechanism are used again and again in the same robot, it makes sense to utilize the closed form analytical solutions to the loop closure constraints based on the type of parallel mechanism used in the design. This leads to analytical derivation of loop closure function for the overall robot where the modularity reveals a block diagonal structure which can be exploited in various kinematics and dynamics algorithms. This results in a computationally efficient and error free formulation of these problems. This approach is published in [Kumar and Mueller, 2019].
- **HyRoDyn Software Framework:** The modular and analytical approach is implemented in the form a software framework called Hybrid Robot Dynamics (HyRoDyn) for solving kinematics and dynamics of series-parallel hybrid robots. The main idea here is to store the analytically derived loop closure functions (LCF) in a *configurable* mechanism library which is identified by its type (for e.g. RH5 ANKLE [Kumar et al., 2018c]). Based on submechanisms defined in a hybrid robot, it can modularly compose the LCF of the overall system in an automated way and transfers them to various kinematics and dynamics algorithms. The input to HyRoDyn is the SMURF file which can be exported through a blender based Visual Editor called Phobos. The modeling approach allows for both bottom up composition and top down decomposition which is very useful

in dealing with highly complex robotic systems. For example, a complex robotic system model can be composed from the models of simpler well-tested modules and vice-versa. HyRoDyn has also been integrated in the RoCK middleware and can be used as a configurable multi-purpose component for various robot control applications. This work is published in [Kumar et al., 2018d].

- **Application in Robotic Rehabilitation:** The real world application of HyRoDyn software tool is demonstrated in the field of robotic rehabilitation where it has been used for the kinematic and dynamic control of a complex series-parallel hybrid Recupera-Reha exoskeleton. It serves as a central object in its control architecture and helps in implementation of several therapy modes for rehabilitation namely gravity compensation, teach and replay, master-slave. Further, it allows the inclusion of human arm dynamic model into control which is very useful for weak patients in early days of stroke. The system was extensively tested with and without human subjects and a preliminary clinical trial was also conducted. The results are published in [Kumar et al., 2019b].

10.3 Future Work

This thesis addresses various challenges one encounters during the development of complex series-parallel hybrid robots. However, it also opens doors to several lines of research which should be pursued in the future.

- **Model order reduction:** The rising complexity of the series parallel hybrid robots can pose a challenge to the real time performance of various kinematics and dynamics models. Future robotic systems may have few hundreds of moving bodies and may require very fast computation of robot dynamics for the purpose of optimal control. In this scenario, it is important to evaluate the effect of dominant bodies in the kinematic chain and neglect the bodies which do not contribute much to the overall dynamics of the system. In the outlook of this thesis [Kumar et al., 2019a], an important observation is made: not all parallel mechanisms contribute equally to the overall dynamics of the system and with the help of some empirical analysis it is possible to find parts of EOM which should be solved. This depends on state space of different bodies and their mass-inertia properties. In the future work, an automated approach for model order reduction in rigid body dynamics will be investigated.

- **Support for numerical resolution for loop constraints:** At the moment, HyRoDyn only allows the analytical resolution of loop closure constraints for the parallel mechanisms that are known to its database. The next obvious step is to extend the software so that it can deal with loop closure constraints numerically for arbitrary mechanisms in a modular way. Numerical resolution of loop closure constraints is well known in the literature and these algorithms will be implemented in the near future. This will enhance the generality of this software tool.

- **Floating base systems and contact constraints:** Presently, only fixed base robotic systems are supported within the software. Since, legged robotics is one of the primary application of series-parallel hybrid robots, it is of urgent importance to extend this software to support floating base systems and contact constraints. However, algorithms for dealing with contact constraints and floating base systems already exist in the state of the art [Featherstone, 2008] and these will be implemented inside HyRoDyn in the near future.

- **Variable Stiffness Mechanisms:** This thesis dealt with the use of parallel mechanisms as an abstraction of a kinematic joint in order to enhance its mechanical properties. A natural extension in this direction would be to introduce flexible elements in the parallel mechanism based modules in order to achieve variable stiffness mechanisms (VSM). A preliminary study of a novel VSM concept was studied by Christoph Stoeffler in his Masters thesis [Stoeffler et al., 2018] under my supervision. This concept involved including a non-linear spring in series with the actuator along with redundant actuation of the parallel mechanism. Using this concept, it has been shown that it is possible to achieve an independent position and stiffness control using this device [Stoeffler et al., 2018]. The future work includes building a testbed to demonstrate its practical feasibility and integrate it in a series-parallel hybrid robot. HyRoDyn also needs to be extended for this purpose in order to allow such modules in overall kinematics and dynamics computations of the robot.

- **Open source release to engage with larger community:** The development of HyRoDyn is presently limited to DFKI-RIC and it has been used successfully in different projects (for e.g. Recupera-Reha, D-RoCK, TransFIT) within the institute. It is planned to make HyRoDyn software open-source in future so that more people from the kinematics community can help in contributing to the submechanism libraries in HyRoDyn so that catalog of supported parallel mechanisms can be enriched. In this way, HyRoDyn will serve the interest of designers and control engineers from the robotics community around the world.

Part VI

Appendix

Appendix A

Lie Groups

"Lie theory is in the process of becoming the most important part of modern mathematics. Little by little it became obvious that the most unexpected theories, from arithmetic to quantum physics, came to encircle this Lie field like a gigantic axis."

– Jean Dieudonne, *French mathematician (1906-1992)*

Lie groups are abstract mathematical objects important for the study of continuous symmetry such as rotational symmetry in three dimensions. These are attributed to Norwegian mathematician Sophus Lie (1842-1899) and are widely used in different areas of mathematics and physics. His original motivation to develop this theory was to model the continuous symmetry in differential equations similar to Galois' motivation to use finite groups in order to model the discrete symmetries of algebraic equations.

In the following, the basic concepts from group and Lie group theory, are introduced which are particularly useful in understanding the mechanics formulations applied to robotics. The content presented here is inspired from [Murray et al., 1994], Wikipedia and Wolfram Mathworld.

A.1 Basics

In this section, basic definitions of a group and its related concepts like rings and fields are presented.

A.1.1 Groups

Definition 15 (Group) *A group is a set G, together with a binary operation $\circ$ (called the group action of G) that combines any two elements g_1 and g_2 from this group to* *form another element, denoted as $g_1 \circ g_2$. In order to qualify as a group, the set and the operation $(G, \circ)$ has to satisfy the following four axioms:*

1. ***Closure**: For all $g_1, g_2 \in G$, the result of the operation $g_1 \circ g_2$ must also be an element of the set G.*

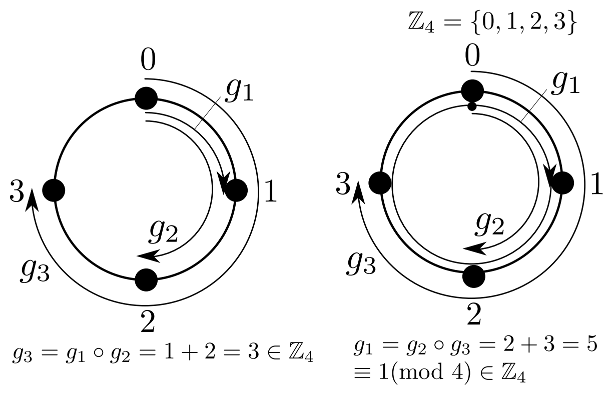
2. ***Associativity**: For all $g_1, g_2, g_3 \in G$, $(g_1 \circ g_2) \circ g_3 = g_1 \circ (g_2 \circ g_3)$.*

3. ***Identity element**: There exists an element $e \in G$ such that for every element g in the set G, the equation $e \circ g = g \circ e = g$ holds.*

4. ***Inverse**: For each g in G, there exists an element $g^{-1} \in G$, such that $g^{-1} \circ g = g \circ g^{-1} = e$, where e is the identity element.*

Commonly encountered group actions include addition $(+)$, for which the identity is a zero element, and the inverse element is subtraction; and multiplication $(\times)$, for which the identity element has a unit value and the inverse element is division. Additive groups are usually commutative in nature meaning that the left-right order of the group elements in a sequence does not affect the result of their operation i.e $g_1 + g_2 = g_2 + g_1$. Multiplicative groups may be commutative, but are often not, giving rise to the notion of *left* ($L_g : h \mapsto g \circ h$) and *right* ($R_g : h \mapsto h \circ g$) group actions. It is to be noted that group $(G, \circ)$ for which the *commutative property* $g_1 \circ g_2 = g_2 \circ g_1 \forall g_1, g_2 \in G$ holds is called an **Abelian group** (in honor of a Norwegian mathematician Niels Henrik Abel). For example, $(\mathbb{R}, +)$ is an Abelian group since addition of real numbers is closed, associative and commutative. Its identity element is zero and inverse operation is subtraction. An example of non-Abelian group is the set of all $(n \times n)$ matrices with multiplication operation, also known as General Linear group $GL(n)$, as matrix multiplication is non-commutative. Both $(\mathbb{R}, +)$ and $GL(n)$ are examples of so called **infinite groups** as they contain infinite elements. If there are a finite number of elements, the group is called a **finite group** and the number of elements is called the **group order** of the group. The set of finite number of integers n form a finite cyclic group $\mathbb{Z}_n = \{0, 1, \ldots, n-1\}$ of order n under the group operation of addition with modulo n (see Fig. A.1 for an example). Its identity element is $e = 0$ and additive inverse is $g^{-1} = n - g$. Other examples of finite groups include finite symmetric groups, permutation groups etc.

The basic concept of a group – a set with an operation for combining its elements – extends to several other mathematical structures, for example rings and fields which are very useful concepts in the abstract algebra.


$\mathbb{Z}_4 = \{0,1,2,3\}$
0
1
2
3
g_1
g_2
g_3
$g_3 = g_1 \circ g_2 = 1 + 2 = 3 \in \mathbb{Z}_4$
$g_1 = g_2 \circ g_3 = 2 + 3 = 5$
$\equiv 1 (\mathrm{mod}\ 4) \in \mathbb{Z}_4$

$+$	**0**	**1**	**2**	**3**
0	0	1	2	3
1	0	1	2	0
2	2	3	0	1
3	3	0	1	2

$e = 0$
$g^{-1} = 4 - g$

Addition Table

Figure A.1: Cyclic group $\mathbb{Z}_4$ with addition operation (modulo 4)

A.1.2 Rings

Definition 16 (Ring) *A Ring is a set R equipped with two binary operations: $(+)$ that acts like addition and $(\cdot)$ that acts like multiplication. This abstract mathematical structure is denoted as $(R, +, \cdot)$ and must satisfy the following axioms:*

1. ***Commutativity under addition**: A ring forms a commutative or Abelian group under addition i.e.*
   - $\forall a, b \in R, a + b \in R$ *(closure property)*
   - $\forall a, b, c \in R, (a + b) + c = a + (b + c)$ *(associative property)*
   - *there exists an element 0 in R such that $a + 0 = a \forall a \in R$ (identity element)*
   - *for every a in R there exists $-a$ in R such that $a + (-a) = 0$ (additive inverse)*
   - $\forall a, b \in R, a + b = b + a$ *(commutative property)*
2. ***Monoid under multiplication**: A ring forms a monoid[1] under multiplication i.e.*
   - $\forall a, b, c \in R, (a \cdot b) \cdot c = a \cdot (b \cdot c)$ *(associative property)*
   - *there exists an element 1 in R such that $a \cdot 1 = a$ and $1 \cdot a = a$ for all a in R (identity element)*
3. ***Distributivity**: The multiplicative operation must distribute over the additive operation, so that the product of a sum is equal to the sum of individual products. This means:*
   - $\forall a, b, c \in R, a \cdot (b + c) = a \cdot b + a \cdot c$ *(left distributivity)*
   - $\forall a, b, c \in R, (b + c) \cdot a = b \cdot a + c \cdot a$ *(right distributivity)*

[1]A monoid in which each element has an inverse is a group.

For example, the real numbers $\mathbb{R}$ form a ring under standard addition and multiplication operations because

- elements of $\mathbb{R}$ form the group $(\mathbb{R}, +)$ under addition,
- multiplication of real numbers produces real numbers and is associative,
- multiplication distributes over addition, and
- the real numbers do not form a group under multiplication as the inverse of 0 is not a real number which is already relaxed for the definition of rings.

Another example includes the set of $(n \times n)$ matrices under addition and multiplication operations form a matrix ring. These are examples of infinite rings. Also, $\mathbb{Z}_n$ when equipped with both addition and multiplication operations of modulo n forms a finite ring. When applied to $\mathbb{Z}_4$ introduced in Fig. A.1, it is trivial to see that even under the multiplication operation with modulo 4, the required algebraic structures are preserved e.g. $g_2 = g_1 \cdot g_2 = 1 \cdot 2$ and $g_2 = g_2 \cdot g_3 = 2 \cdot 3 = 6 \equiv 2 \pmod 4$.

A.1.3 Fields

A field is a ring for which the inverse of every non-zero element is also an element and hence it is a set where four binary operations namely addition, subtraction, multiplication and division are possible (loosely speaking).

Definition 17 (Field) *A field is a set F equipped with primarily two binary operations addition and multiplication which satisfy the following field axioms:*

Property	*addition*	*multiplication*
associativity	$(a+b)+c = a+(b+c)$	$(a \cdot b) \cdot c = a \cdot (b \cdot c)$
commutativity	$a+b = b+a$	$a \cdot b = b \cdot a$
distributivity	$a \cdot (b+c) = a \cdot b + a \cdot c$	$(a+b) \cdot c = a \cdot c + b \cdot c$
identity	$a+0 = a = 0+a$	$a \cdot 1 = a = 1 \cdot a$
inverse	$a+(-a) = 0 = (-a)+a$	$a \cdot a^{-1} = 1 = a^{-1} \cdot a$ if $a \neq 0$

Written in a compact form, a field is a set F equipped two binary operations addition and multiplication such that both $(F, +)$ and $(F, \cdot)$ are commutative groups and these operations are linked together with the distributive property.

Both the real and complex numbers form fields but the ring of square matrices do not form a field since they may contain linearly dependent rows that do not have well defined inverse. These are examples of infinite fields. Finite fields (also called as

Galois fields) are fields with finitely many elements, whose number is also referred to as the order of the field. As seen previously, simplest finite fields are accessible using modular arithmetic. For a given prime number n, $\mathbb{Z}_n$ is a finite field under addition and multiplication operations. It is easy to check that $\mathbb{Z}_4$ is not a finite field because 2 does not have a multiplicative inverse as $2 \cdot 2 = 4 = 0$ mod 4. However, $\mathbb{Z}_5$ is a finite field since $n = 5$ is a prime.

A.2 Some Group Concepts

A.2.1 Building Blocks of Groups

Definition 18 (Subgroup) *A subgroup $(H, \circ)$ is a subset of group elements of a group $(G, \circ)$ that satisfies the group axioms. "H is a subgroup of G" is written as $H \subseteq G$, or sometimes $H \leq G$.*

A proper subgroup of a group G is a subgroup H which is a proper subset of G (i. e. $H \neq G$). In this case, "H is a proper subgroup of G" is written as $H \subset G$, or sometimes $H < G$. Every group G has at least two subgroups. These are the groups containing the identity element e (also known as **trivial subgroup**) and the group itself G. These are not usually very interesting. However, other subgroups, if they exist, might be the building blocks of the group under study. If the group has *only* these two subgroups, it is called a **simple group**. These are the building blocks of the group similar to how prime numbers are building blocks of integers.

Definition 19 (Cosets) *For a subgroup H of a group G and an element g of G, define $g \circ H$ to be the set $\{g \circ h : h \in H\}$ and $H \circ g$ to be the set $\{h \circ g : h \in H\}$. A subset of G of the form $g \circ H$ for some $g \in G$ is said to be a left coset of H and a subset of the form $H \circ g$ is said to be a right coset of H. This distinction is necessary since G may not be Abelian.*

Although derived from a subgroup, cosets are not usually themselves subgroups of G, only subsets.

Definition 20 (Normal Subgroup) *A subgroup H of a group G is called a normal subgroup of G if it is invariant under conjugation; that is, the conjugation of an element of H by an element of G is always in H. It is usually denoted as $H \triangleleft G$.*

$$H \triangleleft G \Leftrightarrow \forall h \in H, g \in G \colon g \circ h \circ g^{-1} \in H \tag{A.1}$$

If a subgroup is a normal subgroup, the sets of left and right cosets of H in G coincide.

Example 7 *Let us consider the group of integers* $\{\ldots, -2, -1, 0, 1, 2, \ldots\}$ *under addition denoted as* $G = (\mathbb{Z}, +)$ *with identity element* $e = 0$ *and inverse* $g^{-1} = -g$. *Any integer multiple of this group will form a subgroup. For example,* $H = 4\mathbb{Z} = \{\ldots, -8, -4, 0, 4, 8, \ldots\}$ *qualifies for a subgroup. In the following table, some examples of (left) cosets are provided for the chosen subgroup* $4\mathbb{Z}$ *and picking some elements* $(0, 1, 2, 3)$ *from the group* $(\mathbb{Z}, +)$.

Remainder	*Coset*	*Elements*
0	$0 + 4\mathbb{Z}$	$\{\ldots, -8, -4, 0, 4, 8, \ldots\}$
1	$1 + 4\mathbb{Z}$	$\{\ldots, -7, -3, 1, 5, 9, \ldots\}$
2	$2 + 4\mathbb{Z}$	$\{\ldots, -6, -2, 2, 6, 10, \ldots\}$
3	$3 + 4\mathbb{Z}$	$\{\ldots, -5, -1, 3, 7, 11, \ldots\}$

It is easy to see that our chosen subgroup $H = (4\mathbb{Z}, +)$ *is also a normal subgroup as* $\forall h \in H, g \in G\colon g + h + (-g) \in H$. *Also, the chosen cosets form a group which is called as* ***quotient group*** *written as* $\mathbb{Z}/4\mathbb{Z}$.

Overall, it is possible to show that when $H \subseteq G$, and its cosets form a group, then $g^{-1} \circ H \circ g = H \forall g \in G$. The converse of this statement is also true. The proof is out of the context of the present text. The coset group is called **factor group** in a general context.

A.2.2 Relationship between Groups

Definition 21 (Group Homomorphism) *Group homomorphism is a structure preserving relationship between two groups. A function* $f : G \mapsto H$ *between two groups* $(G, \cdot)$ *and* $(H, *)$ *is called a homomorphism if the equation*

$$f(g \cdot k) = f(g) * f(k) \tag{A.2}$$

holds for all elements g, k *in the function domain* G.

Example 8 *Consider the two groups: additive group of real numbers* $(\mathbb{R}, +)$ *and the multiplicative group of positive real numbers* $(\mathbb{R}^{+}, \times)$. *Let us define exponential function as* $f : \mathbb{R} \mapsto \mathbb{R}^{+}$ *such that* $f(x) = \exp(x)$. *One could easily check that for any* $x, y \in \mathbb{R}$, *the following holds true.*

$$f(x + y) = \exp(x + y) = \exp(x) \cdot \exp(y) = f(x) \cdot f(y)$$

Hence, the exponential function is a group homomorphism.

Definition 22 (Group Isomorphism) *Two groups $(G, \cdot)$ and $(H, *)$ are called isomorphic if there exists group homomorphisms $f_1 : G \mapsto H$ and $f_2 : H \mapsto G$ such that applying the two functions one after the another in each of the two possible orders gives identity functions of G and H. Consider any pair of elements $(g_1, g_2) \in G$ and $(h_1, h_2) \in H$ and require that $f_2 = f_1^{-1}$ is bijective, then the following holds true.*

$$f_1(f_2(h_1 * h_2)) = f_1(f_2(h_1) \cdot f_2(h_2)) = f_1(f_2(h_1)) * f_1(f_2(h_2)) = h_1 * h_2 \tag{A.3}$$

$$f_2(f_1(g_1 \cdot g_2)) = f_2(f_1(g_1) * f_1(g_2)) = f_2(f_1(g_1)) \cdot f_2(f_1(g_2)) = g_1 \cdot g_2 \tag{A.4}$$

In short, when a group homomorphism is also a bijection (i.e. one-to-one and onto), it is called group isomorphism.

Example 8 (Continued) *Let us define natural logarithm function as $f' : \mathbb{R}^+ \mapsto \mathbb{R}$ such that $f'(x) = \log(x')$.* [2] *One could easily check that for any $(x', y') \in \mathbb{R}^+$, the following holds true.*

$$f'(x' \cdot y') = \log(x' \cdot y') = \log(x') + \log(y') = f'(x') + f'(y') \tag{A.5}$$

Hence, logarithm natural function is also a group homomorphism. Since, natural logarithm and exponential functions are bijective i.e. $f' = f^{-1}$, it is easy to verify that for any $(x, y) \in \mathbb{R}$ and $(x', y') \in \mathbb{R}^+$

$$\exp(\log(x' \cdot y')) = \exp(\log(x') + \log(y')) = \exp(\log(x')) \cdot \exp(\log(y')) = x' \cdot y'$$
$$\log(\exp(x + y)) = \log(\exp(x) \cdot \exp(y)) = \log(\exp(x)) + \log(\exp(y)) = x + y$$

holds true.

Definition 23 (Group Automorphism) *A group automorphism $f : G \mapsto G$ is a group isomorphism from a group G to itself. The group of all automorphisms of G is denoted as $\mathrm{Aut}(G)$.*

Let $(G, \circ)$ be a group and g is an element of it. The map $i_g : G \mapsto G$ given by $i_g(x) = g \circ x \circ g^{-1}$ is an automorphism of G. It is called **conjugation** by g, or the **inner automorphism** corresponding to g. An **outer automorphism** of G is an automorphism which cannot be expressed in this form for $g \in G$, but can be so expressed if g belongs to a larger group containing G.

A.2.3 Group Operations

Groups can be combined together to form larger groups and hence it is important to know how their underlying sets combine, and nature of overall group action. In this

[2]Prime notation is used to distinguish between two groups. It should not be confused with derivative.

context, direct and semi-direct product of the groups are introduced.

Definition 24 (Direct Product) *Consider two groups $(G, \cdot)$ and $(H, *)$, the direct product $K = G \times H$ is defined as follows:*

1. *The underlying set is the Cartesian product, $K = G \times H$ and contains all the ordered pairs (g, h), where $g \in G$ and $h \in H$.*

2. *The binary operation $\circ$ in the resulting group $G \times H$ is defined component-wise:*

$$(g_1, h_1) \circ (g_2, h_2) = (g_1 \cdot g_2, h_1 * h_2) \tag{A.6}$$

The resulting algebraic object satisfies all the group axioms like associativity, existence of an identity element i.e. (e_G, e_H) and inverse i.e. (g^{-1}, h^{-1}). Another implication of the above definition is that both groups $(G, \cdot)$ and $(H, *)$ are normal subgroups of the group $(K, \circ)$ resulting from their direct product.

Example 9 *Let $(\mathbb{R}, +)$ be the group of real numbers under addition. Then the direct product $\mathbb{R} \times \mathbb{R} = \mathbb{R}^2$ is the group of all two-component vectors (x, y) under the operation of vector addition:*

$$(x_1, y_1) + (x_2, y_2) = (x_1 + x_2, y_1 + y_2)$$

If both $(G, \cdot)$ and $(H, *)$ are finite groups, then the order of the direct product group $(G \times H)$ is product of cardinalities of the two groups i.e. $|G| \cdot |H|$. Further, it should be noted that if either of the two groups is non-abelian, then the direct product group is also non-abelian.

In group theory, a semidirect product is a generalization of the direct product which expresses a group as a product of subgroups. One could think about this construction from two viewpoints: intrinsic and extrinsic. The intrinsic viewpoint describes the semidirect product G as the product between two subgroups N and H, one of which should be a normal subgroup. This concept is called inner semidirect product denoted as $G = N \ltimes H$ and is helpful in studying group's behavior and classify its isomorphisms. Another way to think of semidirect products is extrinsic: given two abstract groups N and H with some specified relationship between them, given by a certain homomorphism ϕ, one can construct a new group called the semidirect product (or outer semidirect product) $G = N \ltimes_\phi H$. This allows one to build new, larger groups from smaller ones, with a construction that is more general and richer than a direct product.

Definition 25 (Inner Semidirect Product) *Let $(G, \circ)$ be a group and let $N \triangleleft G$ be the normal subgroup and H be any subgroup of G. Then the following statements are equivalent:*

1. *$G = NH = \{n \circ h \mid n \in N, h \in H\}$ is the product of subgroups*[3] *where the subgroups have a trivial intersection i.e. $N \cap H = \{e\}$.*
2. *Every $g \in G$ can be written uniquely as $g = n \circ h$ and $g = h \circ n$ with $n \in N, h \in H$.*
3. *Let us define $\psi : H \mapsto G/N$ as a function that maps from the subgroup H to the factor group G/N as $\psi(h) = hN$. Then ψ is an isomorphism.*

If any of these conditions hold, then $G = N \ltimes H$ expresses G as an inner semidirect product of N and H. In other words, one could also say that G splits over N.

Definition 26 (Outer Semidirect Product) *Let H and N be unrelated groups, and suppose $\phi : H \mapsto \mathrm{Aut}(N)$ is a homomorphism, which sends elements $h \in H$ to automorphisms ϕ_h of N. Then the group $G = N \ltimes_\phi H$ is defined as the set of ordered pairs (n, h) with $n \in N, h \in H$ and group operation given by the formula:*

$$(n, h) \cdot (n_1, h_1) = (n\phi_h(n_1), hh_1) \tag{A.7}$$

This defines a group in which the identity element is (e_N, e_H) and the inverse of the element (n, h) is $(\phi_{h^{-1}}(n^{-1}), h^{-1})$.

The concept of groups can be further elaborated when it is endowed with additional structures for example topological space, differential manifold or of an algebraic variety. In the next section, the concept of Lie groups is presented.

A.3 Lie Groups

Definition 27 (Lie Group) *A Lie group is a group $(G, \circ)$ which is equipped with an additional structure of a smooth differential manifold and for which the group operation $\circ : (g, h) \mapsto g \circ h$ and inversion $g \mapsto g^{-1}$ are smooth maps. A Lie group is Abelian if $g \circ h = h \circ g$ for all $g, h \in G$.*

Since, Lie groups are not necessarily commutative, the notions of *left* and *right* actions (or translations), as introduced in Section A.1.1, are important. Moreover, it can be easily verified that left and right actions commute when used as composite functions: $L_g(R_h) = R_h(L_g)$ [4].

A.3.1 Translational Group

Definition 28 (Translational Group) *The translational group $T(n)$ is defined by choosing the Euclidean space $\mathbb{R}^n$ as the underlying set and addition $(\boldsymbol{x}, \boldsymbol{y}) \mapsto \boldsymbol{x} + \boldsymbol{y}$ as the group operation.*

[3]This is different from direct product.

[4]It is equivalent to $L_g \circ R_h = R_h \circ L_g$ when $\circ$ is used as a notation for composite functions.

The translational group is an Abelian group with identity element as zero vector $\mathbf{0} \in \mathbb{R}^n$, also known as origin, and the inverse of any element $\boldsymbol{x} \in \mathbb{R}^n$ is $-\boldsymbol{x}$.

A.3.2 General Linear Group

Definition 29 (General Linear Group) *The general linear group of order n is the set of $n \times n$ invertible matrices, together with the operation of ordinary matrix multiplication. It is denoted as $GL(n, \mathbb{F})$ where $\mathbb{F}$ is the underlying set.*

Its identity element is the $n \times n$ identity matrix and inversion is given by matrix inverse. Since, both matrix multiplication and inverse are smooth operations in matrix components, it is a matrix Lie group. As a manifold, it can be regarded as an open subset in $\mathbb{F}^{n^2}$. An example of general linear group is the group of all $n \times n$ real matrices denoted as $GL(n, \mathbb{R})$ with group operation $(\boldsymbol{A}, \boldsymbol{B}) \mapsto \boldsymbol{A} \cdot \boldsymbol{B} \forall \boldsymbol{A}, \boldsymbol{B} \in \mathbb{R}^{n \times n}$ and the inversion is given by $\boldsymbol{A}^{-1}$ with $\det(\boldsymbol{A}) \neq 0$. In the following, some important subgroups of $GL(n, \mathbb{F})$ are discussed.

A.3.2.1 Orthogonal Group

Definition 30 (Orthogonal Group) *The orthogonal group of order n is the set of $n \times n$ invertible matrices, together with the operation of ordinary matrix multiplication such that $\boldsymbol{R}\boldsymbol{R}^T = \boldsymbol{R}^T\boldsymbol{R} = \boldsymbol{I}$. It is denoted as $O(n, \mathbb{F})$ where $\mathbb{F}$ is the underlying set.*

Its identity element is the $n \times n$ identity matrix and inversion is given by matrix transpose. Very frequently encountered are the real orthogonal matrices $O(n, \mathbb{R}) \subset GL(n, \mathbb{R})$, simply denoted as $O(n)$. An element in $O(n)$ can be parameterized by $n(n-1)/2$ unique parameters. Out of n^2 parameters of a general real matrix, the remaining $n(n+1)/2$ parameters are determined by the condition $\boldsymbol{R}\boldsymbol{R}^T = \boldsymbol{I}$. The determinant of any orthogonal matrix is either 1 or -1.

A.3.2.2 Special Orthogonal Group

Definition 31 (Special Orthogonal Group) *The orthogonal $n \times n$ matrices with determinant 1 form a normal subgroup of $O(n, \mathbb{F})$ known as the special orthogonal group $SO(n, \mathbb{F})$.*

$$SO(n) = \{\boldsymbol{R} \in GL(n, \mathbb{F}) : \boldsymbol{R}\boldsymbol{R}^T = \boldsymbol{I}, \det \boldsymbol{R} = +1\} \tag{A.8}$$

Example 10 *The 2-dimensional special orthogonal group $SO(2)$ is useful for describing rotations with $2(2-1)/2 = 1$ degrees of freedom (for e.g. motion produced by a revolute joint). Similarly, $SO(3)$ is useful for describing the motion of a spherical joint i.e. rotations with $3(3-1)/2 = 3$ degrees of freedom.*

A.3.2.3 Special Linear Group

Definition 32 (Special Linear Group) *The general $n \times n$ matrices with determinant 1 form a subgroup of $GL(n, \mathbb{F})$ known as the special linear group $SL(n, \mathbb{F})$.*

$$SL(n) = \{\boldsymbol{A} \in GL(n, \mathbb{F}) : \det \boldsymbol{A} = +1\} \tag{A.9}$$

A.3.3 Euclidean Group

An Euclidean group $E(n)$ is the group of Euclidean isometries of an Euclidean space $\mathbb{R}^n$ i.e. the transformations of that space that preserve the Euclidean distance between any two points.

Definition 33 (Euclidean group) *Let us define an element of the Euclidean group $E(n)$ to be a pair $(\boldsymbol{R}, \boldsymbol{p})$, where $\boldsymbol{R}$ is an element of the orthogonal group $O(n)$ and $\boldsymbol{p}$ is an element in translational group $\mathbb{R}^n$. Any element $(\boldsymbol{R}, \boldsymbol{p})$ of $E(n)$ gives a transformation of n-dimensional Euclidean space $f_{(\boldsymbol{R},\boldsymbol{p})} : \mathbb{R}^n \mapsto \mathbb{R}^n$ and can be interpreted as either a translation followed by an orthogonal transformation $\boldsymbol{x} \mapsto \boldsymbol{R}(\boldsymbol{x} + \boldsymbol{p})$ or the same orthogonal transformation followed by a translation: $\boldsymbol{x} \mapsto \boldsymbol{R}\boldsymbol{x} + \boldsymbol{c}$ with $\boldsymbol{c} = \boldsymbol{R}\boldsymbol{p}$. Hence, the group operation can also be thought of as a set of mappings.*

The identity element of the Euclidean group $E(n)$ is the pair $(\boldsymbol{I}_{n\times n}, \boldsymbol{0})$ where $\boldsymbol{I}_{n\times n}$ is the identity element of $O(n)$ and $\boldsymbol{0} \in \mathbb{R}^n$ is the identity element (or origin) in translational group $T(n)$. The inverse of an element $(\boldsymbol{R}, \boldsymbol{p}) \in E(n)$ is given by $(\boldsymbol{R}^{-1}, -\boldsymbol{R}^{-1}\boldsymbol{p})$.

Dimensionality or Degrees of Freedom The dimension of $E(n)$ is $n(n+1)/2$. Out of these, n can be attributed to available translational symmetry, and the remaining $n(n-1)/2$ to rotational symmetry.

Group Structure It can be checked that for any translation $t \in T(n)$ and isometry $x \in E(n)$, the operation $xtx^{-1} \in T(n)$ and hence, $T(n)$ is the normal subgroup of $E(n)$, also written as $T(n) \triangleleft E(n)$. This implies that $E(n)$ is the semidirect product of $O(n)$ and $T(n)$ written as $E(n) = O(n) \ltimes T(n)$. In other words, $O(n)$ is the quotient or factor group of $E(n)$, expressed as $O(n) \cong E(n)/T(n)$. Another simple way to check whether $E(n)$ is the semidirect product of $O(n)$ and $T(n)$ or not is the fact that inverse element is given by $(\boldsymbol{R}^{-1}, -\boldsymbol{R}^{-1}\boldsymbol{p})$ and not $(\boldsymbol{R}^{-1}, -\boldsymbol{p})$ which would have been the case if it was a direct product.

Direct and Indirect Isometries The direct isometries (i.e., isometries preserving the handedness of chiral subsets) comprise a subgroup of $E(n)$, called the **special Euclidean group** $SE(n)$ with $\det(\boldsymbol{R}) = +1$ also sometimes denoted as $E^+(n)$. The isometries that reverse the handedness are called indirect isometries, denoted as $E^{-}(n)$. These isometries in $E(n)$ are topologically disconnected i.e. there is no continous trajectory that starts in $E^{+}(n)$ and ends in $E^{-}(n)$ or vice-versa.

Matrix representation Euclidean group $E(n)$ can be identified with $(n+1)\times(n+1)$ matrices of the form

$$x = \begin{bmatrix} \boldsymbol{R}_{n\times n} & \boldsymbol{p}_{n\times 1} \\ \boldsymbol{0}_{1\times n} & 1 \end{bmatrix}_{(n+1)\times(n+1)} \tag{A.10}$$

where $\boldsymbol{R} \in O(n)$ and $\boldsymbol{p} \in \mathbb{R}^n$. Hence, $E(n)$ can also be regarded as a subgroup of $GL(n+1, \mathbb{R})$. Similarly, the special Euclidean group $SE(n)$ can be defined as the following:

$$SE(n) = \left\{ \begin{bmatrix} \boldsymbol{R} & \boldsymbol{p} \\ \boldsymbol{0} & 1 \end{bmatrix} \mid \boldsymbol{R} \in SO(n), \boldsymbol{p} \in \mathbb{R}^n \right\} \tag{A.11}$$

Example 11 *The group of motions possible in $SE(2)$ has a total of 3 DOF with 2 translational DOF and 1 rotational DOF. $SE(2)$ can also be elaborated as $SO(2) \ltimes \mathbb{R}^2$ and represents all the possible motions in a plane. Similarly, $SE(3)$ is a Lie group of 6 dimensions out of which 3 are rotational and remaining 3 are translational. $SE(3)$ can also be elaborated as $SO(3) \ltimes \mathbb{R}^3$ and represents all the possible motions in 3D space.*

Appendix B

Differential Geometry

"It is well known that geometry presupposes not only the concept of space but also the first fundamental notions for constructions in space as given in advance. It only gives nominal definitions for them, while the essential means of determining them appear in the form of axioms. The relationship of these presumptions is left in the dark; one sees neither whether and in how far their connection is necessary, nor a priori whether it is possible. From Euclid to Legendre, to name the most renowned of modern writers on geometry, this darkness has been lifted neither by the mathematicians nor the philosophers who have laboured upon it."

– Bernhard Riemann, *German mathematician (1826-1866)*

Differential geometry is a mathematical discipline that uses the techniques of differential and integral calculus, linear and multi-linear algebra to study problems in geometry. Its applications are pervasive in a wide range of topics such as general theory of relativity, mechanics, control theory, information geometry etc. In the last two decades, ideas from differential geometry have played a crucial role in establishing the scientific foundations of robotics. While the most obvious applications of differential geometry in robotics are robot mechanics and planning, it is also now being used to solve problems in control theory, computer vision, machine learning etc. Carl Friedrich Gauss (1777-1855) is considered as the father of differential geometry. He was a cartographer and many terms in modern differential geometry (chart, atlas, map, coordinate system, geodesic, etc.) reflect these origins. One of his most important contributions was Theorema Egregium which explained why the Earth cannot be displayed on a map without distortion. The definitions presented here are based on [Nakahara, 2003] and [Boothby, 2003].

B.1 Basics

The central object in modern differential geometry is the differential manifold. In this section, basic concepts necessary in order to define a manifold are introduced.

B.1.1 Vector Space

Definition 34 (Vector Space) *A vector space (or a linear space) V over a field K is a set in which two operations, addition and multiplication by an element of K (called a scalar), are defined. The elements of V (called vectors) should satisfy the following axioms:*

1. *Associativity of addition:* $(\boldsymbol{u}+\boldsymbol{v})+\boldsymbol{w}=\boldsymbol{u}+(\boldsymbol{v}+\boldsymbol{w})$
2. *Commutativity:* $\boldsymbol{u}+\boldsymbol{v}=\boldsymbol{v}+\boldsymbol{u}$
3. *Identity element of addition:* $\boldsymbol{v}+\mathbf{0}=\boldsymbol{v}\ ,\boldsymbol{v}\in V$
4. *Inverse element of addition: For any $\boldsymbol{v}\in V$, there exists $-\boldsymbol{v}$ such that* $\boldsymbol{v}+(-\boldsymbol{v})=\mathbf{0}$.
5. *Compatibility of scalar multiplication with field multiplication:* $a(b\boldsymbol{v})=(ab)\boldsymbol{v}$
6. *Identity element of scalar multiplication:* $1\boldsymbol{v}=\boldsymbol{v}$*, where* 1 *denotes the multiplicative identity in* K.
7. *Distributivity of scalar multiplication with respect to vector addition:* $a(\boldsymbol{u}+\boldsymbol{v})=a\boldsymbol{u}+a\boldsymbol{v}$
8. *Distributivity of scalar multiplication with respect to field addition:* $(a+b)\boldsymbol{v}=a\boldsymbol{v}+b\boldsymbol{v}$

Here, $\boldsymbol{u},\boldsymbol{v},\boldsymbol{w}\in V$ *and the scalars* $a,b\in K$.

Vector space has the properties of a commutative group under addition $+:V\times V\mapsto V$ but it has an additional structure of scalar multiplication $\cdot:K\times V\mapsto V$ which makes it different from ordinary groups. The simplest example of a vector space over a field K is the field itself, equipped with its standard addition and multiplication. More generally n-tuples $(a_1,a_2,\dots,a_n)$ of elements of the field K forms a vector space and is usually called a coordinate space (denoted as K^n). For example, $\mathbb{R}^3$ is a vector space and can be used to describe the translation of a rigid body in 3D space. However, the idea of a vector space is much more abstract. For example, real polynomials of degree less than or equal to n forms a vector space or the set of functions that are continuous on $\mathbb{R}$ also form a vector space.

Definition 35 (Basis) *A set of linearly independent vectors $\{\boldsymbol{e}_i\}$ is called a basis of V, if any element $\boldsymbol{v} \in V$ is written uniquely as a linear combination of $\{\boldsymbol{e}_i\}$:*

$$\boldsymbol{v} = v_1\boldsymbol{e}_1 + v_2\boldsymbol{e}_2 + \ldots + v_n\boldsymbol{e}_n \tag{B.1}$$

where $v_i \in K$ are the components of V with respect to the basis $\{\boldsymbol{e}_i\}$.

Definition 36 (Dual Vector Space) *The dual V^* of a vector space V is a vector space having the same dimension as V and has the property that a scalar product is defined between V^* and V. If $\boldsymbol{u} \in V$ and $\boldsymbol{v} \in V^*$ then the scalar product is written as $\boldsymbol{u} \cdot \boldsymbol{v} = \sum_i^{\dim(V)} u_i v_i$ or alternatively as $\boldsymbol{v} \cdot \boldsymbol{u}$ (due to commutative property of scalar product).*

Definition 37 (Dual Basis) *Let $E = \{\boldsymbol{e}_1, \boldsymbol{e}_2, \ldots, \boldsymbol{e}_n\}$ be a basis on V and $E^* = \{\boldsymbol{e}_1^*, \boldsymbol{e}_2^*, \ldots, \boldsymbol{e}_n^*\}$ be a basis on its dual vector space V^*. These bases qualify for a dual basis if:*

$$\boldsymbol{e}_i^* \cdot \boldsymbol{e}_i = \delta_{ij} = \begin{cases} 1 & \textit{if} \quad i = j \\ 0 & \textit{otherwise} \end{cases} \tag{B.2}$$

where δ_{ij} is the Kronecker delta function. If $\boldsymbol{E}$ and $\boldsymbol{E}^*$ were matrices containing $\mathbf{e}_i$ and $\mathbf{e}_i^*$ as their column entries respectively, then one could write $\boldsymbol{E}^{*T}\boldsymbol{E} = \boldsymbol{I}$ or $\boldsymbol{E}^* = \boldsymbol{E}^{-T}$.

Example 12 *The standard basis vectors of the Cartesian plane $\mathbb{R}^2$ are $\{\boldsymbol{e}_1, \boldsymbol{e}_2\} = \{[1 \quad 0]^T, [0 \quad 1]^T\}$. The standard basis vectors of its dual space $\mathbb{R}^{2*}$ are $\{\boldsymbol{e}_1^*, \boldsymbol{e}_2^*\} = \{[0 \quad 1]^T, [1 \quad 0]^T\}$.*

In the context of differential geometry, the tangent plane to a surface at a point is a vector space whose origin is identified with the point of contact (also known as tangent space). Its dual space is called a cotangent space.

B.1.2 Topological Space

Definition 38 (Topological Space) *Let X be a non-empty set. A topology T on X is a collection of open subsets of X satisfying the following axioms:*

1. *The empty set $\emptyset$ and X itself belong to T i.e. $\emptyset, X \in T$.*
2. *Any arbitrary (finite or infinite) union of members of T still belongs to the open set T.*
3. *The intersection of any **finite** number of members of T still belongs to the open set T.*

A topological space is thus an ordered pair (X, T).

If X is a set and T is the collection of *all* the subsets of X , then axioms $(1)-(3)$ are automatically satisfied. This topology is called the **discrete topology**. Let X be a set and $T = \{\emptyset, X\}$. Clearly, T satisfies axioms $(1)-(3)$. This topology is called the **trivial topology**. Let X be the real line $\mathbb{R}$. All open intervals (a, b) and their unions define a topology called the **usual topology**; a and b may be $-\infty$ and $+\infty$. respectively.

Example 13 *Let $X = \{a, b, c\}$ be a set and $T = \{\emptyset, \{a\}, \{a, b\}, X\}$ be the collection of subsets of X inducing a topology. One could easily verify that this choice satisfies all the above axioms and thus, the pair (X, T) is a topological space. However, if we choose $T = \{\emptyset, \{a, b\}, \{a, c\}, X\}$, then axioms 1 and 2 are satisfied but axiom 3 fails for e.g. $\{a, b\} \cap \{a, c\} = \{a\} \not\subset T$ and hence the pair (X, T) is not a topological space in this case.*

Definition 39 (Metric Space) *A metric $d : X \times X \mapsto \mathbb{R}$ is a function which satisfies the following conditions:*

1. $d(x, y) = d(y, x)$
2. $d(x, y) \geq 0$ *where the equality holds if and only if* $x = y$.
3. $d(x, y) + d(y, z) = d(x, z)$

for any $x, y, z \in X$. If X is endowed with a metric d, X becomes a topological space whose open sets are given by "open discs",

$$U_\epsilon(X) = \{y \in X \mid d(x, y) < \epsilon\} \tag{B.3}$$

*and all their possible unions. The topology T thus defined is called the metric topology determined by d. The topological space (X, T) is called a **metric space**.*

The usual topology on $\mathbb{R}^n$ includes open balls i.e. $B_\epsilon(\boldsymbol{x}) = \{\boldsymbol{y} \in \mathbb{R}^n \mid \sum_i^n (x_i - y_i)^2 < \epsilon^2\} \subseteq \mathbb{R}^n$ as its open sets for any point $\boldsymbol{x} \in \mathbb{R}^n$.

Definition 40 (Neighborhood) *Suppose T gives a topology to X. N is a neighborhood of a point $x \in X$ if N is a subset of X and N contains some (at least one) open set U_i to which x belongs.*

The subset N need not be an open set. If N happens to be an open set in T, it is called an **open neighborhood**.

Definition 41 (Homeomorphism) *Let X_1 and X_2 be topological spaces. A map $f : X_1 \mapsto X_2$ is a homeomorphism if it is continuous and has an inverse $f^{-1} : X_2 \mapsto X_1$ which is also continuous. If there exists a homeomorphism between X_1 and X_2 , X_1 is said to be homeomorphic to X_2 and vice versa.*

In other words, X_1 is homeomorphic to X_2 if there exist maps $f : X_1 \mapsto X_2$ and $g : X_2 \mapsto X_1$ such that $f \circ g = \mathrm{id}_{X_2}$, and $g \circ f = \mathrm{id}_{X_1}$. Intuitively speaking, we suppose the topological spaces are made out of ideal rubber which we can deform at our will. Two topological spaces are homeomorphic to each other if we can deform one into the other continuously, that is, without tearing them apart or pasting. For example, a coffee cup is homeomorphic to a dough-nut.

B.1.3 Manifold

Manifolds are generalizations of our familiar ideas about curves and surfaces to arbitrary dimensional objects. A manifold is a topological space that *locally* resembles Euclidean space near each point i.e. each point on a manifold with dimension n has a neighborhood that is mapped to the Euclidean space of the same dimension. The examples for one dimensional manifolds include lines and circles and for two dimensional manifolds (also called as surfaces) include plane, sphere, torus etc.

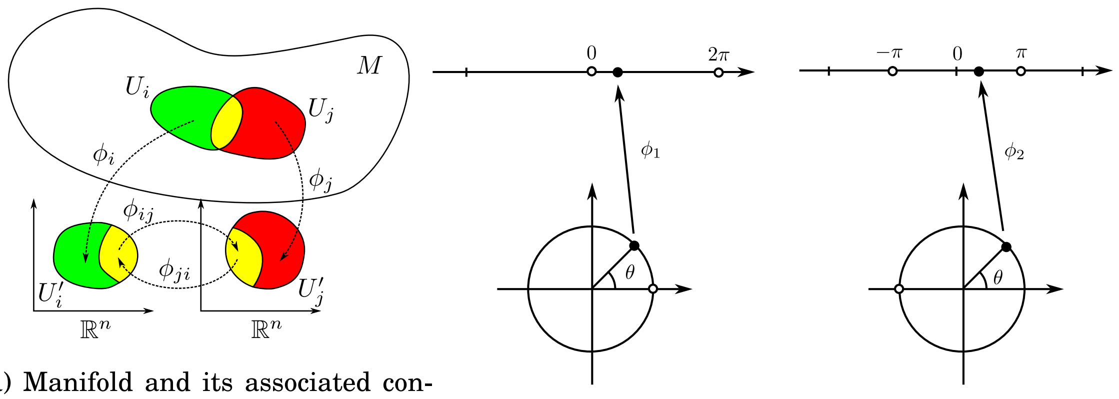
Definition 42 (Manifold) *A manifold M is a n-dimensional metric space such that $\forall x \in M$ in an open neighborhood $U \in M$ around x and it is homeomorphic to n-dimensional Euclidean space $\mathbb{R}^n$.*

An intuitive example to understand the idea of a manifold is that we are living on the Earth whose surface is S^2 , which does not look like $\mathbb{R}^2$ globally. However, it looks like an open subset of $\mathbb{R}^2$ locally. Fig. B.1a shows a manifold and illustrates various associated concepts. The **coordinate neighborhood** U_i is an open set that cover M for any index i and the **coordinate function** $\phi_i : U_i \mapsto \mathbb{R}^n$ (or simply **coordinate**) is a homeomorphism of U_i onto an open subset U_i' of n-dimensional real space $\mathbb{R}^n$. The pair (U_i, ϕ_i) is called a **chart** while the whole family of charts $\{(U_i, \phi_i)\}$ is called an **atlas**. The **transition map** of the atlas $\phi_{ij} : \phi_i(U_i \cap U_j) \mapsto \phi_j(U_i \cap U_j)$ is the function defined by $\phi_{ij} = \phi_j \circ \phi_i^{-1}$. Note that since ϕ_i and ϕ_j are both homeomorphisms, the transition map ϕ_{ij} is also a homeomorphism.

A **differentiable manifold** is a topological manifold with a globally defined differential structure. Any topological manifold can be given a differential structure locally by using the homeomorphisms in its atlas and the standard differential structure on a linear space. To induce a global differential structure on the local coordinate systems induced by the homeomorphisms, their transition maps in the atlas must be differentiable functions on the corresponding linear space.

Example 14 *Let us work out an atlas of the unit circle S^1 given by $x^2 + y^2 = 1$ in the XY-plane as shown in Fig. B.1b. We need at least two charts. Define $\phi_1^{-1} : (0, 2\pi) \to S^1$ by*

$$\phi_1^{-1} : \theta \mapsto (\cos\theta, \sin\theta) \qquad \text{(B.4)}$$



(a) Manifold and its associated concepts

(b) Charts for a unit circle S^1

Figure B.1: Manifold: associated concepts and an example

whose image is $S^1 - \{(1,0)\}$. Define also $\phi_2^{-1} : (-\pi, \pi) \to S^1$ by

$$\phi_2^{-1} : \theta \mapsto (\cos\theta, \sin\theta) \tag{B.5}$$

whose image is $S^1 - \{(-1,0)\}$. Clearly ϕ_1^{-1} and ϕ_2^{-1} are invertible and all the maps ϕ_1, ϕ_2, ϕ_1^{-1} and ϕ_2^{-1} are continuous. Thus, ϕ_1 and ϕ_2 are homeomorphisms. It is also easy to verify that the maps $\phi_{12} = \phi_2 \circ \phi_1^{-1}$ and $\phi_{21} = \phi_1 \circ \phi_2^{-1}$ are smooth.

Definition 43 (Product Manifold) *Let M be an m-dimensional manifold with an atlas $\{(U_i, \phi_i)\}$ and N be an n-dimensional manifold with $\{(V_j, \psi_j)\}$. A product manifold $M \times N$ is an $(m+n)$-dimensional manifold whose atlas is $\{(U_i \times V_j), (\phi_i, \psi_j)\}$. A point in $M \times N$ is written as $(p, q), p \in M, q \in N$, and the coordinate function (ϕ_i, ψ_j) acts on (p, q) to yield $(\phi_i(p), \psi_j(q)) \in \mathbb{R}^{m+n}$.*

Example 15 *The torus T^2 is a product manifold of two circles, $T^2 = S^1 \times S^1$. If we denote the polar angle of each circle as $\theta_i \mod 2\pi (i = 1, 2)$, the coordinates of T^2 are (θ_1, θ_2). Since each S^1 is embedded in $\mathbb{R}^2$, T^2 may be embedded in $\mathbb{R}^4$. We often imagine T^2 as the surface of a dough-nut in $\mathbb{R}^3$, in which case, however, we inevitably have to introduce bending of the surface. This is an extrinsic feature brought about by the 'embedding'. When we say 'a torus is a flat manifold', we refer to the flat surface embedded in $\mathbb{R}^4$.*

B.2 Calculus on Manifolds

The significance of differentiable manifolds resides in the fact that we may use the usual calculus developed in $\mathbb{R}^n$. Smoothness of the coordinate transformations ensures that the calculus is independent of the coordinates chosen.

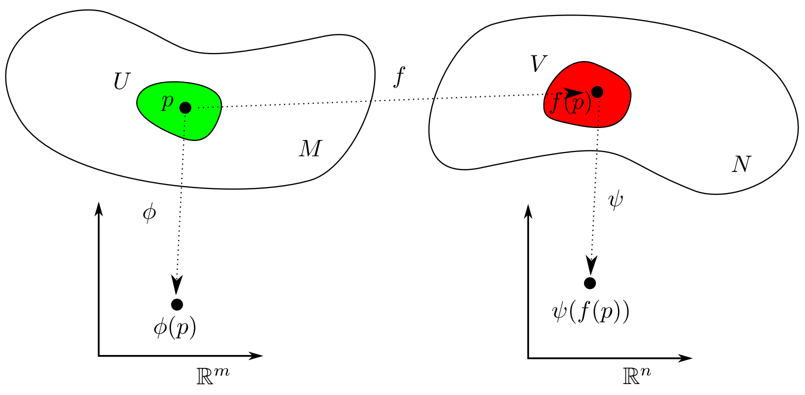


Figure B.2: Differentiable Map

B.2.1 Differentiable Maps

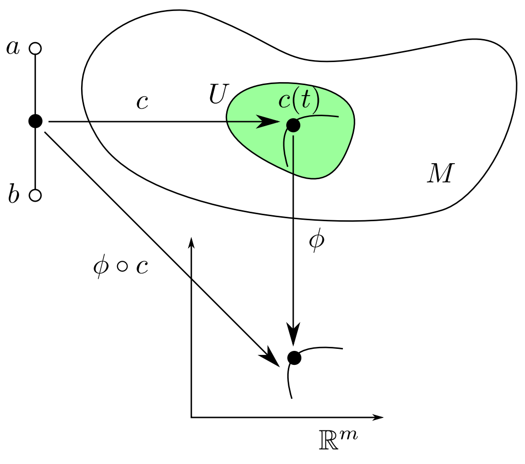
Definition 44 (Differentiable Map) *Let $f : M \mapsto N$ be a map from an m-dimensional manifold M to an n-dimensional manifold N i.e. a point $p \in M$ is mapped to a point $f(p) \in N$, via $f : p \mapsto f(p)$ (see Fig. B.2). Further, let (U, ϕ) and (V, ψ) be coordinate charts on M and N respectively, where $p \in U$ and $f(p) \in V$. The mapping $f : M \mapsto N$ is called a differentiable map if the map $\tilde{f} = \psi \circ f \circ \phi^{-1} : \mathbb{R}^m \mapsto \mathbb{R}^n$ is differentiable for all choices of coordinate charts on M and N.*

Differentiable maps are also said to be **smooth**. By definition, the differentiability of f is independent of the choice of coordinate system.

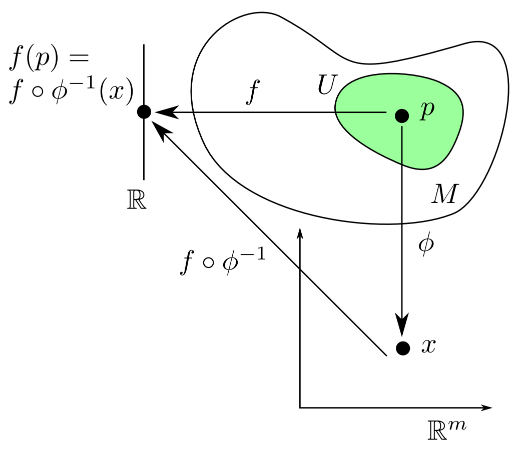
Definition 45 (Diffeomorphism) *Let $f : M \mapsto N$ be a homeomorphism and ϕ and ψ be coordinate functions as previously defined. If $\psi \circ f \circ \phi^{-1}$ is invertible (that is, there exists a map $\phi \circ f^{-1} \circ \psi^{-1}$) and both $\psi \circ f \circ \phi^{-1}$ and $\phi \circ f^{-1} \circ \psi^{-1}$ are infinitely differentiable (also denoted as C^∞), f is called a diffeomorphism and M is said to be diffeomorphic to N and vice versa, denoted by $M \equiv N$.*

It was noted in Section B.1.2 that homeomorphisms classify topological spaces according to whether it is possible to deform one space into another *continuously*. Diffeomorphisms classify spaces according to whether it is possible to deform one space to another *smoothly*. Two diffeomorphic spaces are regarded as the same manifold. Clearly a diffeomorphism is a homeomorphism.

It is now important to introduce two special classes of mappings namely, curves and functions. An open curve in an m-dimensional manifold M is a map $c : (a, b) \mapsto M$ where (a, b) is an open interval such that $a < 0 < b$. If a curve is closed, it is regarded as a map $c : S^1 \mapsto M$. In both cases, the **curve** c is locally a map from an open interval to M. On a chart (U, ϕ), a curve $c(t)$ has the coordinate representation $x = \phi \circ c : \mathbb{R} \mapsto \mathbb{R}^m$ (see Fig. B.3a). A real-valued **function** f on M is a smooth



(a) A Curve on M



(b) A function $f : M \mapsto \mathbb{R}$

Figure B.3: Curve and function on a manifold

map from M to $\mathbb{R}$. On a chart (U, ϕ), the coordinate representation of f is given by $f \circ \phi^{-1} : \mathbb{R}^m \mapsto \mathbb{R}$ which is a real-valued function of m variables (see Fig. B.3b). We denote the set of all real valued smooth functions on M in the neighborhood around point p by $\mathcal{F}(M)$.

B.2.2 Tangent and Cotangent Spaces

In differential geometry, one can attach to every point p of a differentiable manifold M a tangent space T_pM – a real vector space that intuitively contains the possible directions in which one can tangentially pass through p. The elements of the tangent space at p are called the tangent vectors at p. The manifold and its tangent space has the same dimensions.

Definition 46 (Tangent Vector, Tangent Space) *To define a tangent vector we need a curve $c : (a, b) \mapsto M$ and a function $f : M \to \mathbb{R}$, where (a, b) is an open interval containing $t = 0$. We define the tangent vector at $c(0)$ as a directional derivative of a function $f(c(t))$ along the curve $c(t)$ at $t = 0$. The rate of change of $f(c(t))$ at $t = 0$ along the curve is $\left.\frac{df(c(t))}{dt}\right|_{t=0}$. In terms of local coordinates, we have*

$$\left.\frac{df(c(t))}{dt}\right|_{t=0} = \frac{\partial f}{\partial \boldsymbol{x}} \left.\frac{d\boldsymbol{x}(c(t))}{dt}\right|_{t=0} \tag{B.6}$$

where $\frac{\partial f}{\partial \boldsymbol{x}}$ is a little abuse of notation which actually means $\frac{\partial(f \circ \phi^{-1}(\boldsymbol{x}))}{\partial \boldsymbol{x}}$. In other words, $\frac{df(c(t))}{dt}$ at $t = 0$ is obtained by applying the differential operator X to f i.e.

$$\left.\frac{df(c(t))}{dt}\right|_{t=0} \equiv X[f] \tag{B.7}$$

where the operator expression is given by $X = \frac{d\boldsymbol{x}(c(t))}{dt}\Big|_{t=0} \frac{\partial}{\partial \boldsymbol{x}}$. It is X which we now define as the ***tangent vector*** *to M at $p = c(0)$ along the direction given by the curve $c(t)$. The tangent vectors to all the curves crossing at p which share the same differential operator X, form a vector space (see Section B.1.1) called the* ***tangent space*** *of M at p, denoted by T_pM.*

Evidently, $\boldsymbol{e}_i = \frac{\partial}{\partial x_i}(1 \leq i \leq m)$ are the basis vectors of T_pM, and $\dim T_pM = \dim M$. The basis $\{\boldsymbol{e}_i\}$ is called the **coordinate basis**. If a vector $\boldsymbol{V} \in T_pM$ is written as $\boldsymbol{V} = V_i\boldsymbol{e}_i$, the numbers $V_i \in \mathbb{R}$ are called the components of $\boldsymbol{V}$ with respect to $\boldsymbol{e}_i$. By construction, it is obvious that a vector $\boldsymbol{V}$ exists without specifying the coordinate and that the assignment of the coordinate is simply for our convenience. This coordinate independence of a vector enables us to find the transformation property of the components of the vector. The basis of T_pM need not be $\{\boldsymbol{e}_i\}$, and we may think of the linear combinations $\hat{\boldsymbol{e}}_i \equiv \boldsymbol{A}_i\boldsymbol{e}_i$, where $\boldsymbol{A} = (\boldsymbol{A}_i) \in GL(m, \mathbb{R})$. The basis $\{\hat{\boldsymbol{e}}_i\}$ is known as the **non-coordinate basis**.

Definition 47 (Tangent Bundle) *Let M be a manifold of dimension m. The* ***tangent bundle*** *of M, denoted as TM, is a manifold of dimension $2m$ which is a collection of all tangent spaces of M and hence is defined by the disjoint union of the tangent spaces of M i.e.*

$$TM = \bigcup_{p \in M} T_pM \ . \tag{B.8}$$

An element of TM will be written as (p, X_p) where $p \in M$ and $X_p \in T_pM$. There is a natural projection $\pi : TM \mapsto M$ given by $\pi(X_p) = p$.

Definition 48 (Cotangent Vector, Cotangent Space) *Since T_pM is a vector space, there exists a dual vector space to T_pM, whose element is a linear function from T_pM to $\mathbb{R}$. The dual space is called the* ***cotangent space*** *at p, denoted by T_p^*M and it has the same dimension as T_pM. An element $\omega : T_pM \mapsto \mathbb{R}$ of the cotangent space T_p^*M is called a* ***cotangent vector*** *which is also a dual vector. Its simplest example is the differential df of a function $f \in \mathcal{F}(M)$. The action of a vector V on f is $V[f] \in \mathbb{R}$. Then the action of $df \in T_p^*M$ on $V \in T_pM$ is defined by*

$$< df, V >= V[f] \in \mathbb{R} \ . \tag{B.9}$$

Noting that df is expressed in terms of the local coordinate $\boldsymbol{x} = \phi(p)$ as

$$df(\boldsymbol{x}) = \frac{\partial f}{\partial x_1}(\boldsymbol{x})dx_1 + \ldots + \frac{\partial f}{\partial x_n}(\boldsymbol{x})dx_n \tag{B.10}$$

it is natural to regard dx_i as a basis of T_p^*M. Further, its dual basis since

$$< dx_i, \frac{\partial}{\partial x_j} >= \delta_{ij} \tag{B.11}$$

where δ_{ij} is the Kronecker delta function. Thus an arbitrary cotangent vector can be written as $\boldsymbol{\omega} = \omega_i dx_i$ where the ω_i are the components of $\boldsymbol{\omega}$.

The notion of tangent bundle can be easily extended to cotangent bundle when applied to cotangent space. The cotangent bundle is denoted as T^*M.

Definition 49 (Inner product) *The **inner product** between a tangent vector and cotangent vector $< \quad , \quad >: T_p^*M \times T_pM \mapsto \mathbb{R}$ is defined by*

$$< \boldsymbol{\omega}, \boldsymbol{V} >=< \omega_i dx_i, V_j \frac{\partial}{\partial x_j} >= \omega_i V_j \delta_{ij} = \omega_i V_j \tag{B.12}$$

where $\boldsymbol{\omega} \in T_p^*M$ and $\boldsymbol{V} \in T_pM$.

It should be noted that the inner product is defined between a vector and a dual vector and not between two vectors or two dual vectors.

B.3 Lie Algebra

Every Lie group has an associated Lie algebra, which is the tangent space around the identity element of the group (see Appendix A for an introduction to group theory and Lie groups). That is, the Lie algebra is a vector space generated by differentiating the group transformations along chosen directions in the space, at the identity transformation. The tangent space has the same structure at all group elements, though tangent vectors undergo a coordinate transformation when moved from one tangent space to another. In this section, we will establish basic definitions and concepts used in the study of Lie Algebra.

B.3.1 Vector Field, Flow

If a vector is assigned smoothly to each point of the manifold M, it is called a vector field over M. Each point has its own tangent vector space, so a vector field selects one vector from each space.

Definition 50 (Vector field) *A vector field $F : M \mapsto TM$ is a mapping from the manifold M to the tangent bundle TM so that $\pi \circ F$ is the identity mapping where $\pi : TM \mapsto M$ denotes the projection from TM to M. In terms of a local coordinate* *chart (ϕ, U), a vector field is written as*

$$F(x) = F_1(x)\frac{\partial}{\partial x_1} + \ldots + F_n(x)\frac{\partial}{\partial x_n} \tag{B.13}$$

where each F_i is a smooth function defined on an open neighborhood of $x = \phi(p)$.

Vector fields represent differential equations on manifolds. Let $c : (a, b) \mapsto M$ be a curve on the manifold. The curve c is said to be an **integral curve** of the vector field F if

$$\dot{c}(t) = F(c(t)) \ . \tag{B.14}$$

By the existence and uniqueness theorem for ordinary differential equations, the existence of integral curves for a given nonzero vector field is guaranteed locally. The vector field is said to be complete if the domain of definition of the integral curves can be chosen to be $(-\infty, \infty)$. In this case, the integral curves of a vector field define a one-parameter family of diffeomorphisms $\Phi_t(q) : M \mapsto M$ with the understanding that $\Phi_t(q)$ is the point on the integral curve starting from initial condition q at $t = 0$. This one parameter family of diffeomorphisms is referred to as the **flow** of the vector field F.

B.3.2 Lie Derivative, Lie Brackets and Lie Algebra

Definition 51 (Lie Derivative) *Let F be a smooth vector field and $f \in \mathcal{F}(M)$ a smooth function on M. The **Lie derivative** of f with respect to F is a new function $Ff : M \in \mathbb{R}$ defined by*

$$Ff(p) = F_p f \ . \tag{B.15}$$

In terms of a local coordinate chart (ϕ, U), if we write $F = \sum_{i=1}^{n} F_i(x)\frac{\partial}{\partial x_i}$, then

$$Ff(x) = \sum_{i=1}^{n} \frac{\partial f}{\partial x_i} F_i(x) \tag{B.16}$$

where all partial derivatives are evaluated at $x = \phi(p)$.

Definition 52 (Lie Brackets) *Let F and G be two smooth vector fields. The **Lie bracket** of F and G, denoted $[F, G]$, is a new vector field defined by*

$$[F, G]f = F(Gf) - G(Ff) \ . \tag{B.17}$$

In terms of local coordinates, if we write $F = \sum_{i=1}^{n} F_i(x)\frac{\partial}{\partial x_i}$ and $G = \sum_{i=1}^{n} G_i(x)\frac{\partial}{\partial x_i}$, *then the Lie bracket vector field* $[F, G]$ *is given by*

$$[F, G] = \sum_{j=1}^{n} \left(\sum_{i=1}^{n} \frac{\partial G_j}{\partial x_i} F_i - \frac{\partial F_j}{\partial x_i} G_i \right) \frac{\partial}{\partial x_j} \ . \tag{B.18}$$

Definition 53 (Lie algebra) *A vector space* V *(over* $\mathbb{R}$*) is a* ***Lie algebra*** *if there exists a bilinear operator* $V \times V \mapsto V$*, denoted* $[\cdot, \cdot]$*, satisfying*

- *Skew symmetry:* $[v, w] = -[w, v]$ *for all* $v, w \in V$
- *Jacobi identity:* $[[v, w], z] + [[z, v], w] + [[w, z], v] = 0$ *for all* $v, w, z \in V$

A subspace $W \subset V$ is called a **Lie subalgebra** if $[v, w] \in W$ for all $v, w \in W$. The vector space of all smooth vector fields on a manifold M is an infinite-dimensional Lie algebra under the Lie bracket operation on vector fields.

B.3.3 Lie Algebra associated with a Lie Group

Let X be a vector field on a Lie group G which is also a smooth manifold. X is **left invariant** if $(L_g)_* X = X$, i.e.

$$T_h L_g X(h) = X(gh) \tag{B.19}$$

for all $h \in G$. Let $\mathfrak{X}_L(G)$ be the set of all left invariant vector fields on G. Then for all $X, Y \in \mathfrak{X}_L(G)$ we have

$$L_{g*}[X, Y] = [L_{g*}X, L_{g*}Y] = [X, Y] \ . \tag{B.20}$$

Thus, $\mathfrak{X}_L(G)$ is a Lie subalgebra of the Lie algebra $\mathfrak{X}(G)$, the set of all vector fields on G.

For each $\xi \in T_e G$, we define a vector field X_ξ on G by

$$X_\xi(g) = T_e L_g \xi \ . \tag{B.21}$$

Since,

$$\begin{aligned} X_\xi(gh) &= T_e L_{gh} \cdot \xi = T_e(L_g \circ L_h) \cdot \xi \\ &= T_h L_g (T_e L_h \cdot \xi) = T_h L_g (X_\xi(h)) \ , \end{aligned} \tag{B.22}$$

X_ξ is left invariant. The linear maps $\rho_1 : \mathfrak{X}_L(G) \mapsto T_e G$ given by

$$\rho_1(X) = X(e) \tag{B.23}$$

and $\rho_2 : T_e G \mapsto \mathfrak{X}_L(G)$ given by

$$\rho_2(\xi) = X_\xi \tag{B.24}$$

satisfy $\rho_1 \circ \rho_2 = \mathrm{id}_{T_eG}$ and $\rho_2 \circ \rho_1 = \mathrm{id}_{\mathfrak{X}_L(G)}$. Hence, $\mathfrak{X}_L(G)$ and T_eG are isomorphic vector spaces. Defining a Lie bracket in T_eG by

$$[\xi_1, \xi_2] = [X_{\xi_1}, X_{\xi_2}](e) \ , \quad \xi_1 \ , \xi_2 \in T_eG \tag{B.25}$$

makes T_eG into a Lie algebra. The vector space T_eG with this Lie algebraic structure is called the **Lie algebra** of G and is denoted as g.

B.3.3.1 Translational Group or Euclidean space under addition $(\mathbb{R}^n, +)$

For the translational group $(\mathbb{R}^n, +)$, the identity element is $\boldsymbol{e} = 0, T_0\mathbb{R}^n \cong \mathbb{R}^n$, and it is easy to see that the left invariant field defined by $\boldsymbol{v} \in T_0\mathbb{R}^n$ is the constant vector field $X_v(x) = \boldsymbol{v}$ for all $\boldsymbol{x} \in \mathbb{R}^n$. Therefore, the Lie algebra of $\mathbb{R}^n$ is $\mathbb{R}^n$ itself with the trivial Lie Bracket $[\boldsymbol{v}_1, \boldsymbol{v}_2] = 0$ for all $\boldsymbol{v}_1, \boldsymbol{v}_2 \in \mathbb{R}^n$.

B.3.3.2 Spherical Rotation Group $(SO(3), \cdot)$

The Lie algebra of $SO(3)$, denoted $so(3)$, may be identified with the 3×3 skew-symmetric matrices of the form

$$[\boldsymbol{\omega}] = \begin{bmatrix} 0 & -\omega_3 & \omega_2 \\ \omega_3 & 0 & -\omega_1 \\ -\omega_2 & \omega_1 & 0 \end{bmatrix} \tag{B.26}$$

with the Lie Bracket structure

$$[[\boldsymbol{\omega}_1], [\boldsymbol{\omega}_2]] = [\boldsymbol{\omega}_1][\boldsymbol{\omega}_2] - [\boldsymbol{\omega}_2][\boldsymbol{\omega}_1] \tag{B.27}$$

where $[\boldsymbol{\omega}_1], [\boldsymbol{\omega}_2] \in so(3)$. We can identity $so(3)$ with $\mathbb{R}^3$ using the Equation B.26, which maps a vector $\boldsymbol{\omega} \in \mathbb{R}^3$ to a matrix $[\boldsymbol{\omega}] \in so(3)$. It is straight forward to show that

$$[[\boldsymbol{\omega}_1], [\boldsymbol{\omega}_2]] = (\boldsymbol{\omega}_1 \times \boldsymbol{\omega}_2)^\wedge \tag{B.28}$$

where $\boldsymbol{\omega}_1, \boldsymbol{\omega}_2 \in \mathbb{R}^3$. Thus $\boldsymbol{\omega} \mapsto [\boldsymbol{\omega}]$ is a Lie algebra isomorphism between the Lie algebra $(\mathbb{R}^3, \times)$ and the Lie algebra $(so(3), [\cdot, \cdot])$.

B.3.3.3 Special Euclidean Group $(SE(3), \cdot)$

The Lie algebra of $SE(3)$, denoted $se(3)$, may be identified with the 4×4 matrices of the form

$$[\boldsymbol{V}] = \begin{bmatrix} [\boldsymbol{\omega}] & \boldsymbol{v} \\ \boldsymbol{0} & 0 \end{bmatrix} \tag{B.29}$$

where $\boldsymbol{\omega}, \boldsymbol{v} \in \mathbb{R}^3$. If

$$[\boldsymbol{V}_1] = \begin{bmatrix} [\boldsymbol{\omega}_1] & \boldsymbol{v}_1 \\ \mathbf{0} & 0 \end{bmatrix} \quad [\boldsymbol{V}_2] = \begin{bmatrix} [\boldsymbol{\omega}_2] & \boldsymbol{v}_2 \\ \mathbf{0} & 0 \end{bmatrix} ,$$

then the Lie Bracket structure is given by

$$[[\boldsymbol{V}_1], [\boldsymbol{V}_2]] = [\boldsymbol{V}_1][\boldsymbol{V}_2] - [\boldsymbol{V}_2][\boldsymbol{V}_1] = \begin{bmatrix} [\boldsymbol{\omega}_1 \times \boldsymbol{\omega}_2] & \boldsymbol{v}_1 \times \boldsymbol{\omega}_2 + \boldsymbol{\omega}_1 \times \boldsymbol{v}_2 \\ \mathbf{0} & 0 \end{bmatrix} . \tag{B.30}$$

The vector space $se(3)$ is isomorphic to $\mathbb{R}^6$ via the mapping $[\boldsymbol{V}] \mapsto \boldsymbol{V} = (\boldsymbol{v}, \boldsymbol{\omega}) \in \mathbb{R}^6$.

Appendix C

Algebraic Geometry

"Algebra is nothing more than geometry, in words; geometry is nothing more than algebra, in pictures."

– Marie-Sophie Germain, *French mathematician (1776-1831)*

Algebraic geometry is the study of geometries that come from algebra, in particular, from rings. In classical algebraic geometry, the algebra is the ring of polynomials, and the geometry is the set of zeros of polynomials, called an algebraic variety. [Rowland, 2019]

In the following, some basic definitions and concepts from the field of algebraic geometry are introduced which are useful in the kinematic analysis of robot mechanisms. These are based on [Husty, 2017a] which is derived from an introductory book on algebraic geometry titled "Ideals, Varities and Algorithms" [Cox et al., 2007].

C.1 Basics

Definition 54 (Affine Space) *Affine space of n dimensions is written as K^n. Point in this space will be written as n-tuples $(a_1, a_2, \dots, a_n)$ where $a_i \in K$, $1 \leq i \leq n$.*

Definition 55 (Monomial) *A monomial is a product of power of variables of the form $x_1^{\alpha_1} x_2^{\alpha_2} \dots x_k^{\alpha_k}$. The exponents α_i are condensed into a multi-index $\alpha = (\alpha_1, \alpha_2, \dots, \alpha_n) \in \mathbb{Z}^n_{>=0}$. Using this multi-index notation, a monomial can then be concisely written as x^α.*

Definition 56 (Term) *A term is a monomial multiplied by an element from the ground field K expressed as*

$$ax^\alpha = ax_1^{\alpha_1} x_2^{\alpha_2} \dots x_k^{\alpha_k} \tag{C.1}$$

with $a \in K$.

Definition 57 (Polynomial) *A polynomial f in $x_0, \ldots, x_n$ with coefficients in a field K is a finite linear combination of monomials in the form*

$$f = \sum_{\alpha} a_\alpha x^\alpha, a_\alpha \in K, \alpha \in \mathbb{N} \, .$$

Definition 58 (Polynomial Ring) *The set of all polynomials in $x_0, \ldots, x_n$ with coefficients in K is denoted as $K[x_0, .., x_n]$. The sum and product of two polynomials from this set is again a polynomial in $K[x_0, .., x_n]$ and hence it has the structure of a commutative ring denoted as $K[\underline{x}]$.*

C.2 Ideals and Affine Varieties

Definition 59 (Ideal) *A subset $\mathcal{I} \subset K[x_0, .., x_n]$ is an ideal if it satisifies*

1. $0 \in \mathcal{I}$,
2. *If $f, g \in \mathcal{I}$, then $f + g \in \mathcal{I}$,*
3. *If $f \in \mathcal{I}$ and $g \in K$, then $f \cdot g \in \mathcal{I}$.*

It follows that almost all the ideals are infinite sets of polynomials and can not be written down as a whole. The first natural example of an ideal is the ideal generated by a finite number of polynomials.

Definition 60 *Let $f_1, \ldots, f_s$ be polynomials in $K[\underline{x}]$. Then the set*

$$\mathcal{I} =< f_1, \ldots, f_s >= \{g \in K[\underline{x}] : g = \sum_{i}^{s} h_i f_i \quad and \quad h_1, \ldots, h_s \in K[\underline{x}]\}$$

*is the **ideal generated** by $f_1, \ldots, f_s$.*

The ideal generated by the given polynomials is the set of all combinations of these polynomials using coefficients from $K[x_0, .., x_n]$. The polynomials $f_1, \ldots, f_s$ form the so-called *basis* of the ideal $\mathcal{I} =< f_1, \ldots, f_s >$ which is not unique as the same ideal can be generated by another set of polynomials.

Definition 61 (Affine Variety) *For a given ideal $\mathcal{I} =< f_1, \ldots, f_s >$, the set*

$$V(\mathcal{I}) = \{(a_1, .., a_n) \in K^n : f_i(a_1, .., a_n) = 0 \ , 1 \leq i \leq s\} \subseteq K^n$$

is called the affine variety of the ideal $\mathcal{I}$. In simple words, affine variety is the set of all solutions to the system of equations $f_1(x_1, \ldots, x_n) = \ldots = f_s(x_1, \ldots, x_n) = 0$.

It follows immediately that all bases of the ideal describe the same variety. In general, the variety of an ideal is a more interesting object, not the ideal itself, because the variety is exactly the set of solutions of the input equations $f_1, \dots, f_s$. It should be noted that the variety does not contain information about the multiplicity of solutions. It is nothing more than a set of points in $K[\underline{x}]$.

Example 16 *The equations for a circle and line are given by* $x^2 + y^2 - 1 = 0$ *and* $x + y - 1 = 0$ *respectively. Then the ideal generated by these two equations is given by* $\mathcal{I} =< x^2 + y^2 - 1, x + y - 1 >\subset K[x, y]$ *and the corresponding variety* $V(\mathcal{I})$ *is the set* $\{(1, 0), (0, 1)\}$ *(see Fig. C.1).*

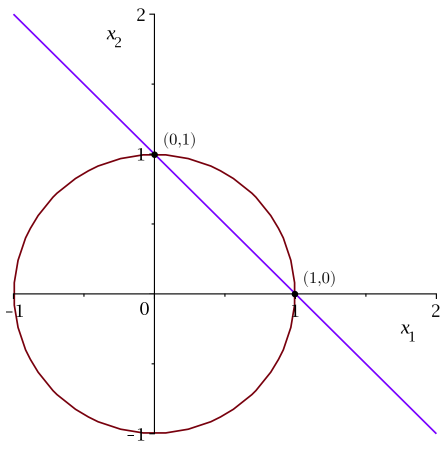


Figure C.1: Variety corresponding to the ideal $\mathcal{I} =< x^2 + y^2 - 1, x + y - 1 >$

It is possible that different ideals describe the same variety.

Definition 62 (Radical) *Let* $\mathcal{I} \subseteq K[\underline{x}]$ *be an ideal. The set*

$$\sqrt{\mathcal{I}} := \{f \in K[\underline{x}] : \exists m \in \mathbb{N}, m \geq 1 \text{ , } f^m \in \mathcal{I}\}$$

is called the radical of $\mathcal{I}$.

The computation of the radical of an ideal $\mathcal{I}$ can be seen as reducing it down to the most important things relevant for its vanishing set $V(\mathcal{I})$.

C.3 Standard Bases

The idea of ordering different terms in polynomial is a key ingredient in polynomial division and row-reduction algorithms.

Definition 63 *Let $\underline{x}^\alpha = x_1^{\alpha_1} \dots x_n^{\alpha_n}$ and $\underline{x}^\beta = x_1^{\beta_1} \dots x_n^{\beta_n}$ be monomials in $K[\underline{x}] = K[x_0,..,x_n]$. To order these monomials a* ***monomial ordering*** *or* ***termorder*** *$>_{\underline{x}}$ on the set of monomials in $K[\underline{x}]$ is defined by an ordering $>$ on the n-tuples $\alpha, \beta \in \mathbb{Z}^n_{>=0}$ which has to fulfill the following conditions:*

- *$>$ is a total ordering on $\mathbb{Z}^n_{>=0}$*
- *if $\alpha > \beta$ and $\gamma \in \mathbb{Z}^n_{>=0}$, then $\alpha + \gamma > \beta + \gamma$*
- *every non-empty subset of $\mathbb{Z}^n_{>=0}$ has a smallest element under $>$*

If such an ordering $>$ on $\mathbb{Z}^n_{>=0}$ is given the monomials are ordered using the following equivalence:

$$\underline{x}^\alpha >_{\underline{x}} \underline{x}^\beta \iff \alpha > \beta$$

So, the monomials are ordered by comparing the ordered n-tuples constructed from the powers of each variable. The most important term orderings are lexicographic order, graded lex order, graded reverse lex order etc. In the scope of the present text, we define only the lexicographic order.

Definition 64 (Lexicographic Order) *Let $\alpha = (\alpha_1, \dots, \alpha_n)$ and $\beta = (\beta_1, \dots, \beta_n)$ be elements of $\mathbb{Z}^n_{>=0}$. We define $\alpha >_{lex} \beta$ if the leftmost non-zero entry of the vector difference $\alpha - \beta \in \mathbb{Z}^n$ is positive.*

Definition 65 (Leading Monomial, Leading Coefficient and Leading Term) *Let $f \in K[\underline{x}]$ be a polynomial with $f = \sum_\alpha a_\alpha x^\alpha$ and let $>_{\underline{x}}$ be a monomial ordering on $K[\underline{x}]$. We define the* ***leading monomial*** *$LM(f)$ as the monomial of f with respect to $>_{\underline{x}}$ with highest degree, the* ***leading coefficient*** *$LC(f)$ as the coefficient of the highest monomial and the* ***leading term*** *as $LT(f) = LC(f) \cdot LM(f)$.*

Definition 66 (Gröbner Basis) *For a fixed monomial order and an ideal $\mathcal{I} \in K[\underline{x}]$, a finite subset $G = g_1, \dots, g_t$ of $\mathcal{I}$ is called a Gröbner Basis or standard basis if*

$$< LM(g_1), \dots, LM(g_t) >=< LM(\mathcal{I}) >$$

where $LM(\mathcal{I})$ is the ideal generated by all the leading terms of the elements of $\mathcal{I}$.

Gröbner Basis is a powerful tool and finds many applications in computational algebra. In the following, we present some of its applications:

1. Solving a system of polynomial equations

   Example 17 *Let us define an ideal $\mathcal{I} =< x^2+y+z = 1, x+y^2+z = 1, x+y+z^2 = 1 >\subset \mathbb{Q}[x,y,z]$ and suppose we want to find all the points on the variety of this* *ideal $\mathbf{V}(\mathcal{I})$. One can compute the Gröbner basis with the lexicographic order $z <_{lex} y <_{lex} x$ as the following:*

$$\begin{aligned} g_1 &:= x + y + z^2 - 1 \\ g_2 &:= y^2 - y - z^2 + z \\ g_3 &:= 2yz^2 + z^4 - z^2 \\ g_4 &:= z^6 - 4z^4 + 4z^3 - z^2 \end{aligned}$$

Thanks to the univariate polynomial (g_4), one can compute all its roots easily and using back substitution in g_3, g_2, g_1, all the points on the ideal $\mathbf{V}(\mathcal{I})$ can be calculated. Powerful computer algebra tools (Maple, Mathematica, MuPAD, Singular etc) can automatically calculate the Gröebner basis and provide all the solutions.

2. Ideal membership problem: Given a polynomial $f \in K[x_0, .., x_n]$ and an ideal $\mathcal{I} =< f_1, \ldots, f_s >$, determine if $f \in \mathcal{I}$. Combing Gröbner Basis with a division algorithm, we can solve the ideal membership problem.

   Theorem 1 *Let $G = \{g_1, \ldots, g_s\} \subseteq K[x_0, .., x_n]$ be a Gröbner basis of an ideal $\mathcal{I}$ and $f \in K[x_0, .., x_n]$. The polynomial f is an element of $\mathcal{I}$ if and only if the remainder of f divided by the ordered s-tuple G is zero, also denoted as $\overline{f}^G = 0$.*

   Example 18 *Let $\mathcal{I} =< xz - y^2, x^3 - z^2 >$ and we want to check if $f = -4x^2y^2z^2 + y^6 + 3z^5$ belongs to the ideal $\mathcal{I}$. Using graded lexicographic ordering, we compute the Gröbner basis of the ideal which is given by:*

$$G = \{f_1, f_2, f_3, f_4, f_5\} = \{xz - y^2, \quad x^3 - z^2, \quad x^2y^2 - z^3, \quad xy^4 - z^4, \quad y^6 - z^5\}$$

   We may now test the polynomial membership in $\mathcal{I}$ by dividing f above by G:

$$f = (-4xy^2z - 4y^4)f_1 + (-3) \cdot f_5 + 0$$

   Since, the remainder is zero, we can see that $f \in \mathcal{I}$.

3. Implicitization problem: Suppose that the parametric equations

$$\begin{aligned} x_1 &= f_1(t_1, \ldots, t_m) \\ &\vdots \\ x_n &= f_n(t_1, \ldots, t_m) \end{aligned} \tag{C.2}$$

   define a subset of an algebraic variety V in K^n. Find out the polynomial equations in x_i that define V.
   We can study the affine variety in K^{m+n} defined by Equation Equation C.2 by

eliminating the variables $t_1, \dots, t_m$ from these equations. This can be done by computing the Gröbner basis with lexicographic ordering $t_1 > \dots > t_m > x_1 > \dots > x_n$. This choice of ordering should eliminate $t_1, \dots, t_m$ and we will be left with candidates of equations of V.

Example 19 *Consider a parametric curve V defined by $x = t^4, y = t^3, z = t^2$ in $\mathbb{C}^3$. We compute the Gröbner basis G of the ideal $\mathcal{I} =< x - t^4, y - t^3, z - t^2 >$ using the lexicographic ordering $\mathbb{C}[t, x, y, z]$ and we find that*

$$G = \{-t^2 + z, ty - z^2, tz - y, x - z^2, y^2 - z^3\}$$

The last two polynomials depend only on x, y, z so they define the affine variety in $\mathbb{C}^3$ containing our curve V.

C.4 Dimension, Primary Decomposition

The most important invariant of a linear subspace of affine space is its dimension. It happens quite often that one is interested in the *dimension* of a variety.

Theorem 2 *Let $V = V(\mathcal{I})$ be an affine variety where $\mathcal{I} \subset K[x_0, .., x_n]$ is an ideal. If the ground field K is algebraically closed, then the dimension of V is the maximum dimension of the coordinate subspace*[1] *in $V(< LT(\mathcal{I}) >)$.*

Another possibility is to compute the Hilbert dimension of the algebraic variety. If the Hilbert dimension is zero, it means that there are finite number of intersections between the polynomials that constitute the corresponding ideal. If the variety contains parts with different dimensions, then the dimension of the whole variety is defined to be the largest of these numbers. A proposal to define the degree of freedom of a mechanism based on the Hilbert dimension of the ideal of the constraint polynomials was proposed in [Husty and Schröcker, 2011].

Example 20 *Consider an ideal $\mathcal{I} =< (2x_1 - x_2 - 2)x_1, (2x_1 - x_2 - 2)x_2^2 >$ whose vanishing set $V(\mathcal{I})$ is a union of a line described by $2x_1 - x_2 - 2$ and an isolated point $(0, 0)$. The Hilbert dimension of the $\mathcal{I}$ is 1 which is the larger of the line and isolated point.*

If the variety is composed of simpler varieties as in the example above, it is also interesting to study its decomposition. To do that, we introduce some preliminary concepts in that direction.

[1]In K^n , a vector subspace defined by setting some subset of the variables $x_1, \dots, x_n$ equal to zero is called a coordinate subspace.

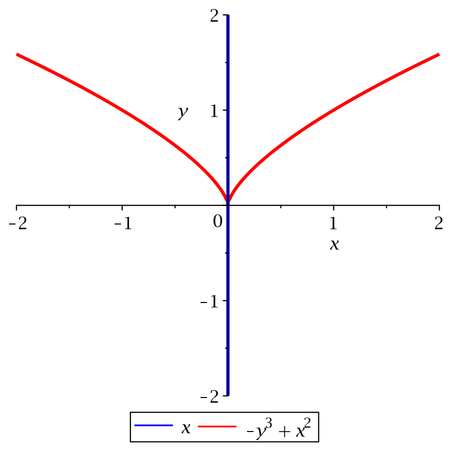


Figure C.2: Variety $V(< x^3 - xy^3 >)$ as the union of two varities $V(< x >)$ and $V(< x^2 - y^3 >)$

Definition 67 (Ideal Intersection) *The intersection of the two ideals $\mathcal{I}$ and $\mathcal{J}$ in $K[\underline{x}]$, $\mathcal{I} \cap \mathcal{J}$, is the set of polynomials which belong to both $\mathcal{I}$ and $\mathcal{J}$.*

The intersection of ideals is equivalent to the union of corresponding varieties according to the following theorem.

Theorem 3 *If $\mathcal{I}$ and $\mathcal{J}$ are two ideals in $K[\underline{x}]$, then $V(\mathcal{I} \cap \mathcal{J}) = V(\mathcal{I}) \cup V(\mathcal{J})$.*

A *primary ideal* and a *prime ideal* are defined as follows.

Definition 68 (Primary Ideal) *An ideal $\mathcal{I} \subseteq K[\underline{x}]$ is primary if $fg \in \mathcal{I}$ implies either $f \in \mathcal{I}$ or some power $g^m \in \mathcal{I}$ for $m > 0$.*

Definition 69 (Prime Ideal) *An ideal $\mathcal{I} \subseteq K[\underline{x}]$ is prime if $fg \in \mathcal{I}$ implies either $f \in \mathcal{I}$ or $g \in \mathcal{I}$.*

In this light, the following theorem is stated.

Theorem 4 *Every ideal $\mathcal{I} \subseteq K[\underline{x}]$ can be written as a finite intersection of primary ideals.*

Example 21 *$< x^2 >$ is a primary ideal, but not prime ideal.*

Definition 70 (Primary Decomposition) *A primary decomposition of a given ideal $\mathcal{I}$ is an expression of $\mathcal{I}$ as an intersection of primary ideals, namely $\mathcal{I} = \cap_{i=1}^{r} \mathcal{Q}_i$. Such a decomposition is called **minimal** if the radicals $\sqrt{\mathcal{Q}_i}$ are all different and $\mathcal{Q}_i \not\supseteq \cap_{i \neq j}^{r} \mathcal{Q}_j$. Furthermore, if no radical $\sqrt{\mathcal{Q}_i}$ is strictly contained in another radical $\sqrt{\mathcal{Q}_j}$, then the primary components are uniquely determined. The radicals $\sqrt{\mathcal{Q}_i} =: \mathcal{P}_i$ are the corresponding prime ideals.*

Example 22 *The primary decomposition of the ideal* $< x^3 - xy^3 > \subset K[x, y]$ *yields two prime ideals* $< x >$ *and* $< x^2 - y^3 >$. *Fig. C.2 shows the primary decomposition of the ideal as the union of its corresponding varieties.*

Appendix D

Screw Theory

"The description of right lines and circles, upon which geometry is founded, belongs to mechanics. Geometry does not teach us to draw these lines, but requires them to be drawn."

– Sir Issac Newton, *English mathematician (1642-1727)*

Screw theory is the algebra and calculus of pairs of vectors, such as forces and moments and angular and linear velocity, that arise in the kinematics and dynamics of rigid bodies. Expressing the motion of a rigid body as a combination of a rotation and a translation about a line was first proposed by Chasles (1830) and further developed by Poinsot (1848). Then, Julius Plücker came up with a way to assign six homogenous coordinates for a line [Plücker, 1865]. In 1876, Sir Robert Stawell Ball developed a mathematical framework of screw theory for applications in rigid body mechanics [Ball, 1876]. K.H. Hunt [Davidson and Hunt, 2004] further developed screw theory with a geometrical emphasis. Using line geometry, the major contribution of Hunt was to classify the various screw systems.

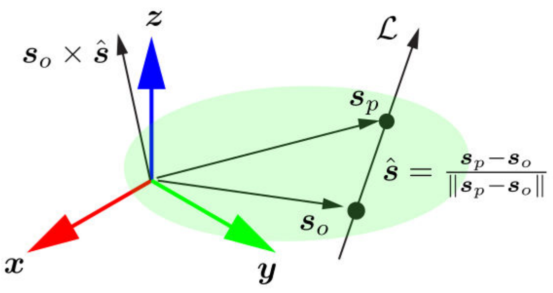
D.1 Basics

Theorem 5 (Chasles' Theorem) *The most general motion of a rigid body consists of a rotation about a line in space together with a translation along it. Such a quantity is called **twist** or **spatial velocity**.*

$$\boldsymbol{V} = \begin{bmatrix} \boldsymbol{\omega} \\ \boldsymbol{v} \end{bmatrix} \in \mathbb{R}^6 \tag{D.1}$$

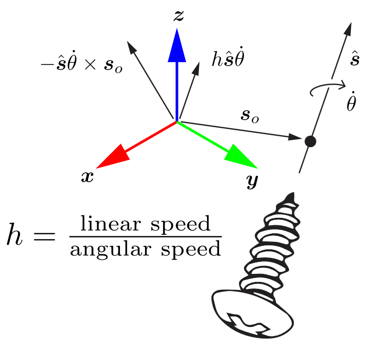
Theorem 6 (Poinsot's Theorem) *The most general force that can act on a rigid body consists of a linear force acting along a line in space, together with a moment* *acting about it. Such a quantity is called a **wrench** or **spatial force**.*

$$\boldsymbol{W} = \begin{bmatrix} \boldsymbol{m} \\ \boldsymbol{f} \end{bmatrix} \in \mathbb{R}^6 \tag{D.2}$$



$\boldsymbol{s}_o \times \hat{\boldsymbol{s}}$
$\boldsymbol{z}$
$\mathcal{L}$
$\boldsymbol{s}_p$
$\hat{\boldsymbol{s}} = \frac{\boldsymbol{s}_p - \boldsymbol{s}_o}{\|\boldsymbol{s}_p - \boldsymbol{s}_o\|}$
$\boldsymbol{s}_o$
$\boldsymbol{x}$
$\boldsymbol{y}$

(a) Plücker coordinates of a line $\mathcal{L}$



z
$-\hat{s}\dot{\theta} \times s_o$
$h\hat{s}\dot{\theta}$
$\hat{s}$
$\dot{\theta}$
s_o
x
y
$h = \frac{\text{linear speed}}{\text{angular speed}}$

(b) Screw motion

Figure D.1: Screw representation with Plücker coordinates of a line along the screw axis and the pitch, Fig. D.1b is adapted from [Lynch and Park, 2017]

Definition 71 (Plücker coordinates of a line) *The Plücker coordinates of a line $\mathcal{L}$ defined by two points in 3-D Euclidean space is given by the unit direction vector between those points $\hat{\mathrm{s}}$ and a moment vector $s_o \times \hat{\mathrm{s}}$ (see Fig. D.1a for a visualization).*

$$\mathcal{L} = \begin{bmatrix} \hat{\boldsymbol{s}} \\ \boldsymbol{s}_o \times \hat{\boldsymbol{s}} \end{bmatrix} \tag{D.3}$$

Definition 72 (Screw) *A screw S is defined by a unit direction axis $\hat{\mathrm{s}}$, the position of a point s_o on this axis with respect to a reference frame and pitch h (see Fig. D.1b).*

$$\boldsymbol{S} = \begin{bmatrix} \hat{\boldsymbol{s}} \\ \boldsymbol{s}_o \times \hat{\boldsymbol{s}} + h\hat{\boldsymbol{s}} \end{bmatrix} \in \mathbb{R}^6 \tag{D.4}$$

It is to be noted that setting $h = 0$, the vector $\begin{bmatrix} \hat{\boldsymbol{s}} \\ \boldsymbol{s}_o \times \hat{\boldsymbol{s}} \end{bmatrix}$ are the Plücker coordinates of a line along the screw axis. Geometrically, a screw is determined by the Plücker coordinates of a line along the screw axis and a pitch (see Fig. D.1). In classical screw theory literature, it is often denoted as \$.

Once, we have defined the notion of a screw $\boldsymbol{V}$, a twist can be interpreted in terms of this screw and a velocity $\dot{\theta}$ about it. The expression for the twist is given by $\boldsymbol{V} = \boldsymbol{S}\dot{\theta}$.

$$\begin{bmatrix} \boldsymbol{\omega} \\ \boldsymbol{v} \end{bmatrix} = \begin{bmatrix} \hat{\boldsymbol{s}} \\ \boldsymbol{s}_o \times \hat{\boldsymbol{s}} + h\hat{\boldsymbol{s}} \end{bmatrix} \dot{\theta} = \begin{bmatrix} \hat{\boldsymbol{s}}\dot{\theta} \\ -\hat{\boldsymbol{s}}\dot{\theta} \times \boldsymbol{s}_o + h\hat{\boldsymbol{s}}\dot{\theta} \end{bmatrix} \tag{D.5}$$

Note that the linear velocity v is the sum of two terms: one due to translation along the screw axis $h\hat{s}\dot{\theta}$ and other due to the linear motion induced by rotation about the axis $-s\dot{\theta} \times s_o$. The first term $\hat{s}$ is in the direction of the screw axis and second term is in the plane orthogonal to $\hat{s}$. Recalling the scalar-[1] and vector-[2] triple product identities, it is easy to show that, for any $\boldsymbol{V} = (\boldsymbol{\omega}, \boldsymbol{v})$ where $\boldsymbol{\omega} \neq \mathbf{0}$, there exists an equivalent screw axis $\{\boldsymbol{s}_o, \hat{s}, h\}$ and velocity $\dot{\theta}$ where:

$$\dot{\theta} = \|\boldsymbol{\omega}\| \qquad \hat{s} = \frac{\boldsymbol{\omega}}{\|\boldsymbol{\omega}\|} \qquad h = \frac{\boldsymbol{\omega} \cdot \boldsymbol{v}}{\|\boldsymbol{\omega}\|} \qquad \boldsymbol{s}_o = \frac{\boldsymbol{\omega} \times \boldsymbol{v}}{\|\boldsymbol{\omega}\|} \tag{D.6}$$

If $\boldsymbol{\omega} = 0$, then the pitch h of the screw is infinite. In this case, $\hat{s}$ is chosen as $\boldsymbol{v}/\|\boldsymbol{v}\|$ and $\dot{\theta}$ is interpreted as linear velocity $\|\boldsymbol{v}\|$ along $\hat{s}$.

The screws can be classified into three basic types:

- *zero-pitch screws:* corresponds to pure rotation movements with $h = 0$, $\boldsymbol{S}_0 = \begin{bmatrix} \hat{\boldsymbol{s}} \\ \boldsymbol{s}_o \times \hat{\boldsymbol{s}} \end{bmatrix}$
- *infinite-pitch screws:* corresponds to pure translation movements with $h = \infty$, $\boldsymbol{S}_\infty = \begin{bmatrix} \mathbf{0}_{3\times 1} \\ \hat{\boldsymbol{s}} \end{bmatrix}$
- *finite-pitch screws:* $h \neq 0$, $h \neq \infty$. These can be written as a linear combination of the zero pitch screw $\boldsymbol{S}_0$ and the infinite pitch screw $\boldsymbol{S}_\infty$.

Definition 73 (Normalized or Unit Screw) *For a given reference frame, a screw axis $\boldsymbol{S}$ is written as*

$$\boldsymbol{S} = \begin{bmatrix} \boldsymbol{\omega} \\ \boldsymbol{v} \end{bmatrix} \in \mathbb{R}^6 \tag{D.7}$$

where either (i) $\|\boldsymbol{\omega}\| = 1$ or (ii) $\|\boldsymbol{\omega}\| = 0$ and $\|\boldsymbol{v}\| = 1$. If (i) holds, then $\boldsymbol{v} = -\boldsymbol{\omega} \times \boldsymbol{s}_o + h\boldsymbol{\omega}$. If (ii) holds, then the pitch of the screw is infinite and the twish is the translation along the axis defined by v [Lynch and Park, 2017].

D.2 Reciprocity Conditions

The dot product between twists and wrenches gives the instantaneous power associated with moving a rigid body through an applied force. A wrench $\boldsymbol{F}$ is said to be reciprocal to a twist $\boldsymbol{V}$ if this *instantaneous power* is zero i.e. $\boldsymbol{F} \cdot \boldsymbol{V} = 0$. Since both twists and wrenches can be represented by screws, we can use them to define the notion of reciprocal screws.

[1] $\boldsymbol{a} \cdot (\boldsymbol{b} \times \boldsymbol{c}) = \boldsymbol{b} \cdot (\boldsymbol{c} \times \boldsymbol{a}) = \boldsymbol{c} \cdot (\boldsymbol{a} \times \boldsymbol{b})$
[2] $\boldsymbol{a} \times (\boldsymbol{b} \times \boldsymbol{c}) = \boldsymbol{b}(\boldsymbol{a} \cdot \boldsymbol{c}) - \boldsymbol{c}(\boldsymbol{a} \cdot \boldsymbol{b})$

Screw 1	Screw 2	Reciprocity condition
$\boldsymbol{S}_0$	$\boldsymbol{S}_\infty$	orthogonal axes
$\boldsymbol{S}_\infty$	$\boldsymbol{S}_0$	orthogonal axes
$\boldsymbol{S}_0$	$\boldsymbol{S}_0$	coplanar axes
$\boldsymbol{S}_\infty$	$\boldsymbol{S}_\infty$	always reciprocal

Table D.1: Reciprocity conditions

Definition 74 (Reciprocal Screws) *Two screws $\boldsymbol{S}_1$ and $\boldsymbol{S}_2$ are said to be reciprocal if the twist $\boldsymbol{V}$ about $\boldsymbol{S}_1$ and the wrench $\boldsymbol{F}$ along $\boldsymbol{S}_2$ are reciprocal. This is equivalent to the following condition:*

$$\boldsymbol{S}_1 \odot \boldsymbol{S}_2 = (\boldsymbol{\Pi}\boldsymbol{S}_1)^T \boldsymbol{S}_2 = 0 \quad \textit{where} \quad \boldsymbol{\Pi} = \begin{bmatrix} \mathbf{0}_{3\times3} & \boldsymbol{I}_{3\times3} \\ \boldsymbol{I}_{3\times3} & \mathbf{0}_{3\times3} \end{bmatrix} \tag{D.8}$$

where $\odot$ denotes the reciprocal product of two screws and expressions for $\boldsymbol{S}_1$ and $\boldsymbol{S}_2$ are given by:

$$\boldsymbol{S}_1 = \begin{bmatrix} \hat{\boldsymbol{s}}_1 \\ \boldsymbol{s}_{o1} \times \hat{\boldsymbol{s}}_1 + h_1\hat{\boldsymbol{s}}_1 \end{bmatrix}, \boldsymbol{S}_2 = \begin{bmatrix} \boldsymbol{s}_{o2} \times \hat{\boldsymbol{s}}_2 + h_2\hat{\boldsymbol{s}}_2 \\ \hat{\boldsymbol{s}}_2 \end{bmatrix}. \tag{D.9}$$

Table D.1 presents some important examples of two reciprocal screws. The notion of reciprocal screws is important in the analysis of parallel mechanisms as well contact modeling.

D.3 Screw Systems

Definition 75 (Screw System) *A screw system of dimension n $(0 \le n \le 6)$ comprises of n linearly independent screws that result in a n degree-of-freedom system.*

[Davidson and Hunt, 2004] provides an extensive treatment on all screw systems, the one-system to the five-system, in their general and special forms. The *one-system* consists of one screw. The *two-system* comprises of ∞^1 screws and the general ruled surface on which these screws lie is a cylindroid. The *three-system* comprises of ∞^2 screws and the general ruled surface on which these screws lie is a hyperboloid and so on. These screw systems explain the transitory mobility of spatial linkages which can be extended to full-cycle mobility for a great number of linkages.

D.3.1 Serial Robots

The twist system $\mathcal{T}$ of a serial kinematic chain with n degrees of freedom is the linear combination of the twist systems of its kinematic joints $\mathcal{T}^i$. Its wrench system is the intersection of the wrench systems of its kinematic joints $\mathcal{W}^i$.

$$\mathcal{T} = \bigoplus_{i=1}^{n} \mathcal{T}^i \quad , \quad \mathcal{W} = \bigcap_{i=1}^{n} \mathcal{W}^i \tag{D.10}$$

Hence, it is easier to find the twist system of a serial mechanism and its wrench system can be computed by finding the reciprocal of its twist system.

D.3.2 Parallel Robots

The twist system $\mathcal{T}$ of a PKM consisting of a set of m serial kinematic chains is the intersection of the twist system from each serial kinematic chain $\mathcal{T}^i$. Its wrench system is the linear combination of the wrench system from each serial kinematic chain $\mathcal{W}^i$:

$$\mathcal{T} = \bigcap_{i=1}^{m} \mathcal{T}^i \quad , \quad \mathcal{W} = \bigoplus_{i=1}^{m} \mathcal{W}^i \tag{D.11}$$

Hence, during the analysis, it is easier to find the wrench system of the whole PM as the sum of wrench systems of the individual legs and then finding the reciprocal twist of the whole manipulator.

D.4 Actuation and Constraint Wrench Systems

A spatial parallel manipulator consists of a *total wrench system* of dimension 6. It can be considered as the linear combination of *actuation wrench system*, $\mathcal{W}^a$ and *constraint wrench system*, $\mathcal{W}^c$.

If the PM has n number of legs, each leg has a twist system $\mathcal{T}_i$, $(i = 1, ..., n)$ of dimension t. Hence, the constraint wrench system, $\mathcal{W}_i^c$ of each leg will be of dimension $6 - t$. Constraint wrench system of the manipulator of dimension c would be:

$$\mathcal{W}_c = \mathcal{W}_1^c \oplus \mathcal{W}_2^c \oplus ... \oplus \mathcal{W}_n^c \tag{D.12}$$

If each leg has $n(< t)$ unactuated joints, the wrench system reciprocal to unactuated joints is of dimension $6 - n$. Then, actuation wrench system of each leg consists of wrenches that do not belong to $\mathcal{W}^c$. Hence, the actuation wrench of the whole manipulator dimension $6 - c$ would be:

$$\mathcal{W}_a = \mathcal{W}_1^a \oplus \mathcal{W}_2^a \oplus ... \oplus \mathcal{W}_n^a \tag{D.13}$$

Therefore, the total wrench system of the PM can be found as

$$\mathcal{W} = \mathcal{W}_a \oplus \mathcal{W}_c \tag{D.14}$$

In general configuration, the constraint and actuation wrench systems of parallel manipulators form a 6-system. It means that by locking the actuators, the moving platform must be fully constrained, otherwise the manipulator reaches parallel singularity [Kong and Gosselin, 2007].

Based on the theory of reciprocal screws [Davidson and Hunt, 2004, Kong and Gosselin, 2007], Joshi and Tsai [Joshi and Tsai, 2002] developed a methodology to express 6×6 Jacobian matrix for lower-mobility PMs that includes constraint and actuation wrenches. The active joint velocity vector $\dot{\boldsymbol{q}}_a$ is related to the platform twist $\boldsymbol{V}_p$ by the following relation.

$$\dot{\boldsymbol{q}}_a = \mathcal{W}_a \boldsymbol{V}_p \tag{D.15}$$

Further, the constraint wrench system and the platform twish are related.

$$\boldsymbol{0} = \mathcal{W}_c \boldsymbol{V}_p \tag{D.16}$$

Combining Equation D.15 and Equation D.16, one could arrive at the following equation.

$$\underbrace{\begin{bmatrix} \dot{\boldsymbol{q}}_a \\ \boldsymbol{0} \end{bmatrix}}_{\dot{\boldsymbol{q}}_o} = \underbrace{\begin{bmatrix} \mathcal{W}_a \\ \mathcal{W}_c \end{bmatrix}}_{\boldsymbol{J}_E} \boldsymbol{V}_p \tag{D.17}$$

One could notice in the above equation that $\boldsymbol{J}_E$ is the concatenation of actuation and constraint wrenches of the PM and is also referred to as extended Jacobian of a PM.

List of Figures

List of Tables

Bibliography

[Mer, 2006] (2006). *Structural synthesis and architectures*, pages 19–94. Springer Netherlands, Dordrecht.

[Abdellatif and Heimann, 2010] Abdellatif, H. and Heimann, B. (2010). Advanced model-based control of a 6-dof hexapod robot: A case study. *IEEE/ASME Transactions on Mechatronics*, 15(2):269–279.

[Al-Widyan et al., 2011] Al-Widyan, K., Ma, X. Q., and Angeles, J. (2011). The robust design of parallel spherical robots. *Mechanism and Machine Theory*, 46(3):335–343.

[Ambrose et al., 2000] Ambrose, R. O., Aldridge, H., Askew, R. S., Burridge, R. R., Bluethmann, W., Diftler, M., Lovchik, C., Magruder, D., and Rehnmark, F. (2000). Robonaut: Nasa's space humanoid. *IEEE Intelligent Systems and Their Applications*, 15(4):57–63.

[ANYbotics, 2016] ANYbotics (2016). ANYdrive. https://www.anybotics.com/anydrive/. Accessed: 15.05.2018.

[Aragão et al., 2016] Aragão, M., Moreno, P., and Bernardino, A. (2016). Middleware interoperability for robotics: A ros–yarp framework. *Frontiers in Robotics and AI*, 3:64.

[Arts2Science, 2018] Arts2Science (2018). How do muscles move bones? Contracting and relaxing. https://arts2science.wordpress.com/humanbody/locomotorsystem/musclesmovebones/. Accessed: 10.05.2018.

[Bai et al., 2017] Bai, S., Christensen, S., and Islam, M. R. U. (2017). An upper-body exoskeleton with a novel shoulder mechanism for assistive applications. In *2017 IEEE International Conference on Advanced Intelligent Mechatronics (AIM)*, pages 1041–1046.

[Ball, 1876] Ball, R. S. (1876). The theory of screws: A study in the dynamics of a rigid body. *Mathematische Annalen*, 9(4):541–553.

[Bargsten and de Gea Fernandez, 2015] Bargsten, V. and de Gea Fernandez, J. (2015). Compi: Development of a 6-dof compliant robot arm for human-robot cooperation. In *Proceedings of the 8th International Workshop on Human-Friendly Robotics*. TU München.

[Bartsch et al., 2016] Bartsch, S., Manz, M., Kampmann, P., Dettmann, A., Hanff, H., Langosz, M., v. Szadkowski, K., Hilljegerdes, J., Simnofske, M., Kloss, P., Meder, M., and Kirchner, F. (2016). Development and control of the multi-legged robot mantis. In *Proceedings of ISR 2016: 47st International Symposium on Robotics*, pages 1–8.

[Bongardt, 2011] Bongardt, B. (2011). *CAD-2-SIM – Kinematic Modeling of Mechanisms Based on the Sheth-Uicker Convention*, pages 465–477. Springer Berlin Heidelberg, Berlin, Heidelberg.

[Bongardt, 2013] Bongardt, B. (2013). Sheth uicker convention revisited. *Mechanism and Machine Theory*, 69:200 – 229.

[Boothby, 2003] Boothby, W. (2003). *An Introduction to Differentiable Manifolds and Riemannian Geometry, Revised*. Pure and Applied Mathematics. Elsevier Science.

[Bottema and Roth, 1990] Bottema, O. and Roth, B. (1990). *Theoretical Kinematics*. Dover Books on Physics. Dover Publications.

[Bridgwater et al., 2012] Bridgwater, L. B., Ihrke, C., Diftler, M. A., Abdallah, M. E., Radford, N. A., Rogers, J., Yayathi, S., Askew, R. S., and Linn, D. M. (2012). The robonaut 2 hand-designed to do work with tools. In *International Conference on Robotics and Automation (ICRA)*, pages 3425–3430. IEEE.

[Brinker and Corves, 2015] Brinker, J. and Corves, B. (2015). A Survey on Parallel Robots with Delta-like Architecture. In *The 14th IFToMM World Congress, 25.10.2015-30.10.2015, Taipei, Taiwan*.

[Brinker et al., 2017] Brinker, J., Funk, N., Ingenlath, P., Takeda, Y., and Corves, B. (2017). Comparative study of serial-parallel delta robots with full orientation capabilities. *IEEE Robotics and Automation Letters*, 2(2):920–926.

[Brockett, 1984] Brockett, R. W. (1984). Robotic manipulators and the product of exponentials formula. In Fuhrmann, P. A., editor, *Mathematical Theory of Networks and Systems*, pages 120–129, Berlin, Heidelberg. Springer Berlin Heidelberg.

[Bruyninckx, 2001] Bruyninckx, H. (2001). Open robot control software: the orocos project. In *Proceedings 2001 ICRA. IEEE International Conference on Robotics and Automation (Cat. No.01CH37164)*, volume 3, pages 2523–2528 vol.3.

[Bruyninckx et al., 2003] Bruyninckx, H., Soetens, P., and Koninckx, B. (2003). The real-time motion control core of the orocos project. In *2003 IEEE International Conference on Robotics and Automation (Cat. No.03CH37422)*, volume 2, pages 2766–2771 vol.2.

[Buffinton, 2005] Buffinton, K. W. (2005). *Robotics and Automation Handbook*, chapter Kane's Method in Robotics. CRC Press.

[Buschmann et al., 2013] Buschmann, T., Favot, V., Schwienbacher, M., Ewald, A., and Ulbrich, H. (2013). Dynamics and control of the biped robot lola. In Gattringer, H. and Gerstmayr, J., editors, *Multibody System Dynamics, Robotics and Control*, pages 161–173, Vienna. Springer Vienna.

[Cafolla et al., 2016] Cafolla, D., Wang, M., Carbone, G., and Ceccarelli, M. (2016). Larmbot: A new humanoid robot with parallel mechanisms. In Parenti-Castelli, V. and Schiehlen, W., editors, *ROMANSY 21 - Robot Design, Dynamics and Control*, pages 275–283, Cham. Springer International Publishing.

[Ceccarelli et al., 2018] Ceccarelli, M., Cafolla, D., Russo, M., and Carbone, G. (2018). Heritagebot platform for service in cultural heritage frames. *International Journal of Advanced Robotic Systems*, 15(4):1729881418790692.

[Cheng et al., 2003] Cheng, H., Yiu, Y.-K., and Li, Z. (2003). Dynamics and control of redundantly actuated parallel manipulators. *IEEE/ASME Transactions on Mechatronics*, 8(4):483–491.

[Cordes et al., 2017] Cordes, F., Babu, A., and Kirchner, F. (2017). Static force distribution and orientation control for a rover with an actively articulated suspension system. In *Proceedings of the 2017 IEEE/RSJ International Conference on Intelligent Robots and Systems. IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS-17), Friendly People, Friendly Robots, September 24-28, Vancouver,, BC, Canada*. IEEE/RSJ.

[Cordes et al., 2018] Cordes, F., Kirchner, F., and Babu, A. (2018). Design and field testing of a rover with an actively articulated suspension system in a mars analog terrain. *Journal of Field Robotics (JFR)*, 35(7):1149–1181.

[Cordes et al., 2014] Cordes, F., Oekermann, C., Babu, A., Kuehn, D., Stark, T., and Kirchner, F. (2014). An active suspension system for a planetary rover. In *Proceedings of the International Symposium on Artificial Intelligence, Robotics and Automation in Space (i-SAIRAS 2014) Montreal, Canada*. o.A.

[Cox et al., 2007] Cox, D. A., Little, J., and O'Shea, D. (2007). *Ideals, Varieties, and Algorithms: An Introduction to Computational Algebraic Geometry and Commutative Algebra, 3/e (Undergraduate Texts in Mathematics)*. Springer-Verlag New York, Inc., Secaucus, NJ, USA.

[D-RoCK, 2018] D-RoCK (2018). Models, methods and tools for the model based software development of robots. `https://robotik.dfki-bremen.de/en/research/projects/d-rock.html`. Accessed: 10.05.2018.

[Davidson and Hunt, 2004] Davidson, J. and Hunt, K. (2004). *Robots and Screw Theory: Applications of Kinematics and Statics to Robotics*. Oxford University Press.

[Delp et al., 2007] Delp, S. L., Anderson, F. C., Arnold, A. S., Loan, P., Habib, A., John, C. T., Guendelman, E., and Thelen, D. G. (2007). Opensim: Open-source software to create and analyze dynamic simulations of movement. *IEEE Transactions on Biomedical Engineering*, 54(11):1940–1950.

[Diftler et al., 2011] Diftler, M. A., Mehling, J., Abdallah, M. E., Radford, N. A., Bridgwater, L. B., Sanders, A. M., Askew, R. S., Linn, D. M., Yamokoski, J. D., Permenter, F., et al. (2011). Robonaut 2-the first humanoid robot in space. In *Robotics and Automation (ICRA), 2011 IEEE International Conference on*, pages 2178–2183. IEEE.

[Elkady and Sobh, 2012] Elkady, A. and Sobh, T. (2012). Robotics middleware: A comprehensive literature survey and attribute-based bibliography. *Journal of Robotics*, 2012.

[Englsberger et al., 2014] Englsberger, J., Werner, A., Ott, C., Henze, B., Roa, M. A., Garofalo, G., Burger, R., Beyer, A., Eiberger, O., Schmid, K., and Albu-Schäffer, A. (2014). Overview of the torque-controlled humanoid robot toro. In *2014 IEEE-RAS International Conference on Humanoid Robots*, pages 916–923.

[Eppstein, 1992] Eppstein (1992). Parallel recognition of series-parallel graphs. *Information and Computation*, 98(1):41 – 55.

[EtherCAT Technology Group, 2003] EtherCAT Technology Group (2003). EtherCAT - the ethernet fieldbus. `https://www.ethercat.org/en/technology.html`. Accessed: 2018-09-06.

[Featherstone, 2008] Featherstone, R. (2008). *Rigid Body Dynamics Algorithm*.

[Felis, 2017] Felis, M. L. (2017). Rbdl: an efficient rigid-body dynamics library using recursive algorithms. *Autonomous Robots*, 41(2):495–511.

[Fiorio et al., 2017] Fiorio, L., Scalzo, A., Natale, L., Metta, G., and Parmiggiani, A. (2017). A parallel kinematic mechanism for the torso of a humanoid robot: Design, construction and validation. In *2017 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 681–688.

[Frindt, 2001] Frindt, M. (2001). *Modulbasierte Synthese von Parallelstrukturen für Maschinen in der Produktionstechnik*. PhD thesis, Technical University of Braunschweig.

[Gallardo-Alvarado, 2016] Gallardo-Alvarado, J. (2016). *Kinematic Analysis of Parallel Manipulators by Algebraic Screw Theory*. 1 edition.

[Garcia et al., 2011] Garcia, E., Arevalo, J. C., Muñoz, G., and Gonzalez-de Santos, P. (2011). On the biomimetic design of agile-robot legs. *Sensors*, 11(12):11305–11334.

[Gautier et al., 1995] Gautier, M., Khalil, W., and Restrepo, P. P. (1995). Identification of the dynamic parameters of a closed loop robot. In *Proceedings of 1995 IEEE International Conference on Robotics and Automation*, volume 3, pages 3045–3050 vol.3.

[Ghorbel et al., 2000] Ghorbel, F. H., Chetelat, O., Gunawardana, R., and Longchamp, R. (2000). Modeling and set point control of closed-chain mechanisms: theory and experiment. *IEEE Transactions on Control Systems Technology*, 8(5):801–815.

[Google, 2017] Google (2017). Protocol Buffers. `https://developers.google.com/protocol-buffers/`. Accessed 18.5.2018.

[Gopura et al., 2011] Gopura, R. A. R. C., Kiguchi, K., and Bandara, D. S. V. (2011). A brief review on upper extremity robotic exoskeleton systems. In *2011 6th International Conference on Industrial and Information Systems*, pages 346–351.

[Gosselin and Hamel, 1994] Gosselin, C. M. and Hamel, J. . (1994). The agile eye: a high-performance three-degree-of-freedom camera-orienting device. In *Proceedings of the 1994 IEEE International Conference on Robotics and Automation*, pages 781–786 vol.1.

[Gosselin et al., 1996] Gosselin, C. M., Pierre, E. S., and Gagne, M. (1996). On the development of the agile eye. *IEEE Robotics Automation Magazine*, 3(4):29–37.

[Guennebaud et al., 2010] Guennebaud, G., Jacob, B., et al. (2010). Eigen v3. http://eigen.tuxfamily.org.

[Gutiérrez et al., 2018] Gutiérrez, C. S. V., Juan, L. U. S., Ugarte, I. Z., and Vilches, V. M. (2018). Time-sensitive networking for robotics. *CoRR*, abs/1804.07643.

[Hartenberg and Scheunemann, 1955] Hartenberg, D. and Scheunemann, R. (1955). A kinematic notation for lower-pair mechanisms based on matrices. *Trans ASME J. Appl. Mech.*

[Hebert et al., 2015] Hebert, P., Bajracharya, M., Ma, J., Hudson, N., Aydemir, A., Reid, J., Bergh, C., Borders, J., Frost, M., Hagman, M., Leichty, J., Backes, P., Kennedy, B., Karplus, P., Satzinger, B., Byl, K., Shankar, K., and Burdick, J. (2015). Mobile manipulation and mobility as manipulation—design and algorithms of robosimian. *Journal of Field Robotics*, 32(2):255–274.

[Herr, 2009] Herr, H. (2009). Exoskeletons and orthoses: classification, design challenges and future directions. *J. of NeuroEngineering and Rehabilitation*, 6(1):21.

[Hildebrandt et al., 2013] Hildebrandt, M., Albiez, J., Fritsche, M., Hilljegerdes, J., Kloss, P., Wirtz, M., and Kirchner, F. (2013). Design of an autonomous under-ice exploration system. In *2013 OCEANS - San Diego*, pages 1–6.

[Hopkins et al., 2015] Hopkins, M. A., Ressler, S. A., Lahr, D. F., Leonessa, A., and Hong, D. W. (2015). Embedded joint-space control of a series elastic humanoid. In *2015 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 3358–3365.

[Huang et al., 2016] Huang, J., Tu, X., and He, J. (2016). Design and evaluation of the rupert wearable upper extremity exoskeleton robot for clinical and inhome therapies. *IEEE Transactions on Systems, Man, and Cybernetics: Systems*, 46(7):926–935.

[Hubicki et al., 2016] Hubicki, C., Grimes, J., Jones, M., Renjewski, D., Spröwitz, A., Abate, A., and Hurst, J. (2016). Atrias: Design and validation of a tether-free 3d-capable spring-mass bipedal robot. *The International Journal of Robotics Research*, 35(12):1497–1521.

[Husty, 1996] Husty, M. (1996). An algorithm for solving the direct kinematics of general stewart-gough platforms. *Mechanism and Machine Theory*, 31(4):365 – 379.

[Husty, 2017a] Husty, M. L. (2017a). Algebraic geometry: The basics. Lecture Notes, Singularity Summer School.

[Husty, 2017b] Husty, M. L. (2017b). Analysis of parallel manipulators using algebraic tools. Lecture Notes, Singularity Summer School.

[Husty et al., 2007] Husty, M. L., Pfurner, M., Schröcker, H.-P., and Brunnthaler, K. (2007). Algebraic methods in mechanism analysis and synthesis. *Robotica*, 25(6):661–675.

[Husty and Schröcker, 2011] Husty, M. L. and Schröcker, H.-P. (2011). A proposal for a new definition of the degree of freedom of a mechanism. In Kecskeméthy, A., Potkonjak, V., and Müller, A., editors, *Interdisciplinary Applications of Kinematics*, pages 109–117, Dordrecht. Springer Netherlands.

[Husty and Schröcker, 2013] Husty, M. L. and Schröcker, H.-P. (2013). *21st Century Kinematics*, chapter Kinematics and Algebraic Geometry, pages 85–123. Springer-Verlag London, London.

[iMatix, 2017] iMatix (2017). ZeroMQ distributed messaging. `http://zeromq.org/`. Accessed: 18.5.2018.

[Jain, 2011a] Jain, A. (2011a). Graph theoretic foundations of multibody dynamics. *Multibody System Dynamics*, 26(3):307–333.

[Jain, 2011b] Jain, A. (2011b). *Robot and Multibody Dynamics: Analysis and Algorithms*. Springer Verlag.

[Jäntsch et al., 2010] Jäntsch, M., Wittmeier, S., and Knoll, A. (2010). Distributed control for an anthropomimetic robot. In *2010 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 5466–5471.

[Joshi and Tsai, 2002] Joshi, S. and Tsai, L.-W. (2002). Jacobian analysis of limited-dof parallel manipulators. *Journal of Mechanical Design - J MECH DESIGN*, 124.

[Joyeux, 2010] Joyeux, S. (2010). Rock: the robot construction kit. `https://www.rock-robotics.org/stable/`. Accessed: 10.05.2018.

[Joyeux and Albiez, 2011] Joyeux, S. and Albiez, J. (2011). Robot development: from components to systems. In *6th National Conference on Control Architectures of Robots, Grenoble, France*.

[Kempe, 1875] Kempe, A. B. (1875). On a General Method of describing Plane Curves of the nth degree by Linkwork. *Proceedings of the London Mathematical Society*, s1-7(1):213–216.

[Kempe, 1877] Kempe, A. B. (1877). *How to Draw a Straight Line*.

[Khalil and Dombre, 2002a] Khalil, W. and Dombre, E. (2002a). Chapter 4 - inverse geometric model of serial robots. In *Modeling, Identification and Control of Robots*, pages 57 – 84. Butterworth-Heinemann, Oxford.

[Khalil and Dombre, 2002b] Khalil, W. and Dombre, E. (2002b). *Modeling, Identification and Control of Robots*. Taylor & Francis, Inc., Bristol, PA, USA, 3rd edition.

[Khalil and Kleinfinger, 1986] Khalil, W. and Kleinfinger, J. (1986). A new geometric notation for open and closed-loop robots. In *Proceedings. 1986 IEEE International Conference on Robotics and Automation*, volume 3, pages 1174–1179.

[Khosravi and Taghirad, 2014] Khosravi, M. A. and Taghirad, H. D. (2014). Robust pid control of fully-constrained cable driven parallel robots. *Mechatronics*, 24(2):87 – 97.

[Kim and Deshpande, 2017] Kim, B. and Deshpande, A. D. (2017). An upper-body rehabilitation exoskeleton harmony with an anatomical shoulder mechanism: Design, modeling, control, and performance evaluation. *The International Journal of Robotics Research*, 36(4):414–435.

[Kinova, 2018] Kinova (2018). Kinovatmactuator. `https://www.kinovarobotics.com/en/products/actuator-series`.

[Kirchner et al., 2016] Kirchner, E. A., Will, N., Simnofske, M., Benitez, L. M. V., Bongardt, B., Krell, M. M., Kumar, S., Mallwitz, M., Seeland, A., Tabie, M., Woehrle, H., Yueksel, M., Hess, A., Buschfort, R., and Kirchner, F. (2016). Recupera-reha: Exoskeleton technology with integrated biosignal analysis for sensorimotor rehabilitation. In *Transdisziplinaere Konferenz SmartASSIST*, pages 504–517.

[Kong and Gosselin, 2007] Kong, X. and Gosselin, C. M. (2007). *Type Synthesis of Parallel Mechanisms*. Springer Publishing Company, Incorporated, 1st edition.

[Kuehn et al., 2014] Kuehn, D., Bernhard, F., Burchardt, A., Schilling, M., Stark, T., Zenzes, M., and Kirchner, F. (2014). Distributed computation in a quadrupedal robotic system. *International Journal of Advanced Robotic Systems*, 11(7):110.

[Kuehn et al., 2018] Kuehn, D., Dettmann, A., and Kirchner, F. (2018). Analysis of using an active artificial spine in a quadruped robot. In *2018 4th International Conference on Control, Automation and Robotics (ICCAR). International Conference on Control, Automation and Robotics (ICCAR-2018), April 20-23, Auckland, New Zealand*. IEEE Xplore online.

[Kuehn et al., 2016] Kuehn, D., Schilling, M., Stark, T., Zenzes, M., and Kirchner, F. (2016). System design and field testing of the hominid robot charlie. *Journal of Field Robotics*, 34(Issue 4):666–703.

[Kumar et al., 2016] Kumar, S., Bongardt, B., Simnofske, M., and Kirchner, F. (2016). Task space controller for the novel active ankle mechanism. In *International Conference on Robotics and Automation for Humanitarian Applications, December 18-20, Amritapuri, India*, RAHA 2016 Poster Proceedings, page 22.

[Kumar et al., 2018a] Kumar, S., Bongardt, B., Simnofske, M., and Kirchner, F. (2018a). Design and kinematic analysis of the novel almost spherical parallel mechanism active ankle. *Journal of Intelligent & Robotic Systems*.

[Kumar et al., 2019a] Kumar, S., Martensen, J., Mueller, A., and Kirchner, F. (2019a). Model simplification in dynamic control of series-parallel hybrid robots – a representative study of the neglected dynamics. In *2019 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages xx–xx.

[Kumar and Mueller, 2019] Kumar, S. and Mueller, A. (2019). An Analytical and Modular Software Workbench for Solving Kinematics and Dynamics of Series-Parallel Hybrid Robots. volume 5A: 43rd Mechanisms and Robotics Conference of *International Design Engineering Technical Conferences and Computers and Information in Engineering Conference*. V05AT07A054.

[Kumar et al., 2018b] Kumar, S., Nayak, A., Bongardt, B., Mueller, A., and Kirchner, F. (2018b). Kinematic analysis of active ankle using computational algebraic geometry. In Zeghloul, S., Romdhane, L., and Laribi, M. A., editors, *Computational Kinematics*, pages 117–125, Cham. Springer International Publishing.

[Kumar et al., 2018c] Kumar, S., Nayak, A., Peters, H., Schulz, C., Mueller, A., and Kirchner, F. (2018c). Kinematic analysis of a novel parallel 2SPRR+1U ankle mechanism in humanoid robot. In Carricato, M., editor, *Advances in Robot Kinematics*, Cham. Springer Verlag GmbH.

[Kumar et al., 2017a] Kumar, S., Simnofske, M., Bongardt, B., Mueller, A., and Kirchner, F. (2017a). Integrating mimic joints into dynamics algorithms – exemplified by the hybrid recupera exoskeleton. In *Advances In Robotics (AIR-2017), June 28 - July 2, New Delhi, India*. ACM-ICPS.

[Kumar et al., 2017b] Kumar, S., Simnofske, M., Bongardt, B., Müller, A., and Kirchner, F. (2017b). Integrating mimic joints into dynamics algorithms: Exemplified by the hybrid recupera exoskeleton. In *Proceedings of the Advances in Robotics*, AIR '17, pages 27:1–27:6, New York, NY, USA. ACM.

[Kumar et al., 2018d] Kumar, S., von Szadkowski, K. A., Mueller, A., and Kirchner, F. (2018d). HyRoDyn: A modular software framework for solving analytical kinematics and dynamics of series-parallel hybrid robots. In *International Conference on Intelligent Robots and Systems*, IROS Poster.

[Kumar et al., 2019b] Kumar, S., Wöhrle, H., Trampler, M., Simnofske, M., Peters, H., Mallwitz, M., Kirchner, E. A., and Kirchner, F. (2019b). Modular design and decentralized control of the recupera exoskeleton for stroke rehabilitation. *Applied Sciences*, 9(4).

[Kumar et al., 2019c] Kumar, S., Wöhrle, H., de Gea Fernandez, J., Mueller, A., and Kirchner, F. (2019c). A survey on modularity and distributivity in series-parallel hybrid robots. *Mechatronics*. under review.

[Kuo and Dai, 2013] Kuo, C.-H. and Dai, J. S. (2013). Task-oriented structure synthesis of a class of parallel manipulators using motion constraint generator. *Mechanism and Machine Theory*, 70:394 – 406.

[Lahr et al., 2016] Lahr, D. F., Yi, H., and Hong, D. W. (2016). Biologically inspired design of a parallel actuated humanoid robot. *Advanced Robotics*, 30(2):109–118.

[Lassila et al., 2014] Lassila, T., Manzoni, A., Quarteroni, A., and Rozza, G. (2014). *Model Order Reduction in Fluid Dynamics: Challenges and Perspectives*, pages 235–273. Springer International Publishing, Cham.

[Lee et al., 2014] Lee, B., Knabe, C., Orekhov, V., and Hong, D. (2014). Design of a human-like range of motion hip joint for humanoid robots.

[Lee, 2014] Lee, B. K. T.-S. (2014). *Design of a humanoid robot for disaster response*. PhD thesis, Virginia Tech.

[Lemburg et al., 2011] Lemburg, J., de Gea Fernández, J., Eich, M., Mronga, D., Kampmann, P., Vogt, A., Aggarwal, A., Shi, Y., and Kirchner, F. (2011). Aila - design of an autonomous mobile dual-arm robot. In *2011 IEEE International Conference on Robotics and Automation (ICRA)*, pages 5147–5153.

[Lesser, 1992] Lesser, M. (1992). A geometrical interpretation of kane's equations. *Proceedings: Mathematical and Physical Sciences*, 436(1896):69–87.

[Li et al., 2018] Li, Z., Schicho, J., and Schröcker, H.-P. (2018). Kempe's universality theorem for rational space curves. *Foundations of Computational Mathematics*, 18(2):509–536.

[Liu et al., 2001] Liu, G. F., Cheng, H., Xiong, Z. H., Wu, X. Z., Wu, Y. L., and Li, Z. X. (2001). Distribution of singularity and optimal control of redundant parallel manipulators. In *Proceedings 2001 IEEE/RSJ International Conference on Intelligent Robots and Systems. Expanding the Societal Role of Robotics in the the Next Millennium (Cat. No.01CH37180)*, volume 1, pages 177–182 vol.1.

[Lohmeier, 2010] Lohmeier, S. (2010). *Design and Realization of a Humanoid Robot for Fast and Autonomous Bipedal Locomotion*. PhD thesis, Technischen Universität München.

[Lohmeier et al., 2006] Lohmeier, S., Buschmann, T., Ulbrich, H., and Pfeiffer, F. (2006). Modular joint design for performance enhanced humanoid robot lola. In *Proceedings 2006 IEEE International Conference on Robotics and Automation, 2006. ICRA 2006.*, pages 88–93.

[Luh JS, 1980] Luh JS, Walker MW, P. R. (1980). On-line computational scheme for mechanical manipulators. *ASME. J. Dyn. Sys., Meas., Control.*

[Luzi et al., 2018] Luzi, L., Sancisi, N., and Parenti Castelli, V. (2018). A new direct position analysis solution for an over-constrained gough-stewart platform. In Zeghloul, S., Romdhane, L., and Laribi, M. A., editors, *Computational Kinematics*, pages 585–592, Cham. Springer International Publishing.

[Lynch and Park, 2017] Lynch, K. M. and Park, F. C. (2017). *Modern Robotics: Mechanics, Planning, and Control*. Cambridge University Press, New York, NY, USA, 1st edition.

[Mahnke et al., 2009] Mahnke, W., Leitner, S.-H., and Damm, M. (2009). *OPC unified architecture*. Springer Science & Business Media.

[Maruyama et al., 2016] Maruyama, Y., Kato, S., and Azumi, T. (2016). Exploring the performance of ros2. In *2016 International Conference on Embedded Software (EMSOFT)*, pages 1–10.

[McCarthy, 2013] McCarthy, J. M. (2013). Polynomials, computers, and kinematics for the 21st century. In McCarthy, J. M., editor, *21st Century Kinematics*, pages 1–12, London. Springer London.

[Merlet, 1988] Merlet, J. . (1988). Force-feedback control of parallel manipulators. In *Proceedings. 1988 IEEE International Conference on Robotics and Automation*, pages 1484–1489 vol.3.

[Merlet, 2006] Merlet, J. (2006). *Parallel Robots*. Springer Netherlands.

[Merlet, 1999] Merlet, J. P. (1999). Parallel robots: Open problems. *9th International Symposium on Robotics Research*.

[Merlet, 2000] Merlet, J.-P. (2000). Parallel robots: Open problems. In Hollerbach, J. M. and Koditschek, D. E., editors, *Robotics Research*, pages 27–32, London. Springer London.

[Metta et al., 2006] Metta, G., Fitzpatrick, P., and Natale, L. (2006). Yarp: Yet another robot platform. *International Journal of Advanced Robotic Systems*, 3(1):8.

[Mirza et al., 2017] Mirza, M. A., Li, S., and Jin, L. (2017). Simultaneous learning and control of parallel stewart platforms with unknown parameters. *Neurocomputing*, 266:114 – 122.

[Mohamed et al., 2009] Mohamed, N., Al-Jaroodi, J., and Jawhar, I. (2009). A review of middleware for networked robots. *International Journal of Computer Science and Network Security*, 9(5):139–148.

[Mueller, 2014] Mueller, A. (2014). Implementation of a geometric constraint regularization for multibody system models. *Arch. Mech. Eng.*

[Muller, 2005] Muller, A. (2005). Internal preload control of redundantly actuated parallel manipulators—its application to backlash avoiding control. *IEEE Transactions on Robotics*, 21(4):668–677.

[Müller, 2017] Müller, A. (2017). Screw and lie group theory in multibody kinematics. *Multibody System Dynamics*.

[Müller, 2018] Müller, A. (2018). Screw and lie group theory in multibody dynamics. *Multibody System Dynamics*, 42(2):219–248.

[Müller, 2019] Müller, A. (2019). *Local Investigation of Mobility and Singularities of Linkages*, pages 181–229. Springer International Publishing, Cham.

[Murray et al., 1994] Murray, R. M., Sastry, S. S., and Zexiang, L. (1994). *A Mathematical Introduction to Robotic Manipulation*. CRC Press, Inc., Boca Raton, FL, USA, 1st edition.

[Müller, 2013] Müller, A. (2013). On the terminology and geometric aspects of redundant parallel manipulators. *Robotica*, 31(1):137–147.

[Müller, 2018] Müller, A. (2018). Kinematic topology and constraints of multi-loop linkages. *Robotica*, 36(11):1641–1663.

[Nakahara, 2003] Nakahara, M. (2003). *Geometry, topology and physics*. CRC Press.

[Nayak et al., 2018] Nayak, A., Caro, S., and Wenger, P. (2018). Kinematic analysis of the 3-rps-3-spr series–parallel manipulator. *Robotica*, page 1–27.

[Nef et al., 2009] Nef, T., Guidali, M., Klamroth-Marganska, V., and Riener, R. (2009). Armin - exoskeleton robot for stroke rehabilitation. In Dössel, O. and Schlegel, W. C., editors, *World Congress on Medical Physics and Biomedical Engineering, September 7 - 12, 2009, Munich, Germany*, pages 127–130, Berlin, Heidelberg. Springer Berlin Heidelberg.

[Negrello et al., 2015] Negrello, F., Garabini, M., Catalano, M. G., Malzahn, J., Caldwell, D. G., Bicchi, A., and Tsagarakis, N. G. (2015). A modular compliant actuator for emerging high performance and fall-resilient humanoids. In *2015 IEEE-RAS 15th International Conference on Humanoid Robots (Humanoids)*, pages 414–420.

[Nejadfard et al., 2018] Nejadfard, A., Schütz, S., Vonwirth, P., Mianowski, K., and Karsten, B. (2018). Moment arm analysis of the biarticular actuators in compliant robotic leg CARL. In *Conference on Biomimetic and Biohybrid Systems*, pages 348–360. Springer, Springer International Publishing.

[Nielsen and Roth, 1999] Nielsen, J. and Roth, B. (1999). On the kinematic analysis of robotic mechanisms. *The International Journal of Robotics Research*, 18(12):1147–1160.

[Niyetkaliyev and Shintemirov, 2014] Niyetkaliyev, A. and Shintemirov, A. (2014). An approach for obtaining unique kinematic solutions of a spherical parallel manipulator. In *IEEE/ASME International Conference on Advanced Intelligent Mechatronics (AIM)*, pages 1355–1360.

[Open Source Robotics Foundation (OSRF), 2016] Open Source Robotics Foundation (OSRF) (2016). ROS2. `https://github.com/ros2/`.

[Paccot et al., 2009] Paccot, F., Andreff, N., and Martinet, P. (2009). A review on the dynamic control of parallel kinematic machines: Theory and experiments. *I. J. Robotics Res.*, 28(3):395–416.

[Pagis et al., 2015] Pagis, G., Bouton, N., Briot, S., and Martinet, P. (2015). Enlarging parallel robot workspace through type-2 singularity crossing. *Control Engineering Practice*, 39:1 – 11.

[Paine et al., 2015] Paine, N., Mehling, J. S., Holley, J., Radford, N. A., Johnson, G., Fok, C.-L., and Sentis, L. (2015). Actuator control for the nasa-jsc valkyrie humanoid robot: A decoupled dynamics approach for torque control of series elastic robots. *Journal of Field Robotics*, 32(3):378–396.

[Pardo-Castellote, 2003] Pardo-Castellote, G. (2003). OMG data-distribution service: architectural overview. In *23rd International Conference on Distributed Computing Systems Workshops, 2003. Proceedings.*, pages 200–206.

[Parmiggiani et al., 2017] Parmiggiani, A., Fiorio, L., Scalzo, A., Sureshbabu, A. V., Randazzo, M., Maggiali, M., Pattacini, U., Lehmann, H., Tikhanoff, V., Domenichelli, D., Cardellino, A., Congiu, P., Pagnin, A., Cingolani, R., Natale, L., and Metta, G. (2017). The design and validation of the r1 personal humanoid. In *2017 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 674–680.

[Peters et al., 2017] Peters, H., Kampmann, P., and Simnofske, M. (2017). Konstruktion eines zweibeinigen humanoiden roboters. In *2. VDI Fachkonferenz Humanoide Roboter*.

[Piedboeuf, 1993] Piedboeuf, J. C. (1993). Kane's equations or jourdain's principle? In *Proceedings of 36th Midwest Symposium on Circuits and Systems*, pages 1471–1474 vol.2.

[Plücker, 1865] Plücker, J. (1865). Xvii. on a new geometry of space. *Philosophical Transactions of the Royal Society of London*, 155:725–791.

[PROFIBUS & PROFINET International (PI), a] PROFIBUS & PROFINET International (PI). PROFIBUS. `https://www.profibus.com/technology/profibus/`. Accessed: 6.9.2018.

[PROFIBUS & PROFINET International (PI), b] PROFIBUS & PROFINET International (PI). PROFINET the leading industrial ethernet standard. `https://www.profibus.com/technology/profinet/`. Accessed: 6.9.2018.

[Q-RoCK, 2019] Q-RoCK (2019). AI-based Qualification of Deliberative Behaviour for a Robotic Construction Kit . `https://robotik.dfki-bremen.de/en/research/projects/q-rock.html`. Accessed: 10.05.2019.

[Quigley et al., 2009] Quigley, M., Conley, K., Gerkey, B. P., Faust, J., Foote, T., Leibs, J., Wheeler, R., and Ng, A. Y. (2009). Ros: an open-source robot operating system. In *ICRA Workshop on Open Source Software*.

[Rader et al., 2017] Rader, S., Kaul, L., Weiner, P., and Asfour, T. (2017). Highly integrated sensor-actuator-controller units for modular robot design. In *2017 IEEE International Conference on Advanced Intelligent Mechatronics (AIM)*, pages 1160–1166.

[Radford et al., 2015] Radford, N. A., Strawser, P., Hambuchen, K., Mehling, J. S., Verdeyen, W. K., Donnan, A. S., Holley, J., Sanchez, J., Nguyen, V., Bridgwater, L., Berka, R., Ambrose, R., Myles Markee, M., Fraser-Chanpong, N. J., McQuin, C., Yamokoski, J. D., Hart, S., Guo, R., Parsons, A., Wightman, B., Dinh, P., Ames, B., Blakely, C., Edmondson, C., Sommers, B., Rea, R., Tobler, C., Bibby, H., Howard, B., Niu, L., Lee, A., Conover, M., Truong, L., Reed, R., Chesney, D., Platt Jr, R., Johnson, G., Fok, C.-L., Paine, N., Sentis, L., Cousineau, E., Sinnet, R., Lack, J., Powell, M., Morris, B., Ames, A., and Akinyode, J. (2015). Valkyrie: Nasa's first bipedal humanoid robot. *Journal of Field Robotics*, 32(3):397–419.

[Raghavan and Roth, 1993] Raghavan, M. and Roth, B. (1993). Inverse kinematics of the general 6r manipulator and related linkages. *Journal of Mechanical Design*, 115(3):502–508.

[Rajeevlochana et al., 2012] Rajeevlochana, C., Saha, S., and Kumar, S. (2012). Automatic extraction of dh parameters of serial manipulators using line geometry. In *The 2nd International Conference on Multibody System Dynamics*.

[Rao et al., 2006] Rao, A. K., Saha, S., and Rao, P. (2006). Dynamics modelling of hexaslides using the decoupled natural orthogonal complement matrices. *Multibody System Dynamics*, 15(2):159–180.

[Recupera-Reha, 2018] Recupera-Reha (2018). Recupera REHA: Full-body exoskeleton for upper body robotic assistance. `https://robotik.dfki-bremen.de/en/research/projects/recupera-reha.html`. Accessed: 10.05.2018.

[Reichardt et al., 2012] Reichardt, M., Föhst, T., and Berns, K. (2012). Introducing finroc: A convenient real-time framework for robotics based on a systematic design approach. Technical report, Robotics Research Lab, Department of Computer Science, University of Kaiserslautern, Kaiserslautern, Germany.

[Reichardt et al., 2017] Reichardt, M., Schütz, S., and Berns, K. (2017). One fits more–on highly modular quality-driven design of robotic frameworks and middleware. *Journal of Software Engineering for Robotics*, 8(1):141–153.

[RobMoSys, 2018] RobMoSys (2018). RobMoSys: Composable Models and Software for Robotic Systems. `https://robmosys.eu/`. Accessed: 10.05.2018.

[Robotis, a] Robotis. Dynamixel pro series. `http://www.robotis.us/dynamixel-pro`. Accessed: 22.08.2018.

[Robotis, b] Robotis. Dynamixel series. `http://www.robotis.us/dynamixel`. Accessed: 22.08.2018.

[Ronacher, 2017] Ronacher, A. (2017). FLASK, web development one drop at a time. http://flask.pocoo.org/.

[Rouillier, 2004] Rouillier, F. (2004). Efficient isolation of polynomial's real roots. *Journal of Computational and Applied Mathematics*, 162(1):33 – 50.

[Rowland, 2019] Rowland, T. (2019). `Algebraic Geometry`. From MathWorld–A Wolfram Web Resource, created by Eric W. Weisstein. `http://mathworld.wolfram.com/AlgebraicGeometry.html`. Accessed: 10.03.2019.

[Sadeqi et al., 2017] Sadeqi, S., Bourgeois, S. P., Park, E. J., and Arzanpour, S. (2017). Design and performance analysis of a 3-rrr spherical parallel manipulator for hip exoskeleton applications. *Journal of Rehabilitation and Assistive Technologies Engineering*, 4:2055668317697596.

[Sawicki and Ferris, 2009] Sawicki, G. S. and Ferris, D. P. (2009). A pneumatically powered knee-ankle-foot orthosis (kafo) with myoelectric activation and inhibition. *J. of NeuroEngineering and Rehabilitation*, 6(1):23.

[Schaeffer et al., 2007] Schaeffer, A. A., Haddadin, S., Ott, C., Stemmer, A., Wimböck, T., and Hirzinger, G. (2007). The dlr lightweight robot: design and control concepts for robots in human environments. *Industrial Robot: An International Journal*, 34(5):376–385.

[Schlesselman et al., 2004] Schlesselman, J. M., Pardo-Castellote, G., and Farabaugh, B. (2004). OMG data-distribution service (DDS): architectural update. In *Military Communications Conference*, volume 2, pages 961–967. IEEE.

[Schütz et al., 2017] Schütz, S., Nejadfard, A., Mianowski, K., Vonwirth, P., and Berns, K. (2017). CARL – A compliant robotic leg featuring mono- and biarticular actuation. In *IEEE-RAS International Conference on Humanoid Robots*.

[Selig, 2005] Selig, J. (2005). *Introduction*, pages 1–9. Springer New York, New York, NY.

[Sercos International e. V., 2005] Sercos International e. V. (2005). SERCOS - the automation bus. `https://www.sercos.org`. Accessed: 2018-09-06.

[Serracín et al., 2012] Serracín, J., Puglisi, L., Saltaren, R., Ejarque, G., Sabater-Navarro, J., and Aracil, R. (2012). Kinematic analysis of a novel 2-d.o.f. orientation device. *Robotics and Autonomous Systems*, 60(6):852 – 861.

[Shang and Cong, 2014] Shang, W. and Cong, S. (2014). Robust nonlinear control of a planar 2-dof parallel manipulator with redundant actuation. *Robotics and Computer-Integrated Manufacturing*, 30(6):597 – 604.

[Shang et al., 2012] Shang, W.-W., Cong, S., and Ge, Y. (2012). Adaptive computed torque control for a parallel manipulator with redundant actuation. *Robotica*, 30(3):457–466.

[Shang et al., 2010] Shang, W.-w., Cong, S., and Jiang, S.-l. (2010). Dynamic model based nonlinear tracking control of a planar parallel manipulator. *Nonlinear Dynamics*, 60(4):597–606.

[Simnofske, 2015] Simnofske, M. (2015). Ausrichtungsvorrichtung zum Ausrichten einer Plattform in drei rotatorischen Freiheiten. Patent application, DE102013018034A1.

[Simnofske et al., 2016] Simnofske, M., Kumar, S., Bongardt, B., and Kirchner, F. (2016). Active ankle - an almost-spherical parallel mechanism. In *47th International Symposium on Robotics (ISR)*.

[Sonsalla et al., 2017] Sonsalla, R., Hanff, H., Schöberl, P., Stark, T., and Mulsow, N. A. (2017). Dfki-x: A novel, compact and highly integrated robotics joint for space applications. In *Proceedings of the 17th European Space Mechanisms and Tribology Symposium. European Space Mechanisms and Tibology Symposium (ESMATS-2017), 17th, September 20-22, Hatfield, United Kingdom*. ESMATS.

[Stasse and Flayols, 2019] Stasse, O. and Flayols, T. (2019). *An Overview of Humanoid Robots Technologies*, pages 281–310. Springer International Publishing, Cham.

[Stasse et al., 2017] Stasse, O., Flayols, T., Budhiraja, R., Giraud-Esclasse, K., Carpentier, J., Mirabel, J., Prete, A. D., Souères, P., Mansard, N., Lamiraux, F., Laumond, J. P., Marchionni, L., Tome, H., and Ferro, F. (2017). Talos: A new humanoid research platform targeted for industrial applications. In *2017 IEEE-RAS 17th International Conference on Humanoid Robotics (Humanoids)*, pages 689–695.

[Stauffer et al., 1977] Stauffer, R. N., Chao, E. Y. S., and Brewster, R. C. (1977). Force and motion analysis of the normal, diseased, and prosthetic ankle joint. *Clinical Orthopaedics and Related Research*, 127(127):189–96.

[Stewart, 1965] Stewart, D. (1965). A platform with six degrees of freedom. *Proceedings of the Institution of Mechanical Engineers*, 180(1):371–386.

[Stoeffler et al., 2018] Stoeffler, C. et al. (2018). Master thesis: Conceptual design of a variable stiffness mechanism using parallel redundant actuation.

[Stoeffler et al., 2018] Stoeffler, C., Kumar, S., Peters, H., Brüls, O., Müller, A., and Kirchner, F. (2018). Conceptual design of a variable stiffness mechanism in a humanoid ankle using parallel redundant actuation. In *2018 IEEE-RAS 18th International Conference on Humanoid Robots (Humanoids)*, pages 462–468.

[Su et al., 2004] Su, Y., Duan, B., and Zheng, C. (2004). Nonlinear pid control of a six-dof parallel manipulator. *IEE Proceedings - Control Theory and Applications*, 151:95–102(7).

[Sureshbabu et al., 2017] Sureshbabu, A. V., Chang, J. H., Fiorio, L., Scalzo, A., Metta, G., and Parmiggiani, A. (2017). A parallel kinematic wrist for the r1 humanoid robot. In *2017 IEEE International Conference on Advanced Intelligent Mechatronics (AIM)*, pages 1215–1220.

[Temple, 1988] Temple, R. K. G. (1988). The cardan suspension. *The UNESCO Courier*.

[Thomas and Ros, 2005] Thomas, F. and Ros, L. (2005). Revisiting trilateration for robot localization. *IEEE Transactions on Robotics*, 21(1):93–101.

[To and Webb, 2012] To, M. and Webb, P. (2012). An improved kinematic model for calibration of serial robots having closed-chain mechanisms. *Robotica*, 30(6):963–971.

[TREE, 2018] TREE (2018). Technology for Robotic systems Entering real Environments (TREE) Actuators. `https://www.treerobotics.eu/actuators/tree-robotics-actuators-info-sheet/file`. Accessed: 10.09.2018.

[Valasek et al., 2010] Valasek, M., Zicha, J., Karasek, M., and Hudec, R. (2010). Hexasphere – redundantly actuated parallel spherical mechanism as a new concept of agile telescope. *Advances in Astronomy*.

[Villgrattner et al., 2011] Villgrattner, T., Schneider, E., Andersch, P., and Ulbrich, H. (2011). Compact high dynamic 3 dof camera orientation system: Development and control. *Journal of System Design and Dynamics*, 5(5):819–828.

[Vischer and Clavel, 2000] Vischer, P. and Clavel, R. (2000). Argos: A novel 3-dof parallel wrist mechanism. *The International Journal of Robotics Research*, 19(1):5–11.

[von Szadkowski and Langosz, 2015] von Szadkowski, K. A. and Langosz, M. (2015). Phobos: 3d robot modelling made easy. Technical Report 1406, DFKI.

[Vonwirth, 2017] Vonwirth, P. (2017). Modular control architecture for bipedal walking on a single compliant leg. Master's thesis, Robotics Research Lab, University of Kaiserslautern. unpublished; supervised by Steffen Schütz.

[Wampler and Sommese, 2013] Wampler, C. W. and Sommese, A. J. (2013). Applying numerical algebraic geometry to kinematics. In McCarthy, J. M., editor, *21st Century Kinematics*, pages 125–159, London. Springer London.

[Wang and Artemiadis, 2013] Wang, Y. and Artemiadis, P. (2013). Closed-form inverse kinematic solution for anthropomorphic motion in redundant robot arms. *Advances in Robotics and Automation*.

[Wöhrle et al., 2017] Wöhrle, H., Tabie, M., Kim, S. K., Kirchner, F., and Kirchner, E. A. (2017). A hybrid fpga-based system for eeg- and emg-based online movement prediction. *Sensors*, 17(7).

[Wu et al., 2015] Wu, G., Caro, S., and Wang, J. (2015). Design and transmission analysis of an asymmetrical spherical parallel manipulator. *Mechanism and Machine Theory*, 94:119 – 131.

[Wu and Tiso, 2016] Wu, L. and Tiso, P. (2016). Nonlinear model order reduction for flexible multibody dynamics: a modal derivatives approach. *Multibody System Dynamics*, 36(4):405–425.

[Yamamoto et al., 2005] Yamamoto, S., Hagiwara, A., Mizobe, T., Yokoyama, O., and Yasui, T. (2005). Development of an ankle–foot orthosis with an oil damper. *Prosthetics and Orthotics Int.*, 29(3).

[Zamalloa et al., 2018] Zamalloa, I., Muguruza, I., Hernández, A., Kojcev, R., and Mayoral, V. (2018). An information model for modular robots: the hardware robot information model (hrim). *arXiv preprint arXiv:1802.01459*.

[Zenzes et al., 2016] Zenzes, M., Kampmann, P., Stark, T., and Schilling, M. (2016). Ndlcom: Simple protocol for heterogeneous embedded communication networks. In *Proceedings of the Embedded World Exhibition & Conference, Nuremberg, Germany*, pages 23–25.

[Zhang et al., 2012] Zhang, L., Slaets, P., and Bruyninckx, H. (2012). An open embedded hardware and software architecture applied to industrial robot control. In *2012 IEEE International Conference on Mechatronics and Automation*, pages 1822–1828.

[Zi et al., 2008] Zi, B., Duan, B., Du, J., and Bao, H. (2008). Dynamic modeling and active control of a cable-suspended parallel robot. *Mechatronics*, 18(1):1 – 12.

[Zoss et al., 2006] Zoss, A. B., Kazerooni, H., and Chu, A. (2006). Biomechanical design of the berkeley lower extremity exoskeleton (bleex). *IEEE/ASME Transactions on Mechatronics*, 11(2):128–138.