JIŘÍ BLÁHA

Research Methodology for Static Stiffness of Six-Axis Serial Robots

BACHELOR THESIS

Submitted to the

Division of Mechanics and Mechatronics
Dept. of Mechanics, Biomechanics, and Mechatronics
Faculty of Mechanical Engineering
Czech Technical University in Prague

in partial fulfilment of the requirements for the degree of Bachelor of Science in the study programme *Theoretical Fundamentals of Mechanical Engineering*.

Written under the supervision of

Ing. Zdeněk Neusser, Ph.D.
Division of Mechanics and Mechatronics

in Prague and Litvínov, Czech Republic, May 2024.

Thesis Supervisor

Ing. Zdeněk Neusser, Ph.D.
Division of Mechanics and Mechatronics
Dept. of Mechanics, Biomechanics, and Mechatronics
Faculty of Mechanical Engineering
Czech Technical University in Prague
Technická 4
160 00 Prague 6
Czech Republic
zdenek.neusser@fs.cvut.cz

Typeset in the LaTeX typesetting system.
Illustrations drawn using TikZ by Till Tantau.

ZADÁNÍ BAKALÁŘSKÉ PRÁCE

I. OSOBNÍ A STUDIJNÍ ÚDAJE

Příjmení: **Bláha** Jméno: **Jiří** Osobní číslo: **508858**

Fakulta/ústav: **Fakulta strojní**

Zadávající katedra/ústav: **Ústav mechaniky, biomechaniky a mechatroniky**

Studijní program: **Teoretický základ strojního inženýrství**

Studijní obor: **bez oboru**

II. ÚDAJE K BAKALÁŘSKÉ PRÁCI

Název bakalářské práce:

Metodika výzkumu statické tuhosti šestiosých sériových robotů

Název bakalářské práce anglicky:

Research Methodology for Static Stiffness of Six-Axis Serial Robots

Pokyny pro vypracování:

1. Seznamte se s problematikou analýzy tuhosti seriových robotů v prostoru.
2. Proveďte teoretický rozbor řešení kinematiky sériové šestiosé antropomorfní struktury se sférickým zápěstím.
3. Vyberte vhodného šestiosého sériového robota a sestavte jeho dynamický model v prostředí MATLAB-Simscape.
4. Analyzujte statickou tuhost robota založenou na linearizaci dynamického modelu.
5. Porovnejte výsledky tuhosti pro jednoho robota a pro dva propojené roboty.

Seznam doporučené literatury:

- Valášek M., Stejskal V., Březina J.: Mechanika A (lecture notes), Prague, CTU, 2002.
- Valášek M., Bauma V., Šika Z.: Mechanika B (lecture notes), Prague, CTU, 2004.
- Siciliano, B. et al. Robotics: Modelling, Planning, and Control. 1st ed. London, England: Springer, 2018.
- http://www.mathworks.com (on-line documentation)

Jméno a pracoviště vedoucí(ho) bakalářské práce:

Ing. Zdeněk Neusser, Ph.D. odbor mechaniky a mechatroniky FS

Jméno a pracoviště druhé(ho) vedoucí(ho) nebo konzultanta(ky) bakalářské práce:

Datum zadání bakalářské práce: **23.04.2024** Termín odevzdání bakalářské práce: **14.08.2024**

Platnost zadání bakalářské práce: _____

Ing. Zdeněk Neusser, Ph.D. podpis vedoucí(ho) práce	prof. Ing. Michael Valášek, DrSc. podpis vedoucí(ho) ústavu/katedry	doc. Ing. Miroslav Španiel, CSc. podpis děkana(ky)

III. PŘEVZETÍ ZADÁNÍ

Student bere na vědomí, že je povinen vypracovat bakalářskou práci samostatně, bez cizí pomoci, s výjimkou poskytnutých konzultací. Seznam použité literatury, jiných pramenů a jmen konzultantů je třeba uvést v bakalářské práci.

. _____ _____

Datum převzetí zadání Podpis studenta

Statutory Declaration

I hereby declare that this thesis is my own work. Where other sources of information have been used, they have been acknowledged in the bibliography.

I understand that my work relates to the rights and obligations under the Act No. 121/2000 Sb., the Copyright Act, in particular the fact that the Czech Technical University in Prague has the right to conclude a license agreement on the use of this work as a school work pursuant to Section 60, §1 of the Copyright Act.

In Prague on ....

Jiří Bláha

Acknowledgements

First and foremost, it is my sincere pleasure to thank the supervisor of this thesis, Ing. Zdeněk Neusser, Ph.D., for his tremendous support, never-ending patience and courage to explore such a fascinating topic with me.

I also thank Filip Vaško, with whom I share a similar topic and who has been a constant spirit of encouragement in my not always successful endeavours. Having a close friend with whom I could discuss this topic helped me a lot.

Finally, I want to extend boundless gratitude to my family and my partner, Anna. The indispensable financial and emotional support from my parents has been pivotal to my academic pursuits, while Anna's love and care have filled each day with anticipation and joy. It is because of them that I now sit here, penning these acknowledgments, filled with pride for what I have accomplished.

Author Jiří Bláha
Title Research Methodology for Static Stiffness of Six-Axis Serial Robots
Název Metodika výzkumu statické tuhosti šestiosých sériových robotů

Year 2023/2024
Department Dept. of Mechanics, Biomechanics, and Mechatronics
Division Division of Mechanics and Mechatronics
Supervisor Ing. Zdeněk Neusser, Ph.D.

Range 193 pages
74 figures
30 tables

Keywords static stiffness, kinetostatics, open kinematic chain, six-axis robot, anthropomorphic structure, coupled robots
Klíč. slova statická tuhost, kinetostatika, otevřený kinematický řetězec, šestiosý robot, antropomorfní struktura, spojení roboti

Abstract The subject of this bachelor thesis is the development of a research methodology for static stiffness of robots, specifically six-axis serial robots with an anthropomorphic structure. It covers theoretical knowledge from the areas of rigid body mechanics, specifically kinematics and statics, nonlinear systems and mathematics, which together form the theoretical framework of static stiffness research. This theoretical analysis is then applied to a model of a real six-axis robot in the Simscape MBS environment, which is simulated in several of its possible configurations. Finally, stiffness maps are produced and the evolution of static stiffness for a given configuration after a second robot is attached is observed.

Abstrakt Předmětem této bakalářské práce je vývoj metodiky výzkumu statické tuhosti robotů, konkrétně šestiosých sériových robotů s antropomorfní strukturou. Zahrnuje teoretické poznatky z oblasti mechaniky tuhých těles, konkrétně kinematiky a statiky, nelineárních systémů a matematiky, které společně tvoří teoretický rámec výzkumu statické tuhosti. Tato teoretická analýza je pak aplikována na model skutečného šestiosého robota v prostředí Simscape MBS, který je simulován v několika svých možných konfiguracích. Nakonec jsou vytvořeny mapy tuhosti a je sledován vývoj statické tuhosti pro danou konfiguraci po připojení druhého robota.

Dedicated to my parents, Kamila and Petr.

Contents

List of Figures

List of Tables

0
Introduction and Problem Statement

(1) "A robot may not injure a human being or, through inaction, allow a human being to come to harm."

(2) "A robot must obey the orders given it by human beings except where such orders would conflict with the First Law."

(3) "A robot must protect its own existence as long as such protection does not conflict with the First or Second Laws."

In his magnum opus, *I, Robot* (1950), Isaac Asimov once formulated the Three Laws of Robotics listed above.[1] [1] The field of robotics has since come a long way, but due to our perception of robots as something quite ordinary in today's day and age, humanity can often find itself unaware of the existence of these laws. Regardless of the widespread integration of robots and intelligent systems, Asimov's laws should remain at the forefront of our collective consciousness, their significance unquestioned. Indeed, the idea behind the three statements above extends beyond robotics itself, and when one starts looking at them in a broader optic, they explode into the fascinating world of all technology. It is no exaggeration to say that, in the contemporary world, automation has permeated almost every aspect of our lives, providing us with unprecedented tools and capabilities, ushering in an era of possibilities inconceivable to our ancestors. While the proliferation of automation makes some concerned and others excited, it is crucial to recognize that the dystopian futures envisioned in works like Asimov's *I, Robot* or Čapek's *Rossum's Universal Robots* are not an imminent

[1] To be more precise, Asimov's laws of robotics were introduced 8 years prior to the release of his magnum opus in a tale called *Runaround* (1942). This tale was subsequently published as part of *I, Robot* where most of us know them from.

reality. The field of robotics, with its interdisciplinary nature encompassing mechanical engineering, electrical engineering, information technology, and more, is evolving in ways unanticipated by the early pioneers of science fiction.

Robots taking over the world is more of a philosophical question rather than an engineering one. Present-day robotics is a far cry from the dystopian visions of Asimov and Čapek, requiring continuous adaptation to contemporary issues. As we navigate the evolving landscape of technology, it is prudent to remain prepared for unforeseen challenges, ensuring a responsible and mindful integration of robotic systems and artificial intelligence alike into our daily lives.

0.1 Static Stiffness

This thesis is focused on exploring the spatial static stiffness of six-axis serial robots, a critical aspect within the field of robotics, particularly relevant in applications such as robotic machining. The increasing prominence of robotic machining is attributed to its advantages, including a larger work envelope, reduced weight, compactness, and improved cost-effectiveness compared to traditional CNC machining methods. In processes like milling, the tool center point is subjected to significant force loads, typically directed in general directions. This phenomenon often results in adverse effects such as vibrations, leading to deviations in the final product's dimensions and geometry. These deviations can range from functional impairments in the produced part to situations where precision is imperative for safety and functionality. To address these challenges, it is imperative for the robotic system to possess adequate stiffness to withstand applied forces and minimize tool displacements. While achieving infinite stiffness, wherein tool displacement is entirely eliminated, is unattainable in practical applications, efforts can be made to enhance stiffness and consequently improve product quality.

One potential strategy for enhancing overall stiffness characteristics is the *coupling of two robots*, both sharing a common end-effector such as a spindle with a milling cutter. The rationale behind this approach lies in the mutual support offered by the two robots, potentially leading to enhanced stiffness. However, it is crucial to note that such enhancements may be directional, with stiffness improvements observed in certain directions while deterioration may occur in others. Consequently, investigating spatial stiffness presents a significant challenge, yet it is essential for understanding and addressing directional variations.

0.2 State of the Art

The research into robotic stiffness is still undergoing development. Almost every research paper considers a different approach to such problem. In [2], a hybrid-parallel robot structure was developed. Subsequently, the authors used finite

element analysis (FEA) to investigate stiffness properties of said structure and stiffness maps were created. As a result, static stiffness was, at some points of the work envelope, comparable or even better compared to a conventional CNC. Yet, the disadvantage of [2] is the stiffness being evaluated only in the direction of the Cartesian axes, x, y, and z, thus not providing an idea about stiffness in a general direction. In [3], a stiffness adjustable snake-like robot was modelled, along with its gravity compensation. The work [4] investigated the joint stiffness of robots, both analytically and under simulation, by attaching one-degree-of-freedom torsion springs to each joint. The article [5] discusses the stiffness increase using topology optimization of a six-degree-of-freedom industrial robotic arm. This optimization is also based upon finite element analysis. In [6], the mathematical model of the cooperation of multiple six-axis robots with flexible interconnection is developed, while [7] explores the control aspects of such cooperative manipulation, and [8] sets the objective of determining the system's nominal motion. Paper [9] presents a variable stiffness strategy of a structure with a compliant wrist, able to adjust its stiffness continuously by virtue of a super-elastic Ni-Ti (Nickel-Titan) wire. A different approach, using compliance control based on impedance control, is shown within [10]. Finally, the parallel to robotic milling is established in [11], where a stability prediction algorithm was developed, forecasting vibration.

Despite these contributions, existing research does not entirely align with our objectives, necessitating the development of a novel methodology. Notably, Neusser et al. from the Division of Mechatronics, CTU-FME, have addressed the topic of static stiffness acting in general directions. Their ongoing research investigates the static stiffness of a 3R planar mechanism, modeled after the Stäubli RX-60 industrial manipulator, with their findings currently under review. The motivation behind this thesis is to build upon the work of Neusser et al. by expanding their research to encompass a spatial model and the investigation of static stiffness in three-dimensional space.

0.3 Objectives

The thesis outlines three primary objectives,

(1) develop a theoretical framework for spatial static stiffness analysis,

(2) apply this framework onto a suitable six-axis robot,

(3) evaluate the results and compare them with two coupled robots,

which will be elaborated upon within its contents. Objective (1) is covered in Chapters 1,2, and 3, which are housed in Part I. The other two objectives are developed in Chapters 4 and 5, as for Objective (2), and Chapter 6, as for Objective (3), both under the umbrella of Part II.

I
Theoretical Background

1
Kinetostatics of Rigid Systems

Kinematics and statics serve as foundational pillars in shaping the design paradigms and computational methodologies employed in the field of robotics. Kinematics pertains to the meticulous analysis of the isolated movement patterns exhibited by systems, such as robotic mechanisms, while statics is concerned with the equilibrium states of stationary bodies and the forces acting upon them. Despite their individual significance, a comprehensive description of the system's behaviour necessitates a synergistic amalgamation of insights gleaned from both domains. In instances where the primary focus does not lie in the meticulous understanding of the system's kinematics, there may be a tendency to eschew detailed exploration in this realm, and vice versa. However, the omission of either discipline invariably engenders lacunae in understanding, whether it pertains to grasping the intricate equilibrium forces at play or capturing the nuances of the system's motion.

The material contained within this chapter serves mainly as a theoretical introduction to concepts applied in subsequent chapters, and also provides necessary context for readers who might have limited prior knowledge on the subject.[2] For those already acquainted with the subjects addressed in subsequent sections, this chapter can be skipped and revisited as required. The opening section delves into the basics of system modelling and representation, emphasizing key ideas such as degrees of freedom, mechanisms, links, and joints. We then shift our focus to kinematics of solitary rigid bodies, where we introduce the concept of the position vector, the rotation matrix, and subsequently, the homogeneous transformation matrix. With these foundational principles in place, we explore the kinematics of open chains, covering both their forward and inverse kinematics. To conclude, we delve into a specific aspect of statics, namely static gravity compensation, where

[2]For further background on some unproven concepts, the reader is encouraged to consult other resources provided by the author.

we outline the common methodology for recalculating the equilibrium position of robot joints in order to account for gravity.

1.1 Fundamental Concepts of Kinetostatics

Each kinetostatic analysis of a system commences with the development of a model thereof. It is imperative to navigate the delicate equilibrium between simplifying the model and preserving its fidelity. While simplified models expedite computational procedures, their departure from reality may yield suboptimal accuracy. Conversely, complex models, while enhancing realism, entail heightened computational resources. This conundrum presents a nuanced optimization challenge, wherein the balance must be struck between sacrificing accuracy and augmenting computational demands. As a guiding principle, it is customary to designate an acceptable threshold of accuracy and simplify accordingly.

1.1.1 System Representations

The determination of how best to portray a system becomes increasingly apparent as we delve into the significance of the system's dimensions and inertial properties, such as mass, within the model. This critical assessment entails evaluating whether these properties remain discernible from the chosen observation point, which serves as the vantage point for scrutinizing the system. In light of this evaluation, we are presented with two principal options: *point mass* or a *set of point masses*. Each choice carries its own implications and considerations, ultimately shaping the fidelity and accuracy of the model.

- ▷ **Point Mass.** The most trivial representation we can choose for a system is *point mass*, a point-like object devoid of dimensions that encapsulates the entire mass of the system. While utilizing point mass simplifies calculations substantially, the results it produces are often skewed. Additionally, many systems cannot be adequately captured using a point mass representation, necessitating a more sophisticated model, a need that is usually apparent from the outset.
- ▷ **Set of Point Masses.** A natural choice one can make to increase the sophistication of point mass is a *set of point masses* (or system of point masses). If we presume the distances between all point masses in this set, of which there are infinitely many,[3] stay *constant* under all circumstances, we call the set a *rigid body*. Provided we allow these distances to be variable, e.g., under applied stress or by motion itself,[4] the set is then referred to as a *flexible body*. Moreover, it is

[3] From a mathematical standpoint, there are infinitely many point masses in the set but for computation purposes, we need to limit ourselves to a finite number.

[4] For instance, when rotating a flexible disc at high angular velocities, its diameter tends to change due to centrifugal forces, a phenomenon not observed when rotating a rigid disc.

apparent that flexible bodies require significantly more computation power as we need to take their ability to deform into account. For the scope of our objectives, we make do with rigid bodies, meaning we shall discard the idea of flexibility in all following sections, allowing us to simplify our analysis by disregarding the deformable nature of flexible bodies and focus on the behavior and interactions of rigid, non-deformable structures.

Lastly, the precision in depicting the geometry of a rigid or flexible body profoundly impacts the model's accuracy, entwined with the computational complexity we're willing to manage. To enhance computational efficiency, we often simplify topological intricacies in models, retaining essential mass and inertia characteristics. For instance, in a kinematic diagram, a robotic arm may be portrayed as a basic beam, with identical mass and inertia parameters as the more complex real arm (see Subsection 1.1.3 later on).

1.1.2 Mechanism Theory and Grübler's Formula

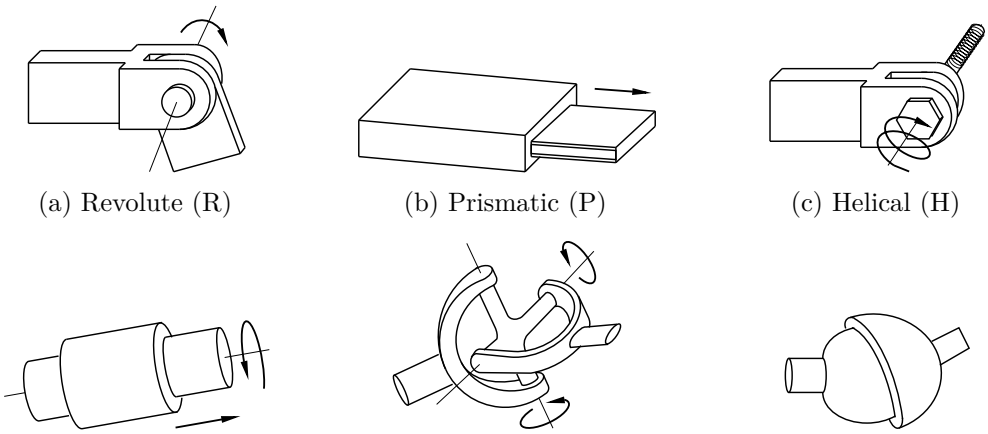
The smallest number of *independent* parameters $\mathscr{D} \in \mathbb{N}_0$ with which we are able to describe the configuration, i.e., the position and orientation, of a system in n-dimensional Euclidean space $\mathbb{E}^n$ is referred to as the number of *degrees of freedom*. A rigid body in three-dimensional Euclidean space $\mathbb{E}^3$ has *six* degrees of freedom as it is able to, when not constrained, move in the direction all three Cartesian axes and also rotate around said axes. Only with all six of these parameters provided, we can state that the configuration of the rigid body is known.

Remark 1.1. A point mass in space has three degrees of freedom due to its presumed infinitesimal size, resulting in its configuration remaining indifferent when rotated about an axis. However, it retains the freedom to move arbitrarily, thereby presenting us with a set of three independent parameters. ◇

Remark 1.2. Following the same rationale, it becomes evident that within a two-dimensional context, an unbounded rigid body exhibits *three* degrees of freedom, while a free point mass has two. ◇

In engineering practice, freely moving bodies are seldom found. More commonly, they are constrained and interconnected to create *mechanisms*, where the individual bodies are referred to as *links*, and the constraints between them are known as *joints* (see Figure 1.1). Further, the two links connected by a joint can be referred to as the *base* link and the *follower* link, with the base link being the one *entering* the joint while the follower link *exits* it. This terminology is heavily dependent on context as when we look at Fig. 1.1, we can hardly distinguish between the base and follower links with all six joint types. Conversely, if we hypothetically designate the larger block in Figure 1.1(b) as stationary, it becomes

evident that it constitutes the base link, while the smaller block is the follower link which performs prismatic motion by "sliding" in and out of the joint.



(a) Revolute (R) (b) Prismatic (P) (c) Helical (H)

(d) Cylindrical (C) (e) Universal (U) (f) Spherical (S)

Figure 1.1 Six most typical robot joints.

We distinguish between two types of mechanisms — open-chain (serial) and closed-chain.[5] Most industrial manipulators and robotic arms are open-chain mechanisms (see Fig. 1.2), delta robots are an example of parallel, closed-chain mechanisms (see Fig. 1.3). It is also common practice to distinguish between the types of open-chain robots by joints they are composed of. For example, a RRRRRR arm, or 6R arm for short, is composed of six revolute joints; the first two joints of a RRP robot[6] are revolute, the last one is prismatic; etc.

		Joint type					
		R	P	H	C	U	S
Space (3D)	Constraints c	5	5	5	4	4	3
	Freedoms f	1	1	1	2	2	3
Plane (2D)	Constraints c	2	2	✗	✗	✗	✗
	Freedoms f	1	1	✗	✗	✗	✗

Table 1.1 Constraints and freedoms provided by predominant robot joints.

Determining the number of degrees of freedom of mechanisms can sometimes prove difficult and, as opposed to solitary rigid bodies, requires more than a quick, logical analysis of the available movement, especially when the mechanism in question is of increased complexity. Let us consider an arbitrary mechanism

[5]Closed-chain mechanisms have at least one closed loop.
[6]The RRP robot is also referred to as the SCARA robot.

composed of $U \in \mathbb{N}$ links and $J \in \mathbb{N}$ joints, each taking away $c \in \mathbb{N}$ degrees of freedom (as per Tab. 1.1), then the total number of degrees of freedom $\mathscr{D} \in \mathbb{N}_0$ of this mechanism is determined by *Grübler's formula* as

$$\mathscr{D} = \widetilde{\mathscr{D}}(U - 1) - \sum_{i=1}^{J} c_j, \tag{1.1}$$

where

$$\widetilde{\mathscr{D}} = \begin{cases} 3 & \text{for planar mechanisms,} \\ 6 & \text{for spatial mechanisms,} \end{cases}$$

is the number of degrees of freedom of a rigid body. Expression (1.1) holds if and only if all joint constraints are *independent*. If not, only a lower bound on the degrees of freedom can be calculated from the formula. [12]

1.1.3 Kinetostatic Topology

As noted earlier, simplifying complex topological details in real systems is common. This reduces computational demands and speeds up result generation. While simplifying geometry is beneficial, preserving accurate inertial characteristics remains a crucial aspect of modelling.

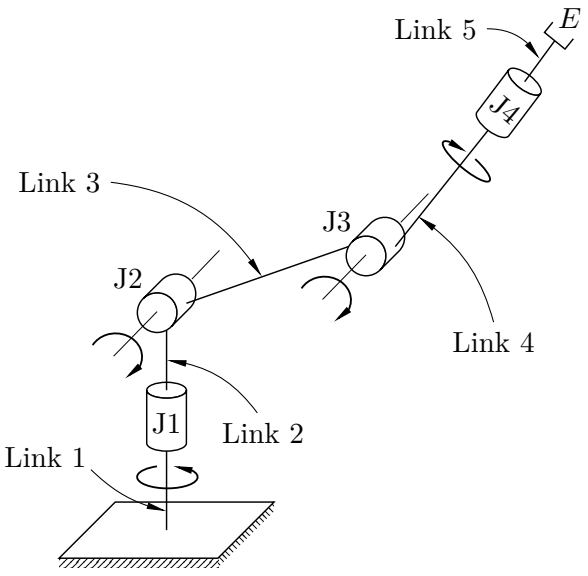


Figure 1.2 Kinematic diagram of a 4R serial manipulator.

Figure 1.2 depicts a 4R (RRRR) *serial* manipulator model with streamlined topology. Links are depicted as beams and are conventionally numbered from the ground to the end-effector (E), same as the revolute joints (J), depicted as cylinders. While this simplification is evidently more pleasant for computation

purposes, all links need to retain inertial properties as on the real manipulator. Similarly, all joint revolution limits need to be preserved.

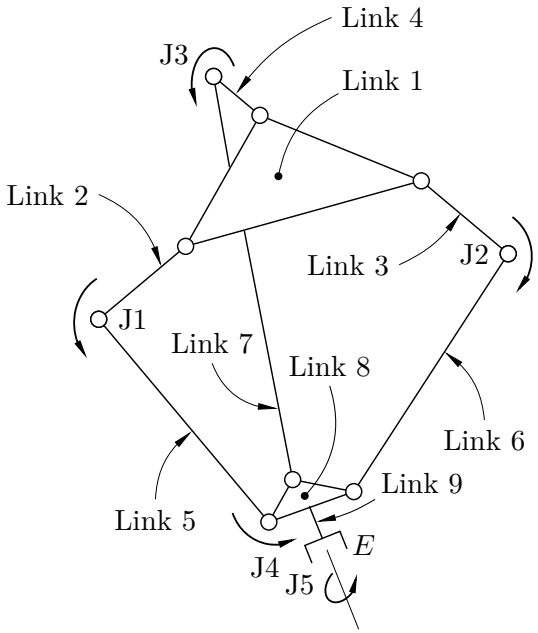


Figure 1.3 Kinematic diagram of a 5R parallel delta robot.

In contrast to the depiction in Figure 1.2, Figure 1.3 presents an alternative configuration, showcasing a 5R (RRRRR) *parallel* delta robot. A notable distinction between the two diagrams is evident: while Figure 1.2 follows an open-chain arrangement, Figure 1.3 displays a closed-loop formation involving links 1, 2, 4, 5, 7, and 8. As the primary focus of this thesis centers on open-chain configurations, the inclusion of the kinetostatic topological representation of the parallel delta robot serves primarily as an illustrative example. It underscores the potential of such diagrams in simplifying visualizations, thus enhancing computational efficiency without compromising precision.

1.2 Rigid Body Motion

Consider a rigid body freely navigating three-dimensional Euclidean space $\mathbb{E}^3$. To describe its position in space, we need to be able to describe the position of its every point. Let us define a stationary (base) coordinate system $O_1x_1y_1z_1$ anchored arbitrarily in space and affix a coordinate system $O_2x_2y_2z_2$ to the body (see Figure 1.4). Since the body is free to move, the system $O_2x_2y_2z_2$ moves relative to $O_1x_1y_1z_1$ as it's tightly bound to the body. We can then transform the *position vector* of an arbitrary point of the body from $O_2x_2y_2z_2$ to $O_1x_1y_1z_1$,

allowing us to transform the problem of describing the position of the body to describing the relative position of two coordinate systems. [13, 14]

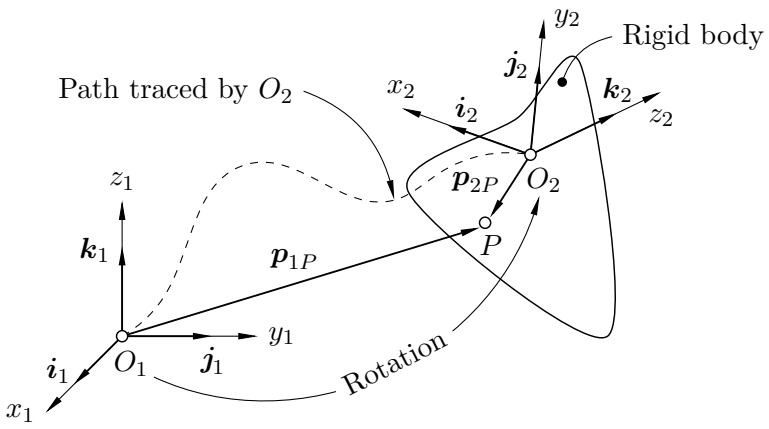


Figure 1.4 Rigid body motion.

Following Figure 1.4, the position vector of point P in the $O_1x_1y_1z_1$ coordinate system can be expressed in the form[7]

$$\boldsymbol{p}_{1P} = p_{1P_x}\boldsymbol{i}_1 + p_{1P_y}\boldsymbol{j}_1 + p_{1P_z}\boldsymbol{k}_1 = \begin{bmatrix} p_{1P_x} & p_{1P_y} & p_{1P_z} \end{bmatrix}^T \in \mathbb{R}^3,$$

in which $\boldsymbol{i}_1, \boldsymbol{j}_1, \boldsymbol{k}_1 \in \mathbb{R}^3$ denote the unit vectors along the x_1, y_1, and z_1 axes, respectively, and are defined as

$$\boldsymbol{i}_1 \coloneqq \begin{bmatrix} 1 & 0 & 0 \end{bmatrix}^T \in \mathbb{R}^3, \quad \boldsymbol{j}_1 \coloneqq \begin{bmatrix} 0 & 1 & 0 \end{bmatrix}^T \in \mathbb{R}^3, \quad \boldsymbol{k}_1 \coloneqq \begin{bmatrix} 0 & 0 & 1 \end{bmatrix}^T \in \mathbb{R}^3,$$

This notation generalizes to ${}^i\boldsymbol{p}_{jP} \in \mathbb{R}^n$, signifying that $\boldsymbol{p} \in \mathbb{R}^n$ represents the position vector of point P from the origin of coordinate system j, with its components expressed within system i. Further, in cases when $i = j$, i is often omitted. [13, 14] When considering a three-dimensional case, i.e., when $n = 3$, this general position vector can be written as

$${}^i\boldsymbol{p}_{jP} \coloneqq \begin{bmatrix} {}^ip_{jP_x} & {}^ip_{jP_y} & {}^ip_{jP_z} \end{bmatrix}^T \in \mathbb{R}^3, \tag{1.2}$$

where, if i is to *not be* equal to j to keep the prescript, $i \in \{1, \ldots, C-1\}$, $j \in \{i+1, \ldots, C\}$, with $C \in \mathbb{N}$ being the total number of coordinate systems. On the other hand, if i is to *be* equal to j, then $i, j \in \{1, \ldots, C\}$ and the position vector is denoted $\boldsymbol{p}_{i/jP}$, with respect to $O_ix_iy_iz_i$ or $O_jx_jy_jz_j$, respectively.

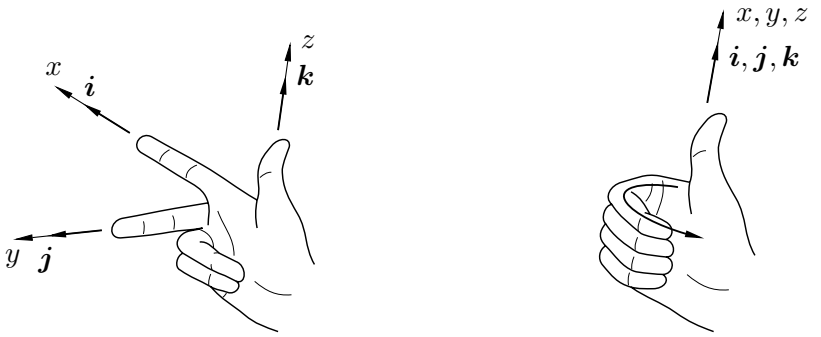
It is clear that, in the context of Figure 1.4, besides change of position, i.e., translation, the rigid body has also undergone change of orientation, i.e., rotation.

[7] A similar expression holds for the position vector of P with respect to $O_2x_2y_2z_2$.

This behaviour is described using a *rotation matrix* $\boldsymbol{R}_{ij} \in \mathbb{R}^{n\times n}$ which, within the framework of three-dimensional Euclidean space $\mathbb{E}^3$, takes the form

$$\boldsymbol{R}_{ij} \coloneqq \begin{bmatrix} r_{11} & r_{12} & r_{13} \\ r_{21} & r_{22} & r_{23} \\ r_{31} & r_{32} & r_{33} \end{bmatrix} = \begin{bmatrix} | & | & | \\ {}^i\boldsymbol{i}_j & {}^i\boldsymbol{j}_j & {}^i\boldsymbol{k}_j \\ | & | & | \end{bmatrix} \in \mathbb{R}^{3\times 3}, \tag{1.3}$$

where it is *always* the case that $i \in \{1, \ldots, C-1\}$ and $j \in \{i+1, \ldots, C\}$.



(a) Coordinate systems (b) Positive rotation

Figure 1.5 Right hand rule.

The three columns of $\boldsymbol{R}_{ij}$ encompass projections of the unit vectors of the x_j, y_j, and z_j axes of the j-th coordinate system onto the i-th coordinate system. Therefore, two conditions must necessarily be satisfied:

(i) All columns of $\boldsymbol{R}_{ij}$ need to have a unit norm given they correspond to the Cartesian unit axes of coordinate system $O_j x_j y_j z_j$, i.e.,

$$\|{}^i\boldsymbol{i}_j\| = \|{}^i\boldsymbol{j}_j\| = \|{}^i\boldsymbol{k}_j\| = 1; \tag{1.4}$$

(ii) since $O_j x_j y_j z_j$ is orthonormal and, as per the first condition, all columns of matrix $\boldsymbol{R}_{ij}$ have a unit norm, they also need to be mutually orthogonal, i.e.,

$$\langle {}^i\boldsymbol{i}_j, {}^i\boldsymbol{j}_j\rangle = \langle {}^i\boldsymbol{j}_j, {}^i\boldsymbol{k}_j\rangle = \langle {}^i\boldsymbol{i}_j, {}^i\boldsymbol{k}_j\rangle = 0, \tag{1.5}$$

where $\langle \cdot, \cdot \rangle$ denotes the Euclidean inner product in $\mathbb{R}^3$.

The six conditions stated in Eqs. (1.4) and (1.5) can be rewritten as a single constraint for the rotation matrix, i.e.,

$$\boldsymbol{R}_{ij}^T \boldsymbol{R}_{ij} = \boldsymbol{R}_{ij} \boldsymbol{R}_{ij}^T = \boldsymbol{I}_3, \tag{1.6}$$

where $\boldsymbol{I}_3 \in \mathbb{R}^{3\times 3}$ denotes the *identity matrix*. Equation (1.6) states that, as a consequence of (1.4) and (1.5), $\boldsymbol{R}_{ij}$ is an *orthonormal matrix*. [12, 15]

Lastly, we need to take the left/right-handedness of $O_jx_jy_jz_j$ into account. We presume *all* coordinate systems we define are *right-handed* (see Fig. 1.5), so

$$\det(\boldsymbol{R}_{ij}) = {}^i\boldsymbol{i}_j^T({}^i\boldsymbol{j}_j \times {}^i\boldsymbol{k}_j) = {}^i\boldsymbol{j}_j^T({}^i\boldsymbol{k}_j \times {}^i\boldsymbol{i}_j) = {}^i\boldsymbol{k}_j^T({}^i\boldsymbol{i}_j \times {}^i\boldsymbol{j}_j) = 1. \tag{1.7}$$

By virtue of $\boldsymbol{R}_{ij}$ being an arbitrary rotation matrix and satisfying conditions (1.6) and (1.7), it, along with all other rotation matrices defined in (1.3), belongs to the *special orthogonal group* $SO(3)$ (see Definition 1.1). [12, 16]

Definition 1.1. The *special orthogonal group* $SO(3) \subset \mathbb{R}^{3\times3}$, also referred to as the group of rotation matrices in $\mathbb{R}^3$, is the set of 3×3 real matrices as defined in (1.3) and satisfying conditions (1.6) and (1.7). [12] ◇

The rotation of $O_jx_jy_jz_j$ by φ_{ij} with respect to $O_ix_iy_iz_i$ can be performed around an arbitrary axis, $\mathfrak{a}$, and is then denoted $\boldsymbol{R}_{ij}(\mathfrak{a}, \varphi_{ij})$ (see Appendix A). If we, for now, choose to limit ourselves to rotations around the three Cartesian axes, x, y, and z, the respective rotation matrices then read

$$\boldsymbol{R}_{ij}(x, \varphi_{ij}) = \begin{bmatrix} 1 & 0 & 0 \\ 0 & \cos\varphi_{ij} & -\sin\varphi_{ij} \\ 0 & \sin\varphi_{ij} & \cos\varphi_{ij} \end{bmatrix} \in SO(3) \subset \mathbb{R}^{3\times3}, \tag{1.8a}$$

$$\boldsymbol{R}_{ij}(y, \varphi_{ij}) = \begin{bmatrix} \cos\varphi_{ij} & 0 & \sin\varphi_{ij} \\ 0 & 1 & 0 \\ -\sin\varphi_{ij} & 0 & \cos\varphi_{ij} \end{bmatrix} \in SO(3) \subset \mathbb{R}^{3\times3}, \tag{1.8b}$$

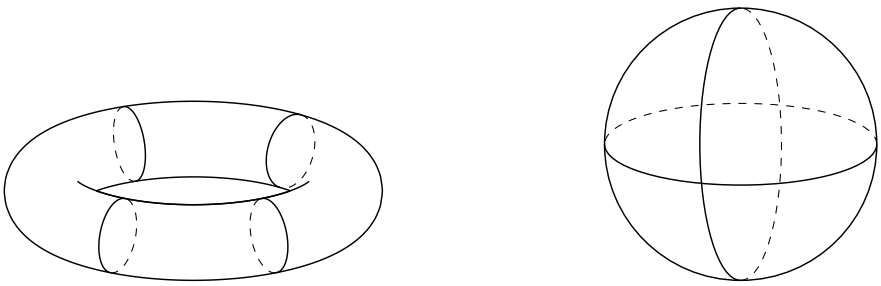
$$\boldsymbol{R}_{ij}(z, \varphi_{ij}) = \begin{bmatrix} \cos\varphi_{ij} & -\sin\varphi_{ij} & 0 \\ \sin\varphi_{ij} & \cos\varphi_{ij} & 0 \\ 0 & 0 & 1 \end{bmatrix} \in SO(3) \subset \mathbb{R}^{3\times3}, \tag{1.8c}$$

where $\varphi_{ij} \in [0, 2\pi)$. The sign of the angle of revolution can be determined via the right hand rule for positive rotation [see Figure 1.5(b)].

1.2.1 Elementary Differentiable Manifolds

In earlier text, we established the special orthogonal group $SO(3)$. This group can be simplified to two dimensions to yield its planar equivalent, $SO(2)$, which exhibits analogous behavior but in one fewer dimension. The reduction of $SO(3)$ to yield $SO(2)$ prompts a natural extension of this concept to arbitrarily many dimensions n. This generalized n-dimensional group, denoted as $SO(n)$, follows the same principles as $SO(3)$ and $SO(2)$. It encompasses $n \times n$ matrices satisfying constraints similar to those outlined in Eqs. (1.6) and (1.7). This extension is introduced to emphasize the expansive nature of rotation matrices across dimensions. However, we elect to establish a maximum limit of three dimensions for $SO(n)$ groups, given that real physical systems are spatial at most. Further, the elements of $SO(n)$ form a *differentiable manifold*, categorizing $SO(n)$ groups as

matrix *Lie Groups*.[8] To augment understanding of the mathematical concepts and provide the reader with foundational knowledge of the more intricate mathematics surrounding rotations, we introduce two elementary differentiable manifolds that the reader has previously encountered, while refraining from delving extensively into topics such as topology, differential geometry, and related subjects. [12, 19]



(a) Torus $[\mathscr{T}^3(\omega,\sigma) \subseteq \mathbb{R}^3]$ (b) Sphere $[\mathscr{S}^2(\omega,\sigma) \subseteq \mathbb{R}^3]$

Figure 1.6 Elementary differentiable manifolds.

▷ **Torus.** The n-dimensional torus, $\mathscr{T}^n \subseteq \mathbb{R}^n$ is formally defined as the Cartesian product of n circles, denoted $\mathscr{S}^1$, i.e.,

$$\mathscr{T}^n \coloneqq \underbrace{\mathscr{S}^1 \times \mathscr{S}^1 \times \cdots \times \mathscr{S}^1}_{n\text{-times}} \subseteq \mathbb{R}^n, \tag{$\dagger$}$$

or, if we limit ourselves to three dimensions, ($\dagger$) becomes

$$\mathscr{T}^3(\omega,\sigma) \coloneqq \begin{bmatrix} (R + r\cos\omega)\cos\sigma \\ (R + r\cos\omega)\sin\sigma \\ r\sin\omega \end{bmatrix} \subseteq \mathbb{R}^3, \quad \forall\ \omega,\sigma \in [0, 2\pi), \tag{$\dagger\dagger$}$$

where $R > 0$ is the distance from the center of the torus to the center of the tube, also referred to as the *major* radius, and $r > 0$ is the radius of the tube, also referred to as the *minor radius*. [20, 21, 22, 23, 24]

▷ **Sphere.** The n-dimensional sphere $\mathscr{S}^n \subseteq \mathbb{R}^{n+1}$ is the set of points in $\mathbb{R}^{n+1}$ at a fixed radius $R > 0$ from the origin. Using set notation,

$$\mathscr{S}^n \coloneqq \left\{(x_0, x_1, \ldots, x_n) \in \mathbb{R}^{n+1} | x_0^2 + x_1^2 + \cdots + x_n^2 = R^2\right\} \subseteq \mathbb{R}^{n+1}, \tag{$*$}$$

or, in the fashion of ($\dagger\dagger$), the three-dimensional sphere reads

$$\mathscr{S}^2(\omega,\sigma) \coloneqq \begin{bmatrix} R\sin\omega\cos\sigma \\ R\sin\omega\sin\sigma \\ R\cos\omega \end{bmatrix} \subseteq \mathbb{R}^3, \quad \omega \in [0,\pi] \text{ and } \sigma \in [0, 2\pi). \tag{$**$}$$

If $R = 1$, ($**$) is referred to as the *unit sphere* in $\mathbb{R}^3$. [20, 21, 22, 23, 24]

[8] For additional insights on Lie groups and Lie algebras, see [17, 18].

1.2.2 Properties of Rotation Matrices

Besides (1.6, 1.7), the membership of rotation matrices in n-dimensional special orthogonal groups implies various inherent characteristics. We choose to list out two important properties rotation matrices as entries of $SO(n)$ posses: the *inverse element existence* and *closure* (see Theorems 1.1, 1.2).[9] [12, 19]

Theorem 1.1. The inverse of a rotation matrix $\boldsymbol{R}_{ij} \in SO(3)$ is equal to its transpose, and is also a rotation matrix in $SO(3)$ ◇

Proof. As $\boldsymbol{R}_{ij} \in SO(3)$ is an orthogonal matrix, it is inherently invertible, with its inverse, denoted $\boldsymbol{R}_{ij}^{-1} \equiv \boldsymbol{R}_{ji}$, being equivalent to its transpose $\boldsymbol{R}_{ij}^{T}$, implying

$$\boldsymbol{R}_{ij}^{T}\boldsymbol{R}_{ij} = \boldsymbol{R}_{ij}\boldsymbol{R}_{ij}^{T} = \boldsymbol{I}_3. \tag{1.9}$$

Furthermore, since $\boldsymbol{R}_{ij}$ is square and satisfies Eq. (1.7), $\det(\boldsymbol{R}_{ij}^{T}) = \det(\boldsymbol{R}_{ij}) = 1$. Hence, the inverse $\boldsymbol{R}_{ij}^{-1}$ fulfills Def. 1.1, thereby concluding the proof. [12] □

The inverse element existence property is paramount in the context of reversing rotations imparted onto rigid bodies. In particular, when a rigid body's arrangement results from a sequence of rotations encapsulated within a composite rotation matrix (as outlined in Theorem 1.2), obtaining the inverse of this composite rotation matrix allows for the precise restoration of the body's original configuration, irrespective of the initial number of rotations applied. This property ensures that even complex sequences of rotations can be effectively undone, thereby maintaining the integrity of the body's orientation.

Theorem 1.1 is built upon foundational concepts of linear algebra, such as the equivalence between the inverse and transpose of an orthogonal matrix. However, exhaustive proofs for these assertions are omitted within this thesis, as their comprehensive treatment lies beyond its intended scope.[10]

Theorem 1.2. The product of two rotation matrices $\boldsymbol{R}_{ij}$,$\boldsymbol{R}_{jk}$, both elements of $SO(3)$, is also a rotation matrix in $SO(3)$. ◇

Proof. Let $\boldsymbol{R}_{ij}, \boldsymbol{R}_{jk} \in SO(3)$ be two arbitrary rotation matrices defined in (1.3) and satisfying (1.6, 1.7). Since both $\boldsymbol{R}_{ij}$ and $\boldsymbol{R}_{jk}$ are orthogonal, their product, $\boldsymbol{R}_{ik} = \boldsymbol{R}_{ij}\boldsymbol{R}_{jk}$ is *also* orthogonal and hence satisfies

(i) the inverse element existence, i.e.,

$$[\boldsymbol{R}_{ij}\boldsymbol{R}_{jk}]^{T}\boldsymbol{R}_{ij}\boldsymbol{R}_{jk} = \boldsymbol{R}_{jk}^{T}\boldsymbol{R}_{ij}^{T}\boldsymbol{R}_{ij}\boldsymbol{R}_{jk} = \boldsymbol{R}_{jk}^{T}\boldsymbol{R}_{jk} = \boldsymbol{I}_3,$$

where, as per inverse element existence (Theorem 1.1), $\boldsymbol{R}_{ij}^{T}\boldsymbol{R}_{ij} = \boldsymbol{I}_3$,

[9] An enlightening discussion on all $SO(n)$ properties is held in [12], pp. 70 — 71, and [19], p. 72.

[10] In case of need, for the proof and subsequent discussion on the inverse-transpose equivalence, the reader is encouraged to consult [25], pp. 345 — 346, Theorems 6,7 and Example 7; for the proof of the determinant and transpose determinant equivalence, see [26], p. 390, Thm. 9.55(a).

(ii) the right-handedness constraint, i.e., Eq. (1.7),

$$\det(\boldsymbol{R}_{ik}) = \det(\boldsymbol{R}_{ij}\boldsymbol{R}_{jk}) = \det(\boldsymbol{R}_{ij})\det(\boldsymbol{R}_{jk}) = 1.$$

It is obvious the former property is condition (1.6). Since (1.6) and (1.7) define $SO(3)$, we have proven $\boldsymbol{R}_{ik} = \boldsymbol{R}_{ij}\boldsymbol{R}_{jk} \in SO(3)$. [12] □

The significance of the closure property lies in its facilitation of straightforward multiplication of rotation matrices for executing rotation sequences on rigid bodies. This property extends to an indefinite number of rotation matrices, resulting in a composite rotation matrix that remains within $SO(3)$. As we'll discover further, this property is equally applicable to homogeneous transformation matrices, with its significance being most prominent in this context.

In a manner reminiscent of Theorem 1.1, Theorem 1.2 is predicated on fundamental assertions within linear algebra. This theorem hinges on the recognition of the equivalence between the determinant of a matrix product and the product of the determinants of the constituent matrices. Once more, we abstain from presenting the proof for this statement, given its comprehensive treatment in numerous textbooks on linear algebra and related subjects.[11]

1.2.3 Vector Transformations

Understanding the geometrical meaning of rotation matrices is easier when we look at how a point is represented in different coordinate systems. Consider a point $P \in \mathbb{E}^3$ located somewhere in space and two coordinate systems $O_i x_i y_i z_i$ and $O_j x_j y_j z_j$ whose origins coincide, i.e., $O_i \equiv O_j$, and $O_j x_j y_j z_j$ is otherwise arbitrarily rotated with respect to $O_i x_i y_i z_i$ (see Figure 1.7).

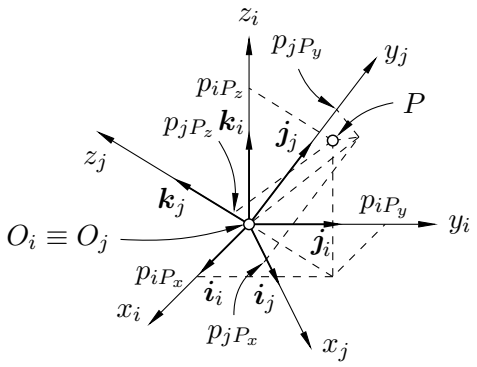


Figure 1.7 Representing a point in different coordinate systems.

When we examine the construct from Figure 1.7, we shall find out that the two respective position vectors of P, $\boldsymbol{p}_{iP} \in \mathbb{R}^3$, i.e., within $O_i x_i y_i z_i$, and $\boldsymbol{p}_{jP} \in \mathbb{R}^3$, i.e.,

[11] The intrigued reader is referred to Theorem 6 and Example 5 in [25], p. 175.

within $O_j x_j y_j z_j$, are related only by the rotation matrix $\boldsymbol{R}_{ij} \in SO(3)$ describing the orientation of $O_j x_j y_j z_j$ within $O_i x_i y_i z_i$ (see Theorem 1.3).

Theorem 1.3. The position vectors $\boldsymbol{p}_{iP} \in \mathbb{R}^3$ and $\boldsymbol{p}_{jP} \in \mathbb{R}^3$ of any point $P \in \mathbb{E}^3$ in two coordinate systems whose origins coincide and are otherwise arbitrarily rotated relative to each other are related by $\boldsymbol{p}_{iP} = \boldsymbol{R}_{ij}\boldsymbol{p}_{jP}$. ◇

Proof. We can express the position vector of point P either in the form

$$\boldsymbol{p}_{iP} = \begin{bmatrix} p_{iP_x} & p_{iP_y} & p_{iP_z} \end{bmatrix}^T \in \mathbb{R}^3,$$

with respect to the $O_i x_i y_i z_i$ system, or as

$$\boldsymbol{p}_{jP} = \begin{bmatrix} p_{jP_x} & p_{jP_y} & p_{jP_z} \end{bmatrix}^T \in \mathbb{R}^3.$$

when expressed with respect to the rotated system $O_j x_j y_j z_j$. Given both $\boldsymbol{p}_{iP}$ and $\boldsymbol{p}_{jP}$ represent the same point,

$$\boldsymbol{p}_{iP} = p_{jP_x}{}^i\boldsymbol{i}_j + p_{jP_y}{}^i\boldsymbol{j}_j + p_{jP_z}{}^i\boldsymbol{k}_j \in \mathbb{R}^3$$

must hold. The above expression can be rewritten to the form

$$\boldsymbol{p}_{iP} = \begin{bmatrix} | & | & | \\ {}^i\boldsymbol{i}_j & {}^i\boldsymbol{j}_j & {}^i\boldsymbol{k}_j \\ | & | & | \end{bmatrix} \boldsymbol{p}_{jP} = \boldsymbol{R}_{ij}\boldsymbol{p}_{jP} \in \mathbb{R}^3, \tag{1.10}$$

which proves the statement presented in Theorem 1.3.[12] [15] □

Besides describing the orientation of two coordinate systems or representing a point in different coordinate systems, we can also interpret a rotation matrix as an operator allowing for rotation of a vector by a given angle around an arbitrary axis in space, an observation hinted at by Equation (1.10). Notably, both the original and the rotated vector have the same norm, which is easy to prove.[13] [15]

1.2.4 Euler XYZ Angles

Given that rotation matrices are composed of nine entries which are not independent but related by the orthonormality conditions (1.4, 1.5), their description of the orientation of a system is surplus. Further, Eqs. (1.4, 1.5) also imply the need of only *three* parameters for rigid body orientation description as the *minimal representation* of the $SO(n)$ group demands $n(n-1)/2$ parameters. [15]

Let us consider a minimal representation of $SO(3)$ using a set of three angles $\boldsymbol{\phi} = [\alpha \quad \beta \quad \gamma]^T \in \mathbb{R}^3$. As per Theorem 1.2, a generic rotation matrix can be

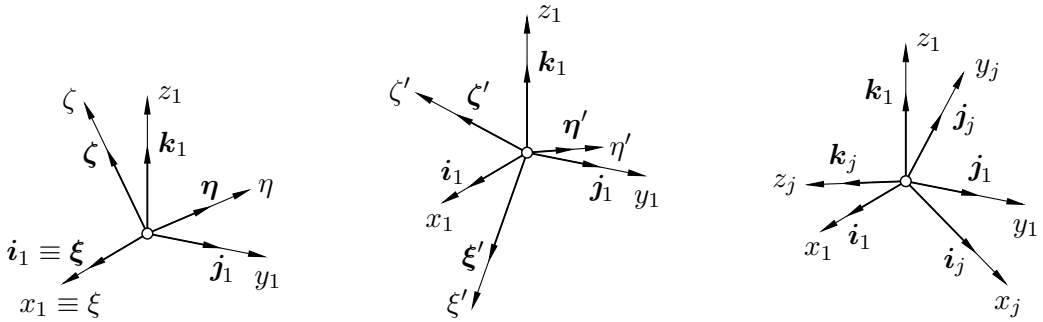
[12] A different version of the same proof can be found in [27], p. 57, Theorem 2.5.1.
[13] The proof is done by employing the inverse element existence property.

composed as the product of three elementary rotations by α, β, and γ. For this representation of $SO(3)$ to be minimal yet properly relate two independent orthonormal coordinate systems, successive rotations need to occur about non-parallel axes. Consequently, among the 27 possible combinations of rotations, only 12 sets are permissible, i.e.,

XYZ	YZX	ZXY
XZY	YXZ	ZYX
XYX	YZY	ZXZ
XZX	YXY	ZYZ

where each set forms a triplet known as the *Euler angles*. [15, 28, 29]

One distinct set of Euler angles for our case are the XYZ angles,[14] also referred to as the *Roll-Pitch-Yaw* (RPY) angles (see Figure 1.8). The orientation of $O_j x_j y_j z_j$ resulting from the XYZ (RPY) rotation sequence is *always* expressed within the the global coordinate system $O_1 x_1 y_1 z_1$ hence why the XYZ sequence is essential in robotics for end-effector orienting. [12, 31]



(a) 1st rotation, Yaw (Y) (b) 2nd rotation, Pitch (P) (c) 3rd rotation, Roll (R)

Figure 1.8 Euler XYZ rotation sequence.

Remark 1.3. The axis notation for the RPY angles can vary depending on the author, i.e., some authors denote them by the respective rotation sequence (XYZ), e.g., [12, 29], while others use the ZYX denotation, e.g., [15], due to the fact that the RPY angles express orientation with respect to the global coordinate system and hence the product of these elementary rotations goes in the ZYX order [see Eq. (1.11)]. We shall use the standard XYZ notation. ◇

Remark 1.4. Contrary to the XYZ sequence, the ZYX sequence corresponds to the orientation of $O_j x_j y_j z_j$ within $O_i x_i y_i z_i$ which was formerly coincident with

[14] For further material on some other Euler angles sets, consult [15], pp. 49 — 51 (ZYZ Euler angles) or [30] pp. 150 — 154 (ZXZ Euler angles).

the j-th system before the sequence took place. As we will later discover, the resulting rotation matrix is identical for both sequences, yet it allows for varying physical interpretations. [12, 29] ◇

To obtain $O_jx_jy_jz_j$ from $O_1x_1y_1z_1$, the Euler XYZ rotation sequence is:

rotate $O_1x_1y_1z_1$ by γ	Start with a stationary
about the x_1 axis to obtain ⟵	global coordinate
$(O)\xi\eta\zeta$ [Fig. 1.8(a)],	system $O_1x_1y_1z_1$,
↓	
rotate $(O)\xi\eta\zeta$ by β	rotate $(O)\xi'\eta'\zeta'$ by α
about the y_1 axis to obtain ⟶	about the z_1 axis to obtain
$(O)\xi'\eta'\zeta'$ [Fig. 1.8(b)],	$O_jx_jy_jz_j$ [Fig. 1.8(c)].

Then, the resulting orientation of $O_jx_jy_jz_j$ with respect to the global coordinate system $O_1x_1y_1z_1$ reads

$$\begin{aligned}\boldsymbol{R}_{1j}(\boldsymbol{\phi}) &= \overbrace{\boldsymbol{R}_{(O)\xi'\eta'\zeta'j}(z_1,\alpha)}^{\text{Roll (R)}}\overbrace{\boldsymbol{R}_{(O)\xi\eta\zeta(O)\xi'\eta'\zeta'}(y_1,\beta)}^{\text{Pitch (P)}}\overbrace{\boldsymbol{R}_{1(O)\xi\eta\zeta}(x_1,\gamma)}^{\text{Yaw (Y)}} \\ &= \begin{bmatrix} \mathrm{c}_\alpha\mathrm{c}_\beta & \mathrm{c}_\alpha\mathrm{s}_\beta\mathrm{s}_\gamma - \mathrm{s}_\alpha\mathrm{c}_\gamma & \mathrm{c}_\alpha\mathrm{s}_\beta\mathrm{c}_\gamma + \mathrm{s}_\alpha\mathrm{s}_\gamma \\ \mathrm{s}_\alpha\mathrm{c}_\beta & \mathrm{s}_\alpha\mathrm{s}_\beta\mathrm{s}_\gamma + \mathrm{c}_\alpha\mathrm{c}_\gamma & \mathrm{s}_\alpha\mathrm{s}_\beta\mathrm{c}_\gamma - \mathrm{c}_\alpha\mathrm{s}_\gamma \\ -\mathrm{s}_\beta & \mathrm{c}_\beta\mathrm{s}_\gamma & \mathrm{c}_\beta\mathrm{c}_\gamma \end{bmatrix} \in SO(3),\end{aligned} \tag{1.11}$$

where s_α and c_α are shorthand notation for $\sin\alpha$ and $\cos\alpha$, etc. Coming back to Remark 1.4, the ZYX rotation sequence would have the same resulting orientation if one were to compute it. Yet, their physical interpretations are different as highlighted in previous discussion.[15]

Besides Euler angles, there exist numerous other orientation representations, such as the Euler-Rodrigues parameters,[16] the Cayley-Rodrigues parameters, the Cardan angles, among others, the first two of which are subject of Appendix A.

1.2.5 Homogeneous Transformation Matrices

The obvious approach to define the configuration of a rigid body concerning $O_ix_iy_iz_i$ is by combining the position vector $\boldsymbol{p}_{iO_j} \in \mathbb{R}^3$ of the origin of $O_jx_jy_jz_j$, affixed to the body, with the rotation matrix $\boldsymbol{R}_{ij} \in SO(3)$ describing the orientation of $O_jx_jy_jz_j$ relative to $O_ix_iy_iz_i$. Instead of treating $\boldsymbol{p}_{iO_j}$ and $\boldsymbol{R}_{ij}$ as separate entities, let us introduce a *homogeneous transformation matrix* denoted as $\boldsymbol{H}_{ij} \in \mathbb{R}^{4\times4}$, which integrates both the position vector and the rotation matrix into a combined representation of $O_jx_jy_jz_j$ within $O_ix_iy_iz_i$.

[15] See Remark 1.4 for such discussion.

[16] The Euler-Rodrigues parameters are also referred to as the *unit quaternion* representation.

Definition 1.2. The *special Euclidean Group* $SE(3) \subset \mathbb{R}^{4\times4}$, also referred to as the group of homogeneous transformation matrices in $\mathbb{R}^3$, is the set of 4×4 real (compound) matrices of the form

$$\boldsymbol{H}_{ij} \coloneqq \left[\begin{array}{c|c} \boldsymbol{R}_{ij} \in SO(3) & \boldsymbol{p}_{iO_j} \in \mathbb{R}^3 \\ \hline \boldsymbol{0}_3^T \in \mathbb{R}^3 & 1 \end{array}\right] \in \mathbb{R}^{4\times4}, \tag{1.12}$$

where $\boldsymbol{0}_3 \in \mathbb{R}^3$ is referred to as the *null* vector. [12] ◇

Homogeneous transformation matrices behave in similar fashion to rotation matrices. For one, as with rotation matrices, the product of two homogeneous transformation matrices $\boldsymbol{H}_{ij}$ and $\boldsymbol{H}_{jk} \in SE(3)$ is *also* a homogeneous transformation matrix, yet generally $\boldsymbol{H}_{ij}\boldsymbol{H}_{jk} \neq \boldsymbol{H}_{jk}\boldsymbol{H}_{ij}$, analogous to $\boldsymbol{R}_{ij}\boldsymbol{R}_{jk} \neq \boldsymbol{R}_{jk}\boldsymbol{R}_{ij}$.

A notable deviation from this analogy arises concerning the orthogonality property, where for homogeneous transformation matrices, the orthogonality property does not hold, and therefore

$$\boldsymbol{H}_{ij}^{-1} \neq \boldsymbol{H}_{ij}^T,$$

in contrast to rotation matrices. Consequently, the inverse of a homogeneous transformation matrix is obtained through direct computation and reads [12, 15]

$$\boldsymbol{H}_{ij}^{-1} \equiv \boldsymbol{H}_{ji} = \left[\begin{array}{c|c} \boldsymbol{R}_{ij}^T \in SO(3) & -\boldsymbol{R}_{ij}^T\boldsymbol{p}_{iO_j} \in \mathbb{R}^3 \\ \hline \boldsymbol{0}_3^T \in \mathbb{R}^3 & 1 \end{array}\right] \in SE(3) \subset \mathbb{R}^{4\times4}.$$

Theorem 1.4. A homogeneous transformation $\boldsymbol{H}_{ij} \in SE(3)$ is *isometric*, i.e., it preserves distances and angles between objects. ◇

Proof. Let $\boldsymbol{m}, \boldsymbol{n}, \boldsymbol{o} \in \mathbb{R}^3$ be column vectors and $\boldsymbol{H}_{ij} \in SE(3)$ a homogeneous transformation. By direct computation, one can verify the following:

(i) $\|\boldsymbol{H}_{ij}\boldsymbol{m} - \boldsymbol{H}_{ij}\boldsymbol{n}\| = \|\boldsymbol{m} - \boldsymbol{n}\|$ and vice versa,

(ii) $\langle \boldsymbol{H}_{ij}\boldsymbol{m} - \boldsymbol{H}_{ij}\boldsymbol{o}, \boldsymbol{H}_{ij}\boldsymbol{n} - \boldsymbol{H}_{ij}\boldsymbol{o}\rangle = \langle \boldsymbol{m} - \boldsymbol{o}, \boldsymbol{n} - \boldsymbol{o}\rangle$ and vice versa,

where the former identity is the preservation of distances, while the latter one is the preservation of angles, thus completing the proof. [12] □

Theorem 1.4 holds significant importance, as it establishes a critical assurance: post-transformation, the intrinsic distances and relative angles among any three arbitrarily chosen points on a rigid body persist unaltered. This guarantee plays a pivotal role in fostering a thorough comprehension of the body's spatial configuration, ensuring the faithful preservation of its structural integrity and spatial relationships throughout the transformation process.

1.3 Open-Chain Kinematics

Creating the kinematic model of an open kinematic chain can be decoupled into two problems. The first, typically regarded as simpler in the context of an open-chain structure, is known as the *forward kinematics* problem. The second, considered significantly more challenging, is the *inverse kinematics* problem. Furthermore, the inverse solution encounters kinematic singularities, which prevent the end-effector of a robotic structure from being positioned arbitrarily within its workspace. Let us begin our exploration from the forward kinematics problem and gradually progress towards inverse kinematics and singularity analysis.

1.3.1 Forward Kinematics

The forward kinematics problem involves determining the end-effector position and orientation, concerning the global coordinate system $O_1x_1y_1z_1$, based on the joint coordinates of the robotic manipulator,[17] i.e., the mapping[18]

$$\underbrace{\boldsymbol{\varphi} = \begin{bmatrix} \varphi_{12} & \cdots & \varphi_{J,J+1} \end{bmatrix}^T \in \mathbb{R}^J}_{\text{known joint coordinates}} \mapsto \underbrace{\tilde{\boldsymbol{\varepsilon}}(\boldsymbol{\varphi}) = \begin{bmatrix} \boldsymbol{p}_{1E}^T(\boldsymbol{\varphi}) & \boldsymbol{\phi}_E^T(\boldsymbol{\varphi}) \end{bmatrix}^T \in \mathbb{R}^6}_{\text{unknown end-effector pose}}, \tag{1.13}$$

where $J \in \mathbb{N}$ is the number of joints of the robot, $\boldsymbol{\varphi} \in \mathbb{R}^J$ is the vector of joint coordinates, $\tilde{\boldsymbol{\varepsilon}}(\boldsymbol{\varphi}) \in \mathbb{R}^6$ is the end-effector configuration vector, $\boldsymbol{p}_{1E}(\boldsymbol{\varphi}) \in \mathbb{R}^3$ is the position vector of the end-effector, and $\boldsymbol{\phi}_E(\boldsymbol{\varphi}) = [\alpha_E \quad \beta_E \quad \gamma_E]^T \in \mathbb{R}^3$ is the orientation vector of the end-effector, expressed using the global XYZ (RPY) Euler angles (see Subsection 1.2.4).

In cases when a tool, e.g., a milling cutter, drill, etc., is attached as the end-effector, it is more convenient to talk about the *tool center point* (often abbreviated as TCP), instead of a general end-effector point. The forward kinematics problem then becomes a slightly modified version of (1.13), i.e.,

$$\underbrace{\boldsymbol{\varphi} = \begin{bmatrix} \varphi_{12} & \cdots & \varphi_{J,J+1} \end{bmatrix}^T \in \mathbb{R}^J}_{\text{known joint coordinates}} \mapsto \underbrace{\boldsymbol{\varepsilon}(\boldsymbol{\varphi}) = \begin{bmatrix} \boldsymbol{p}_{1\text{TCP}}^T(\boldsymbol{\varphi}) & \boldsymbol{\phi}_{\text{TCP}}^T(\boldsymbol{\varphi}) \end{bmatrix}^T \in \mathbb{R}^6}_{\text{unknown TCP pose}}, \tag{1.13'}$$

where $\boldsymbol{\varepsilon}(\boldsymbol{\varphi}) \in \mathbb{R}^6$ is the TCP configuration vector, in which $\boldsymbol{p}_{1\text{TCP}}(\boldsymbol{\varphi}) \in \mathbb{R}^3$ and $\boldsymbol{\phi}_{\text{TCP}}(\boldsymbol{\varphi}) = [\alpha_{\text{TCP}} \quad \beta_{\text{TCP}} \quad \gamma_{\text{TCP}}]^T \in \mathbb{R}^3$ are the position and orientation vectors of the tool center point, respectively.

Seeing as (1.13') is analogous to (1.13), the question now is how does one proceed to precisely adhere to the Scheme (1.13'), or (1.13) for that matter. The approach we shall use is by utilizing homogeneous transformations $\boldsymbol{H}_{ij} \in SE(3)$, more specifically the *product* of multiple homogeneous transformations, resulting

[17] By the term "joint coordinates", we mean the angular displacement of each actuated joint.
[18] A mapping is when an element is linked to another through a function or transformation.

in a homogeneous transformation matrix $\boldsymbol{H}_{1\mathrm{TCP}}(\boldsymbol{\varphi}) \in SE(3)$, relating the tool center point to the global coordinate system $O_1x_1y_1z_1$, i.e.,

$$\begin{aligned}\boldsymbol{H}_{1\mathrm{TCP}}(\boldsymbol{\varphi}) &= \prod_{i=1}^{J+1}\prod_{j>i}^{C}\boldsymbol{H}_{ij}(\boldsymbol{\varphi}) = \boldsymbol{H}_{12}(\varphi_{12})\boldsymbol{H}_{23}(\varphi_{23})\cdots\boldsymbol{H}_{J+1,C}(\varphi_{J,J+1}) \\ &= \left[\begin{array}{c|c} \boldsymbol{R}_{1\mathrm{TCP}}(\boldsymbol{\varphi}) \in SO(3) & \boldsymbol{p}_{1\mathrm{TCP}}(\boldsymbol{\varphi}) \in \mathbb{R}^3 \\ \hline \mathbf{0}_3^T \in \mathbb{R}^3 & 1 \end{array}\right] \in SE(3),\end{aligned} \tag{1.14}$$

assuming the TCP is the origin of the last, C-th coordinate system $O_Cx_Cy_Cz_C$.

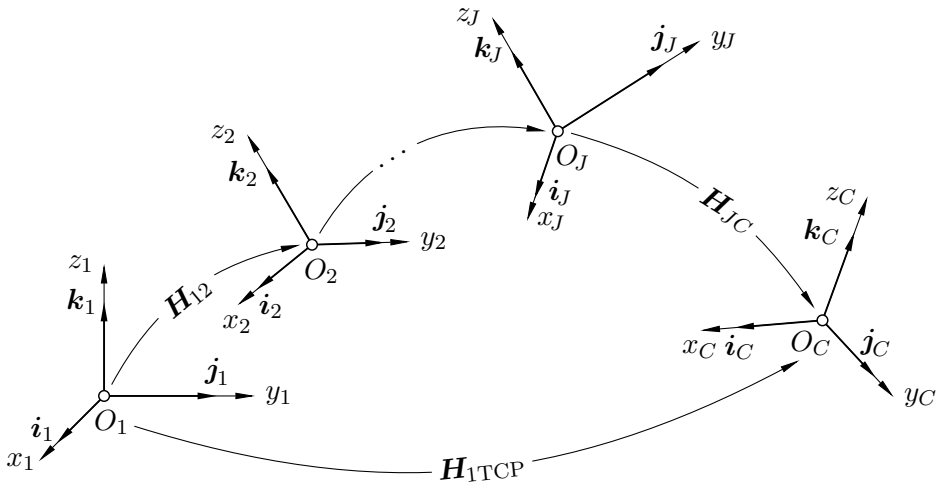


Figure 1.9 Open-chain forward kinematics.

The overall transformation $\boldsymbol{H}_{1\mathrm{TCP}}(\boldsymbol{\varphi})$ from (1.14), gives only the first portion of $\boldsymbol{\varepsilon}(\boldsymbol{\varphi})$ directly. To obtain the remaining components of $\boldsymbol{\varepsilon}(\boldsymbol{\varphi})$, i.e., the orientation vector of the tool center point $\boldsymbol{\phi}_{\mathrm{TCP}}(\boldsymbol{\varphi})$, and thus the XYZ Euler angles of the TCP, one has to do a little more work. Given an overall transformation matrix $\boldsymbol{H}_{1\mathrm{TCP}}(\boldsymbol{\varphi})$, the Euler XYZ angles can be obtained by comparing $\boldsymbol{R}_{1\mathrm{TCP}}(\boldsymbol{\varphi})$ to the general rotation matrix $\boldsymbol{R}_{1\mathrm{TCP}}(\boldsymbol{\phi}_{\mathrm{TCP}})$ from (1.11), resulting in[19]

$$\alpha_{\mathrm{TCP}} = \operatorname{atan2}\left[\frac{r_{21}}{\cos(\arcsin r_{31})}, \frac{r_{11}}{\cos(\arcsin r_{31})}\right], \tag{1.15a}$$

$$\beta_{\mathrm{TCP}} = -\arcsin r_{31}, \tag{1.15b}$$

$$\gamma_{\mathrm{TCP}} = \operatorname{atan2}\left[\frac{r_{32}}{\cos(\arcsin r_{31})}, \frac{r_{33}}{\cos(\arcsin r_{31})}\right], \tag{1.15c}$$

[19]The denominators in Eqs. (1.15a, 1.15c) are nothing but $\cos\beta_{\mathrm{TCP}}$, having taken into account that $\arcsin(-x) = -\arcsin x$ and $\cos(-x) = \cos x$.

where $\beta_{\mathrm{TCP}} \in [-\pi/2, \pi/2]$, $\alpha_{\mathrm{TCP}}, \gamma_{\mathrm{TCP}} \in (-\pi, \pi]$, and

$$\operatorname{atan2}(r,q) = \begin{cases} \arctan(r/q) & \text{if } q > 0, \\ \arctan(r/q) + \pi & \text{if } r \geq 0 \text{ and } q < 0, \\ \arctan(r/q) - \pi & \text{if } r < 0 \text{ and } q < 0, \\ \pi/2 & \text{if } r > 0 \text{ and } q = 0, \\ -\pi/2 & \text{if } r < 0 \text{ and } q = 0, \\ \text{undefined} & \text{if } q = r = 0 \end{cases}$$

is a two-argument version of the arctangent function, commonly used in computer programming to compute the angle from the positive x-axis to the point $[q, r]$ in the Cartesian plane, yielding an angle θ in [rad] such that $\theta \in (-\pi, \pi]$. Having now computed $\boldsymbol{\phi}_{\mathrm{TCP}}(\boldsymbol{\varphi})$ from (1.15a — 1.15c) and $\boldsymbol{p}_{1\mathrm{TCP}}(\boldsymbol{\varphi})$ from (1.14), the transition from $\boldsymbol{\varphi}$ to $\boldsymbol{\varepsilon}(\boldsymbol{\varphi})$ as per Scheme (1.13') is complete.

Remark 1.5. Solutions (1.15a) and (1.15c) encounter singularities at $r_{31} = \pm 1$, corresponding to $\beta_{\mathrm{TCP}} = \pm\pi/2$, which remain valid solutions. In such scenarios, it becomes necessary to derive expressions from $\boldsymbol{R}_{1\mathrm{TCP}}$ and subsequently solve for the orientation vector of the TCP, $\boldsymbol{\phi}_{\mathrm{TCP}}(\boldsymbol{\varphi})$, accordingly. ◇

The methodology we have described assumes that all coordinate systems can be defined arbitrarily, provided they are right-handed. An alternative method for open-chain forward kinematics involves utilizing the Denavit-Hartenberg parameters, which are discussed in most major textbooks on robotics. [12, 15, 29]

1.3.2 Closed-Form Inverse Kinematics

As hinted in the introduction to this section, the inverse kinematics problem presents a considerably greater challenge compared to its forward counterpart. The fundamental concept involves a straightforward reversal of the forward kinematics problem, specifically the desire for computing all joint variables based on the desired pose of the end-effector or the tool center point (TCP), i.e.,

$$\underbrace{\boldsymbol{\varepsilon} = \begin{bmatrix} \boldsymbol{p}_{1\mathrm{TCP}}^T & \boldsymbol{\phi}_{\mathrm{TCP}}^T \end{bmatrix}^T \in \mathbb{R}^6}_{\text{known (desired) TCP pose}} \mapsto \underbrace{\boldsymbol{\varphi}(\boldsymbol{\varepsilon}) = \begin{bmatrix} \varphi_{12} & \cdots & \varphi_{J,J+1} \end{bmatrix}^T \in \mathbb{R}^J}_{\text{unknown joint coordinates}}, \tag{1.16}$$

and similarly for an arbitrary end-effector. As the complexity and dimension of the manipulator increase, Schemes similar to (1.16) become increasingly challenging. While planar arms with fewer links generally allow for relatively straightforward solutions to the planar form of (1.16), transitioning to spatial arms with more joints and consequently more links presents challenges in finding a closed-form solution. In fact, certain types of spatial arms have no closed-form solutions at all, necessitating the use of numerical methods, the subject of Appendix B.

The majority of spatial serial structures, potentially capable of accommodating closed-form solutions for their inverse kinematics, give rise to sets of nonlinear equations[20] when attempting to derive such solutions owing to their intricate geometric arrangements. As a result, closed-form solutions may be within reach, albeit regrettably only under specific configurations or simplified representations of the arms, and are not universally accessible for arbitrary poses of the end-effector, or the tool center point for that matter. A solution, yet still partial, is the employment of the 6R *anthropomorphic arm with a spherical wrist*,[21] designed to mimic the structure and movement capabilities of the human arm. This type of arm finds applications in a diverse array of fields such as machining, welding, material handling, and is favoured in many industrial manipulators due to its capacity for precise positioning and enhanced dexterity. [12, 15, 29, 31]

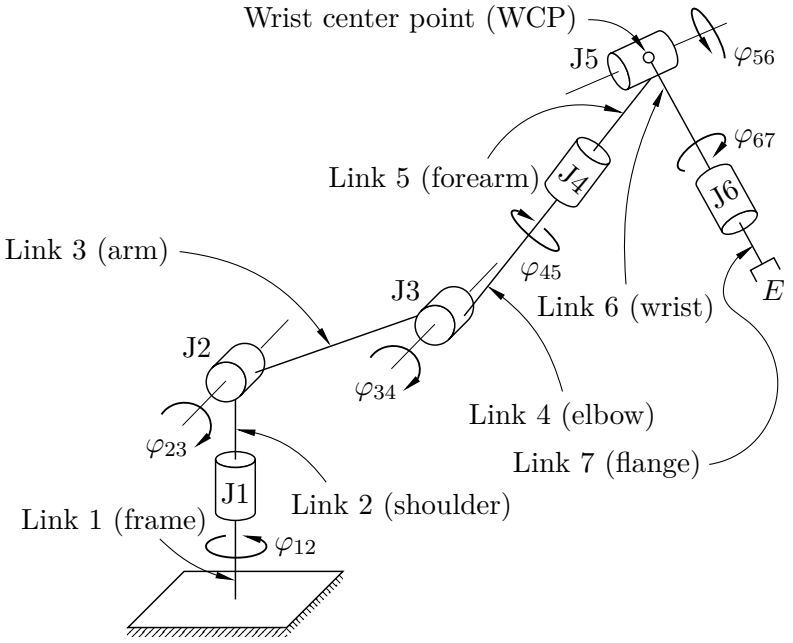


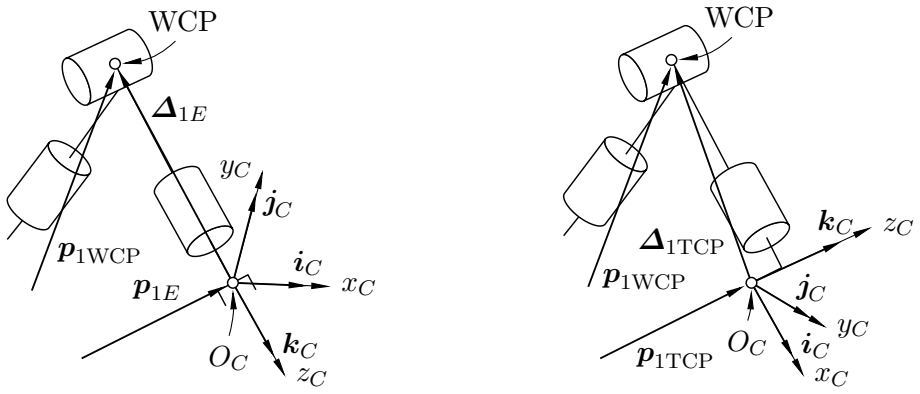
Figure 1.10 6R anthropomorphic arm with a spherical wrist.

The 6R anthropomorphic arm depicted in Figure 1.10 exhibits a configuration where its ultimate three axes of revolution converge at a solitary point termed the *wrist center point* (WCP). This design allows for the representation of the final three rotations via a spherical joint, thereby warranting the designation *spherical wrist*. Further, the implementation of the spherical wrist facilitates the segregation of the inverse kinematics problem into two separate problems, the inverse *position* kinematics problem and the inverse *orientation* kinematics problem.

[20] Equations in which the dependent variable or variables do not form a linear relationship with the independent variable or variables, i.e., ones with square roots, trigonometric functions, etc.

[21] Understand the word "anthropomorphic" as resembling human anatomy.

▷ **Inverse Position Kinematics.** The fundamental concept of inverse position kinematics involves ascertaining the precise spatial coordinates of the wrist center point based on the desired pose of the end-effector or the tool center point. This process entails deriving the position vector $\boldsymbol{p}_{1\text{WCP}}(\boldsymbol{\varepsilon} \vee \widetilde{\boldsymbol{\varepsilon}}) \in \mathbb{R}^3$ from the position vector of the end-effector/TCP, which necessitates a stepwise approach of adding constant vectors to $\boldsymbol{p}_{1E}$ or $\boldsymbol{p}_{1\text{TCP}}$ until the location of the WCP is established, with all constant vectors being defined within the overarching system $O_1x_1y_1z_1$.



(a) Arbitrary end-effector (E) (b) Tool center point (TCP)

Figure 1.11 Inverse position kinematics.

Suppose we have a rotation matrix $\boldsymbol{R}_{1E}(\boldsymbol{\phi}_E)$ or $\boldsymbol{R}_{1\text{TCP}}(\boldsymbol{\phi}_{\text{TCP}})$, which characterizes the desired orientation of either the end-effector or the tool center point using the Euler XYZ angles. Additionally, we are provided with a *constant* displacement vector representing the total displacement of the end-effector or the tool center point from the wrist center point, denoted as $\boldsymbol{d}_E = [d_{E_x} \quad d_{E_y} \quad d_{E_z}]^T \in \mathbb{R}^3$ or $\boldsymbol{d}_{\text{TCP}} = [d_{\text{TCP}_x} \quad d_{\text{TCP}_y} \quad d_{\text{TCP}_z}]^T \in \mathbb{R}^3$, respectively. The expression for the position vector of the wrist center point can be articulated as

$$\boldsymbol{p}_{1\text{WCP}}(\boldsymbol{\varepsilon} \vee \widetilde{\boldsymbol{\varepsilon}}) = \begin{cases} \boldsymbol{p}_{1E} + \boldsymbol{\Delta}_{1E}(\boldsymbol{\phi}_E) & \text{for } E, \\ \boldsymbol{p}_{1\text{TCP}} + \boldsymbol{\Delta}_{1\text{TCP}}(\boldsymbol{\phi}_{\text{TCP}}) & \text{for the TCP}, \end{cases} \tag{1.17}$$

where

$$\boldsymbol{\Delta}_{1E}(\boldsymbol{\phi}_E) = \boldsymbol{R}_{1E}(\boldsymbol{\phi}_E) \left(\boldsymbol{d}_E \odot \begin{bmatrix} \boldsymbol{i}_C & \boldsymbol{j}_C & \boldsymbol{k}_C \end{bmatrix}^T \right),$$
$$\boldsymbol{\Delta}_{1\text{TCP}}(\boldsymbol{\phi}_{\text{TCP}}) = \boldsymbol{R}_{1\text{TCP}}(\boldsymbol{\phi}_{\text{TCP}}) \left(\boldsymbol{d}_{\text{TCP}} \odot \begin{bmatrix} \boldsymbol{i}_C & \boldsymbol{j}_C & \boldsymbol{k}_C \end{bmatrix}^T \right),$$

assuming E or the TCP is the origin of the last, C-th coordinate system. The operator $\odot$ represents the Hadamard product, a binary operation capable of operating on two matrices or vectors with identical dimensions. This operation

yields a new matrix or vector in which each element is computed as the product of the corresponding elements from the input matrices or vectors. [25, 32, 33]

Now that the position of the wrist center point is determined, the inverse position kinematics transitions into a trigonometric exercise confined to a plane. This resulting 2R sub-mechanism operates within the J1-J2-J5 plane $\mathcal{P} \in \mathbb{E}^3$, rotating about the first axis of revolution by φ_{12}.

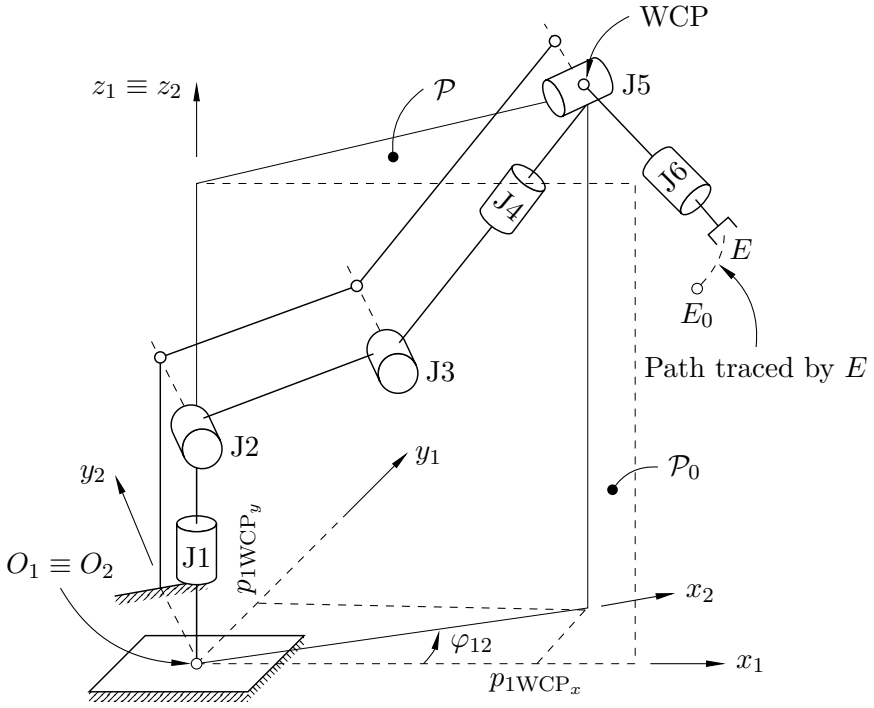


Figure 1.12 2R planar sub-mechanism of the 6R anthropomorphic arm.

The angular displacement φ_{12} between coordinate systems $O_1x_1y_1z_1$ and $O_2x_2y_2z_2$, or the initial plane $\mathcal{P}_0 \in \mathbb{E}^3$ and $\mathcal{P}$ for that matter, reads

$$\varphi_{12} = \operatorname{atan2}(p_{1\mathrm{WCP}_y}, p_{1\mathrm{WCP}_x}), \tag{1.18}$$

as per Figure 1.12. Solving for φ_{23} and φ_{34} entails the examination of two potential configurations that both achieve the designated WCP position — the *elbow-up* and *elbow-down* configurations. In the depiction provided in Figure 1.12, we specifically portray the elbow-up configuration, where the elbow (link 4), is oriented upwards. Each configuration yields distinct equations governing φ_{23} and φ_{34}, respectively, which will be systematically derived within Chapter 5.

▷ **Inverse Orientation Kinematics.** Upon computing the angles for the initial three joints, namely φ_{12}, φ_{23}, and φ_{34}, utilizing inverse position kinematics, the subsequent step involves determining φ_{45}, φ_{56}, and φ_{67}. A notable advantage of

the 6R anthropomorphic arm with a spherical wrist lies in its capability to uphold the prescribed position of the end-effector (TCP) irrespective of the angular displacements exhibited by the last three joints, i.e., the position of the end-effector (TCP) remains *independent* of the spherical wrist, and is unambiguously determined by φ_{12}, φ_{23}, and φ_{34}. Consequently, the function of the spherical wrist predominantly revolves around achieving the desired orientation.

Consider a rotation matrix $\boldsymbol{R}_{1E}(\boldsymbol{\phi}_E)$ or $\boldsymbol{R}_{1\mathrm{TCP}}(\boldsymbol{\phi}_{\mathrm{TCP}})$ encompassing the desired orientation of the end-effector or the tool center point within $O_1x_1y_1z_1$, respectively. Since the Euler XYZ (RPY) angles are inputs of the inverse kinematics problem, this rotation matrix is known. Moreover, now that φ_{12}, φ_{23}, and φ_{34} are given, we are also provided with the rotation matrix $\boldsymbol{R}_{14}(\varphi_{12}, \varphi_{23}, \varphi_{34})$, relating the orientation of $O_4x_4y_4z_4$ within $O_1x_1y_1z_1$. As per Theorem 1.2, we can express either $\boldsymbol{R}_{1E}(\boldsymbol{\phi}_E)$ or $\boldsymbol{R}_{1\mathrm{TCP}}(\boldsymbol{\phi}_{\mathrm{TCP}})$ in the form

$$\boldsymbol{R}_{1C}(\boldsymbol{\phi}_C) = \boldsymbol{R}_{14}(\varphi_{12}, \varphi_{23}, \varphi_{34})\boldsymbol{R}_{4C},$$

from where we can obtain

$$\boldsymbol{R}_{4C}(\varphi_{12}, \varphi_{23}, \varphi_{34}, \boldsymbol{\phi}_C) = \boldsymbol{R}_{14}^T(\varphi_{12}, \varphi_{23}, \varphi_{34})\boldsymbol{R}_{1C}(\boldsymbol{\phi}_C), \tag{1.19}$$

assuming the end-effector or the TCP is the origin of the C-th coordinate system. The rotation from $O_4x_4y_4z_4$ to $O_Cx_Cy_Cz_C$ can also be obtained from the defined coordinate systems of the 6R anthropomorphic arm, i.e.,

$$\boldsymbol{R}'_{4C}(\varphi_{45}, \varphi_{56}, \varphi_{67}) = \prod_{i=4}^{J+1} \prod_{j>i}^{C} \boldsymbol{R}_{ij}(\varphi_{45}, \varphi_{56}, \varphi_{67}), \tag{1.20}$$

which is similar to (1.14). Seeing as (1.19) is ultimately independent of φ_{45}, φ_{56} and φ_{67}, juxtaposing it with (1.20) yields algebraic expressions for the last three joint angles, thereby finalising the closed-form inverse kinematics solution.

1.3.3 Differential Kinematics and the Jacobian

Following the derivation of both forward and inverse kinematics solutions, it is pertinent to address the phenomenon of *kinematic singularities*. These configurations denote states wherein the robot's motion exhibits heightened sensitivity to minor alterations in joint velocities. Kinematic singularities pose significant challenges to motion planning and control, potentially leading to suboptimal performance or even mechanical failure under extreme conditions. Consequently, a thorough understanding and characterization of these singularities are imperative for ensuring the safety and reliability of robotic systems.

Consider a forward kinematics solution of the form (1.14). Our aim is to find the relationship between the joint velocities $\dot{\boldsymbol{\varphi}} \in \mathbb{R}^J$ and linear and angular velocities

of the tool center point (TCP), denoted $\dot{\boldsymbol{p}}_{1\mathrm{TCP}}(\boldsymbol{\varphi},\dot{\boldsymbol{\varphi}}) \in \mathbb{R}^3$ and $\boldsymbol{\omega}_{1\mathrm{TCP}}(\boldsymbol{\varphi},\dot{\boldsymbol{\varphi}}) \in \mathbb{R}^3$, respectively, i.e., to obtain the mapping

$$\boldsymbol{\nu}_{\mathrm{TCP}}(\boldsymbol{\varphi},\dot{\boldsymbol{\varphi}}) \coloneqq \begin{bmatrix} \dot{\boldsymbol{p}}_{1\mathrm{TCP}}(\boldsymbol{\varphi},\dot{\boldsymbol{\varphi}}) & \boldsymbol{\omega}_{1\mathrm{TCP}}(\boldsymbol{\varphi},\dot{\boldsymbol{\varphi}}) \end{bmatrix}^T \mapsto \boldsymbol{J}_g(\boldsymbol{\varphi})\dot{\boldsymbol{\varphi}}, \tag{1.21}$$

where $\boldsymbol{J}_g(\boldsymbol{\varphi}) \in \mathbb{R}^{6\times\mathscr{D}}$ is termed the *geometric Jacobian* and is given by

$$\boldsymbol{J}_g(\boldsymbol{\varphi}) \coloneqq \begin{bmatrix} \boldsymbol{J}_p(\boldsymbol{\varphi}) \in \mathbb{R}^{3\times\mathscr{D}} \\ \boldsymbol{J}_\omega(\boldsymbol{\varphi}) \in \mathbb{R}^{3\times\mathscr{D}} \end{bmatrix} \in \mathbb{R}^{6\times\mathscr{D}},$$

where $\boldsymbol{J}_p(\boldsymbol{\varphi})$ and $\boldsymbol{J}_\omega(\boldsymbol{\varphi})$ satisfy the mappings

$$\dot{\boldsymbol{p}}_{1\mathrm{TCP}}(\boldsymbol{\varphi},\dot{\boldsymbol{\varphi}}) \mapsto \boldsymbol{J}_p(\boldsymbol{\varphi})\dot{\boldsymbol{\varphi}}, \tag{1.22}$$

$$\boldsymbol{\omega}_{1\mathrm{TCP}}(\boldsymbol{\varphi},\dot{\boldsymbol{\varphi}}) \mapsto \boldsymbol{J}_\omega(\boldsymbol{\varphi})\dot{\boldsymbol{\varphi}}, \tag{1.23}$$

for a $\mathscr{D}$-degree-of-freedom open-chain manipulator. [12, 15, 27, 29]

In order to compute the geometric Jacobian efficiently, it is advantageous to address the linear velocity, i.e., $\boldsymbol{J}_p(\boldsymbol{\varphi})$, and angular velocity, i.e., $\boldsymbol{J}_\omega(\boldsymbol{\varphi})$, separately. The contribution from the linear velocity $\dot{\boldsymbol{p}}_{1\mathrm{TCP}}(\boldsymbol{\varphi},\dot{\boldsymbol{\varphi}})$ reads

$$\dot{\boldsymbol{p}}_{1\mathrm{TCP}}(\boldsymbol{\varphi},\dot{\boldsymbol{\varphi}}) = \sum_{i=1}^{J}\sum_{j>i}^{J+1} \frac{\partial \boldsymbol{p}_{1\mathrm{TCP}}(\boldsymbol{\varphi})}{\partial \varphi_{ij}}\dot{\varphi}_{ij} \tag{1.24}$$

and by reorganising (1.24) to the form

$$\dot{\boldsymbol{p}}_{1\mathrm{TCP}}(\boldsymbol{\varphi},\dot{\boldsymbol{\varphi}}) \mapsto \begin{bmatrix} j_{p_{11}} & \cdots & j_{p_{1\mathscr{D}}} \\ \vdots & \ddots & \vdots \\ j_{p_{31}} & \cdots & j_{p_{3\mathscr{D}}} \end{bmatrix} \begin{bmatrix} \dot{\varphi}_{12} \\ \vdots \\ \dot{\varphi}_{J,J+1} \end{bmatrix}, \tag{1.25}$$

one obtains precisely (1.22). Assuming, the tool center point is part of the last, U-th link of the manipulator in question, the contribution from the angular velocity $\boldsymbol{\omega}_{1\mathrm{TCP}}(\boldsymbol{\varphi},\dot{\boldsymbol{\varphi}})$ is then given by

$$\boldsymbol{\omega}_{1\mathrm{TCP}}(\boldsymbol{\varphi},\dot{\boldsymbol{\varphi}}) = \sum_{u=1}^{U} \boldsymbol{\omega}_{u-1,u} = \sum_{i=1}^{J}\sum_{j>i}^{J+1} \dot{\varphi}_{ij}\boldsymbol{k}_j, \tag{1.26}$$

since J+1 has to be equal to U (see, e.g., Figure 1.10). Again, reorganising (1.26) to the form similar to (1.25), it yields (1.23). [12, 15]

Kinematic singularities occur when $\boldsymbol{J}_g(\boldsymbol{\varphi})$ is *rank deficient*, i.e., when

$$\mathrm{rank}[\boldsymbol{J}_g(\boldsymbol{\varphi})] < \min(6,\mathscr{D}),$$

or alternatively, if $\boldsymbol{J}_g(\boldsymbol{\varphi})$ is *square*,[22]

$$\det[\boldsymbol{J}_g(\boldsymbol{\varphi})] = 0,$$

which is computed, e.g., via the cofactor expansion. [15, 32]

[22]The geometric Jacobian becomes square for a 6-degree-of-freedom arm.

1.4 Static Gravity Compensation

As gravity constitutes an intrinsic force within the fabric of nature, its influence on the robotic system remains inherent, inevitably inducing deviations from the intended pose of the end-effector or the tool center point, as prescribed by the inverse kinematics solution, requiring a correction of the elements of the vector of joint coordinates $\boldsymbol{\varphi} \in \mathbb{R}^J$. When the aim is to solely attain equilibrium using some counter-torque to balance out gravitational torque, irrespective of the robot's dynamics, the process is referred to as *static gravity compensation*.

1.4.1 Torsion Spring Compensation

One approach to account for gravitational effects is to use *torsion spring compensation*, i.e., attach torsion springs with stiffness κ_τ [N m rad^{-1}] and dampers with damping coefficient b [N m s rad^{-1}] to each joint i and alter their equilibrium position $\varphi_{ij} \in \boldsymbol{\varphi}$ by some angle $\Delta\varphi_{ij}$ to initiate a counter-torque acting against gravitational torque and thereby create equilibrium.

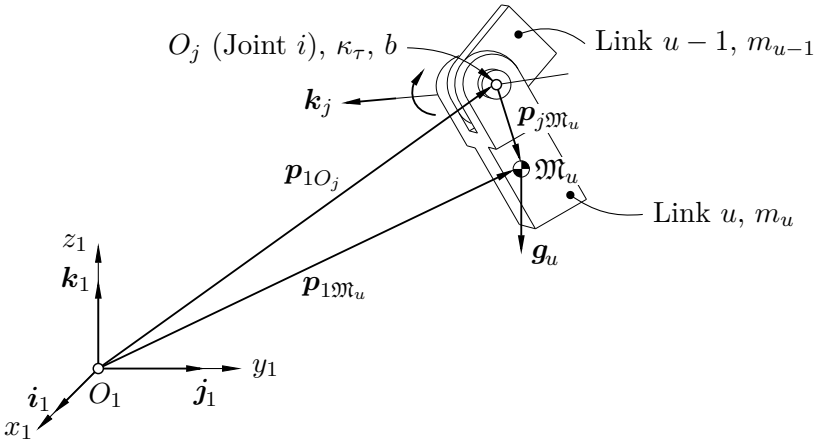


Figure 1.13 Static gravity compensation.

Let us consider an open kinematic chain composed of $J \in \mathbb{N}$ revolute joints and $U \in \mathbb{N}$ links, each of mass m_u. The equilibrium of joint i reads

$$\kappa_\tau \Delta\varphi_{ij} + g \sum_{i}^{J} \sum_{u>i}^{U} \det(\boldsymbol{T}_{ui}) m_u = 0, \tag{1.27}$$

where $g \approx 9.806\,65$ [m s^{-2}] is the gravitational acceleration and $\boldsymbol{T}_{ui} \in \mathbb{R}^{3\times3}$ denotes the *gravitational torque matrix* of joint i, given by [34]

$$\boldsymbol{T}_{ui} := \begin{bmatrix} k_{j_x} & k_{j_y} & k_{j_z} \\ p_{1\mathfrak{M}_{u_x}} & p_{1\mathfrak{M}_{u_y}} & p_{1\mathfrak{M}_{u_z}} \\ 0 & 0 & -1 \end{bmatrix} \in \mathbb{R}^{3\times3}, \quad \forall\, j = i+1,$$

if $\langle \boldsymbol{g}_u, \boldsymbol{k}_1 \rangle < 0$ for gravitational forces $\boldsymbol{g}$ of all links u. The vector $\boldsymbol{p}_{1\mathfrak{M}_u}$ denotes the position vector from the origin of $O_1x_1y_1z_1$ to the center of mass $\mathfrak{M}$ of link u and the axis of revolution of joint i is denoted $\boldsymbol{k}_j$, as $O_1x_1y_1z_1$ is reserved for the frame. The elements of $\boldsymbol{T}_{ui}$ can, in general form, be computed as

$$\begin{aligned} \boldsymbol{k}_j &= \boldsymbol{R}_{1j}\boldsymbol{k}_1, \\ \boldsymbol{p}_{1\mathfrak{M}_u} &= \boldsymbol{R}_{1u}\boldsymbol{p}_{u\mathfrak{M}_u} + \sum_{\substack{u>j \\ u\neq U}}^{U} \boldsymbol{R}_{1j}\boldsymbol{p}_{jO_u}. \end{aligned}$$

The gravity compensation algorithm start from the last, J-th joint of the robot, for which, from (1.27), we have

$$\Delta\varphi_{J,J+1} = -\frac{g}{\kappa_\tau}\sum_{u>J}^{U} \det(\boldsymbol{T}_{u,J+1})m_u,$$

which, for the 6R anthropomorphic arm, becomes

$$\Delta\varphi_{67} = -\frac{g}{\kappa_\tau}\det(\boldsymbol{T}_{77})m_7,$$

wherein

$$\boldsymbol{k}_7 = \boldsymbol{R}_{17}\boldsymbol{k}_1, \quad \boldsymbol{p}_{1\mathfrak{M}_7} = \boldsymbol{R}_{17}\boldsymbol{p}_{7\mathfrak{M}_7}.$$

As for joint 5, (1.27) reads

$$\Delta\varphi_{56} = -\frac{g}{\kappa_\tau}\sum_{i=5}^{6}\sum_{u>i}^{7} \det(\boldsymbol{T}_{ui})m_u = -\frac{g}{\kappa_\tau}\Big[\det(\boldsymbol{T}_{65})m_6 + \det(\boldsymbol{T}_{75})m_7\Big],$$

for which

$$\boldsymbol{k}_6 = \boldsymbol{R}_{16}\boldsymbol{k}_1, \quad \boldsymbol{p}_{1\mathfrak{M}_6} = \boldsymbol{R}_{16}\boldsymbol{p}_{6\mathfrak{M}_6}, \quad \boldsymbol{p}_{1\mathfrak{M}_7} = \boldsymbol{R}_{17}\boldsymbol{p}_{7\mathfrak{M}_7} + \boldsymbol{R}_{16}\boldsymbol{p}_{6O_7},$$

and so on up to joint 1. All solutions for $\Delta\varphi_{ij}$ form a vector of joint compensations,

$$\Delta\boldsymbol{\varphi} = \begin{bmatrix} \Delta\varphi_{12} & \cdots & \Delta\varphi_{J,J+1} \end{bmatrix} \in \mathbb{R}^J,$$

which, when added to the vector of joint coordinates $\boldsymbol{\varphi} \in \mathbb{R}^J$, corrects the inverse kinematics solution to account for gravitational effects and ultimately leads in achieving the desired pose of the end-effector or the tool center point.

Remark 1.6. This approach can also be interpreted as the motors actuating each joint being stationary under compensation (hence *static* gravity compensation), and us modelling the electromechanical stiffness behaviour of the motors and the transmissions using these mechanical springs and dampers. ◇

2
Simscape MBS Modelling

Simscape Multibody, formerly recognized as SimMechanics, stands as a robust tool embedded within the MATLAB and Simulink ecosystem. Tailored for the modeling and simulation of multi-domain physical systems, this software empowers engineers and researchers to meticulously craft mechanical models ranging from elementary mechanisms to sophisticated mechatronic systems. By leveraging Simscape MBS (multibody systems), users can effortlessly design mechanical components, joints, and constraints, thereby facilitating precise simulations of dynamic motion, forces, and torques. Notably, its user-friendly interface and extensive library of pre-assembled components streamline the modeling process, while its seamless integration with Simulink ensures smooth analysis and design of control systems. Ultimately, the utilization of Simscape MBS enables users to delve deeply into the intricacies of mechanical systems, refining performance and expediting the development of innovative designs.

This chapter heavily relies on sources such as [35, 36, 37, 38] in its attempt to develop simple yet effective theoretical foundation into using Simscape MBS, discussing mainly the function of commonly used blocks and the software's relation to MATLAB and Simulink, ultimately resulting in a symbiosis of all three tools. Similarly to the previous chapter, may the well-versed reader feel free to skip this portion of the thesis and come back as they find necessary.

2.1 Introduction to Multi-Domain Modelling

Before delving into specifics of individual blocks within Simscape MBS, the author feels the need to briefly address some preliminary subjects in multi-domain modelling. Upon initiating MATLAB, the user simply writes `simulink` in the command window and the Simulink landing page appears. As Simscape MBS is an

embedded part of Simulink, creating a new multibody model is done by creating a new Simulink model. On the now visible blank canvas, users can start importing blocks from Simulink's library, specifically from the "Simscape" category.

2.1.1 Block Wiring and Signal Types

The blocks in Simscape MBS (or plain Simulink for that matter) need to be connected together in an order the mechanism is composed. This connecting is referred to as *block wiring* and is done by dragging from the output of one block into the input of another. This action is represented graphically as Simulink creates a line between the blocks. Each block has a different number of input/output ports, depending on its use case. Most commonly, blocks in Simscape MBS have *two* ports, usually representing coordinate systems yet more ports can sometimes be added, e.g., in a scenario where more coordinate systems are needed.

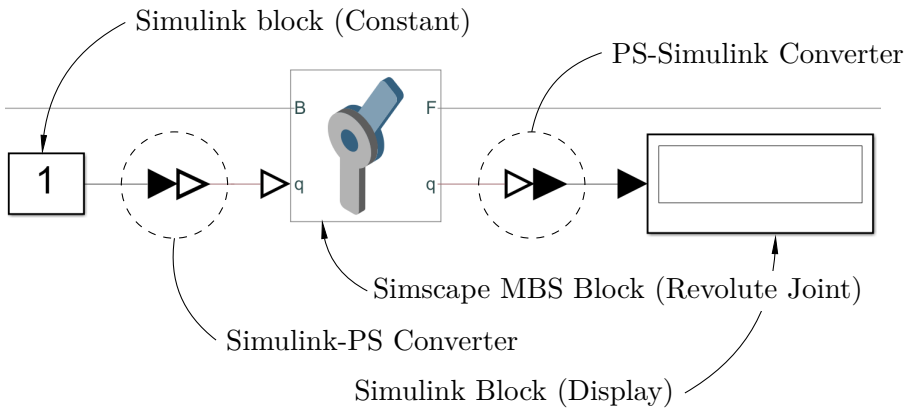


Figure 2.1 Simulink/Simscape MBS signal types.

The connection of two blocks is dependent on if they represent the same *signal type*. Simulink blocks not drawn from the "Simscape" library category, such as displays, "Gain" blocks, "Mux" and "Demux" blocks, etc., are of the *Simulink signal type*. On the other hand, *all* blocks from the "Simscape" library category are of the *physical system signal type* (PS), hence some sort of "adapter" is needed to convert one signal type to another and vice versa.

This conversion is achieved by virtue of the "Simulink-PS Converter" and "PS-Simulink Converter" blocks (see Figure 2.1). Both blocks have only one input and output ports under all circumstances. In the case of the former mentioned block, the Simulink-type signal is fed to its input, while its output is of PS-type signal and can be fed only to a block with PS-type signal inputs. On the contrary, the latter mentioned block takes a PS-type signal as its input and outputs a Simulink-type signal for further input into Simulink blocks.

2.1.2 Signal Rerouting, Distribution and Merging

A signal does not need to follow only the output-input behaviour. In many instances, it is the case the user needs to feed the same signal, containing the same information, to various places at once, i.e., convert one output to multiple inputs. Doing so is as easy as right-clicking on a chosen output signal and dragging it to another input of the same signal type (PS/Simulink).[23]

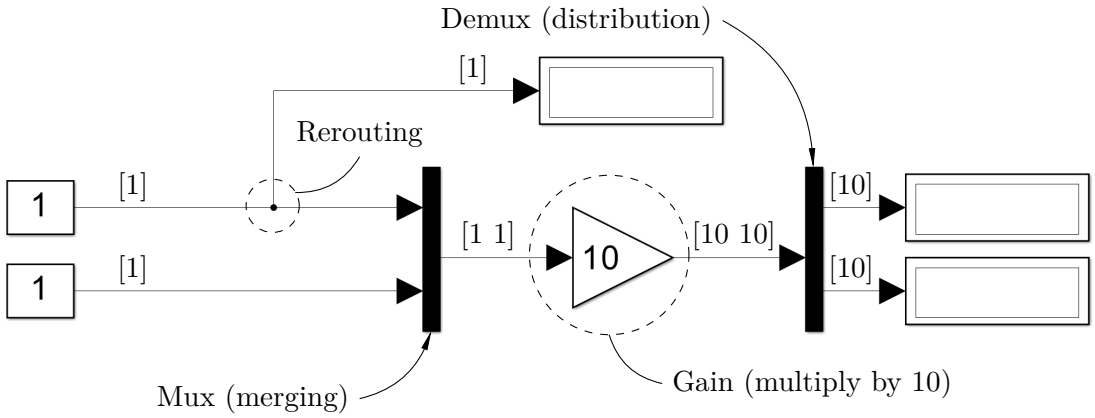


Figure 2.2 Rerouting, distribution, and merging of signals.

In scenarios when the output is of, e.g., vector form, there may be a desire to distribute the signal into individual vector components. This can be done by first feeding the signal through the "Demux" block which takes one input and outputs n signals depending on the dimension of the array on input.[24]

In contrast, when the user would like to feed multiple outputs into one input of, e.g., a graphical interface such as the "Scope" block, the "Mux" block is needed. Its input are n signals and on the output side is a n-dimensional array.

2.2 Commonly Used Blocks

Given most common Simulink blocks have already been established, e.g., "Mux", "Display", etc., let us only turn our attention to Simscape MBS blocks as the main structure of a robotic system is modelled using these blocks.

2.2.1 Preliminary Blocks

Each Simscape MBS model has to start with some preliminary blocks, commencing the model and providing a reference. These include the "World Frame", "Mechanism Configuration", and "Solver Configuration" blocks (see Fig. 2.3).

[23] If the input requires a signal of different type, one uses a converter.
[24] In the context of Fig. 2.2, the array is a 1×2 vector.

▷ **World Frame.** Advancing from left to right, the "World Frame" block does not have any options dialogue box as it only serves as the overarching coordinate system $O_1x_1y_1z_1$. This means it is orthogonal and right-handed. The port `W` is connected, together with the "Mechanism Configuration" and "Solver Configuration" blocks, to the first solid within the model.

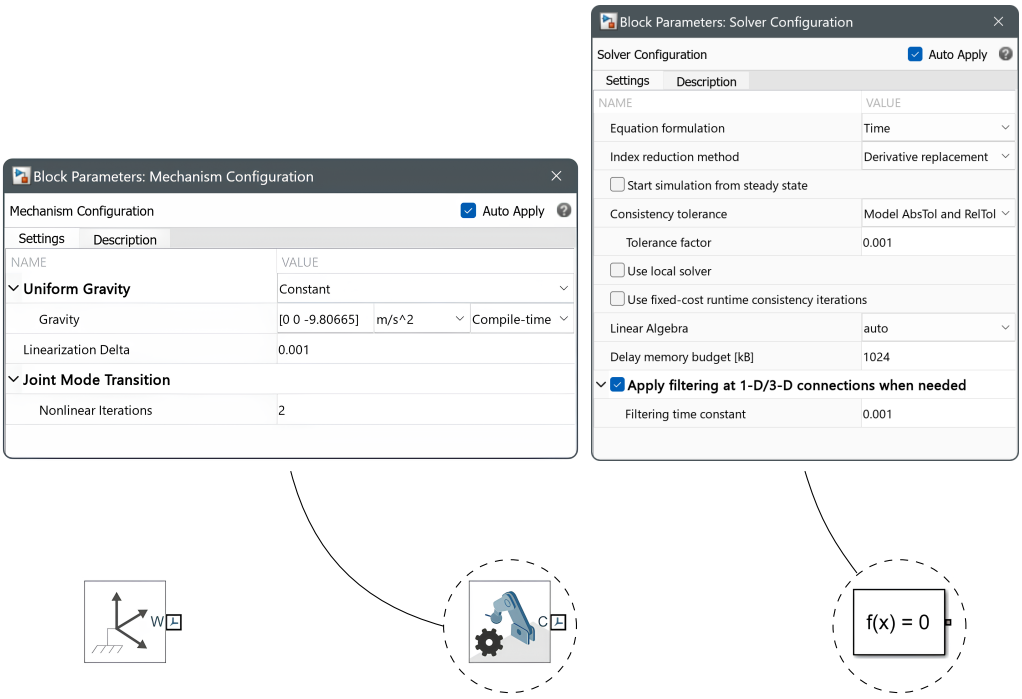


Figure 2.3 Preliminary Simscape MBS blocks.

▷ **Mechanism Configuration.** The middle block in Figure 2.3 is the "Mechanism Configuration" block. It is used to define global mechanical settings for the model, such as uniform gravity. Its `C` port is connected, together with the "World Frame" and "Solver Configuration" blocks, to the first solid within the model.

▷ **Solver Configuration.** The "Solver Configuration" block (rightmost block in Figure 2.3) is used to set global solver settings for simulation. Its output port is connected, together with the "World Frame" and "Mechanism Configuration" block, to the first solid within the model.

2.2.2 Solids and Joints

Having now described the necessary preliminary blocks, the main portion of the mechanism is composed of solids and joints, to which we turn now. In the case of solids, there exists a considerably large number of options. Simscape MBS has a built-in modeller, with which one can model simple shapes of cubic, cylindrical

and other topologies. Given the topological intricacy of the model we'll be using (see Part II), the best option we are left with is the "File Solid" block.

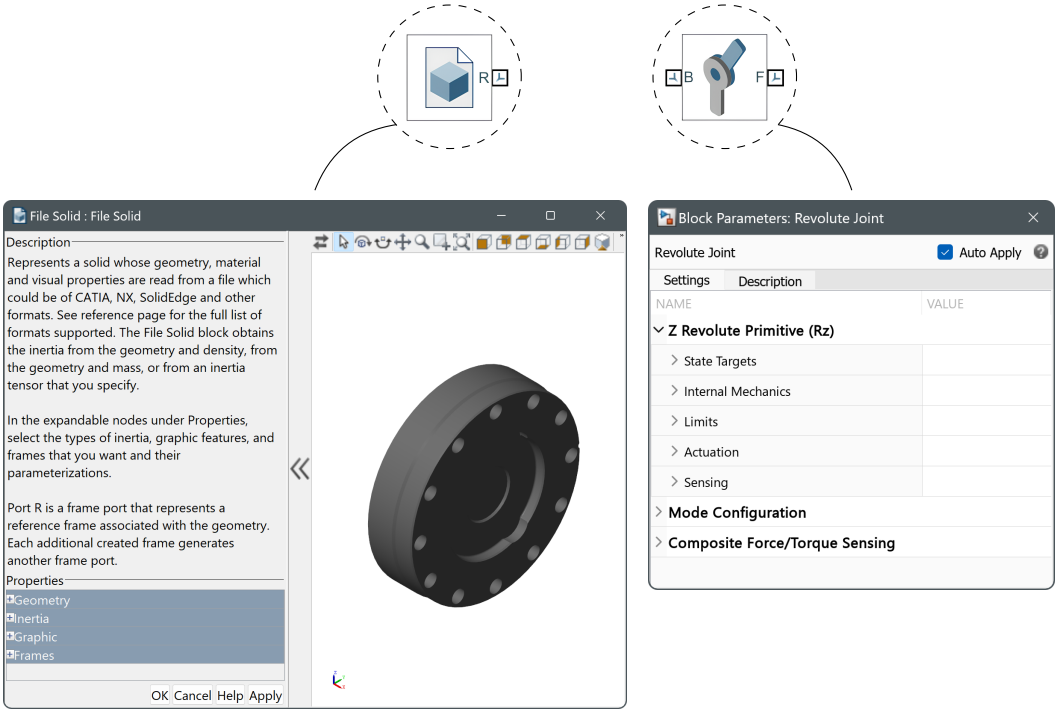


Figure 2.4 Common Simscape MBS solids and joints.

▷ **File Solid.** The "File Solid" block (Figure 2.4) serves as an extension to the Simscape MBS embedded modeller as it enables users to input any file given they provide its path in the "Geometry" option on the left hand side of the block's configuration dialogue box. Under the "Inertia" and "Graphic" options, users can, similarly to the Simscape MBS embedded modeller, define either density of the component or mass, and let software determine other inertial parameters such as center of mass coordinates, moments of inertia, etc. Moreover, the appearance of the part can be changed under the "Graphic" option and more coordinate systems can be added under the "Frames" option, extending the block's input/output ports beyond the reference port `R`, usually situated at the center of mass $\mathfrak{M}$.

The solids are usually connected through joints as on the actual robot.[25] Given our attention resides on the 6R anthropomorphic arm with a spherical wrist (extensively showcased in Chapter 1), the most important block in our use case is the "Revolute Joint" block (depicted in Figure 2.4 right). On the other hand, all joint types depicted in Figure 1.1 can be used in Simscape MBS to model different types of robotic systems or mechanisms.

[25] An exception may be made for rigidly connected parts, e.g., via screws or pins.

▷ **Revolute Joint.** The "Revolue Joint" block has two ports by default, the base port `B` and the follower port `F`, i.e., Simscape MBS uses the same terminology we have introduced in Chapter 1. This means that, if one has two solids (robot links), i and $i+1$, the revolute joint i has its `B` port connected to the output coordinate system of solid i, whilst its `F` port connects to the input coordinate system of solid $i+1$. The axis of revolution of this joint *has* to be the z-axis, somewhat constraining the model's coordinate system definition (see Chapter 5 in Part II). Configuring the joint mainly involves setting its state targets, i.e., desired angle and, optionally, velocity; its internal mechanics, i.e., torsion spring stiffness and damping coefficient (see Chapter 5), and actuation settings for its motion and torque, which can be either provided by input, automatically computed, and, as for torque, there can be none. Additionally, one can set limits for the angle of revolution and sense forces, torques, etc. in the joint during simulation.

2.2.3 Coordinate Transformations

In many instances, the user will need to perform a transformation from one, already defined, coordinate system to another without specifically defining this system within, e.g., the "File Solid" block. This transformation is realised by virtue of the "Rigid Transform" block (see Figure 2.5).

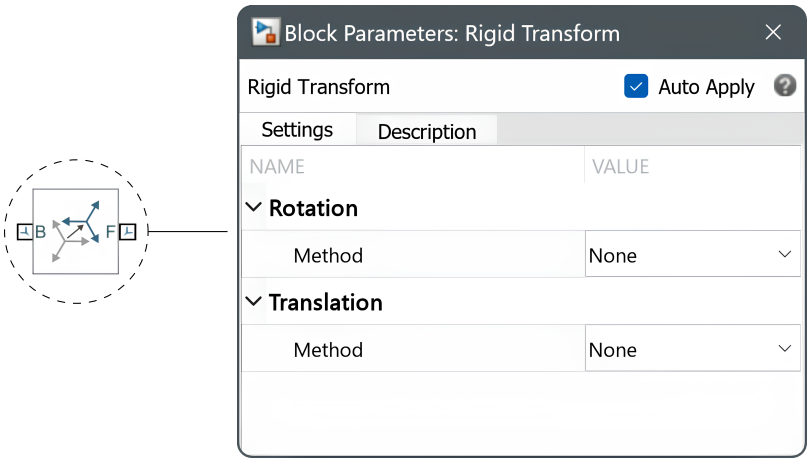


Figure 2.5 Coordinate transformation within Simscape MBS.

▷ **Rigid Transform.** The "Rigid Transform" block transforms one coordinate system, situated at the `B` port, into another at its `F` port. The user can specify rotation of the follower system along with its translation and has multiple options to do so, e.g., rotation matrix, standard axis, quaternion, etc., as for rotation; and Cartesian, standard axis, and cylindrical, as for translation. The resulting coordinate system remains orthogonal and right handed.

2.2.4 Forces, Torques and Measuring

To apply load on a system, e.g., force on the tool center point of a robot, one can use the "External Force and Torque" block, where this load can be configured as desired. As this block is not an essential building block of Simscape MBS, the author refers the reader to [37, 38] for further insight.

Besides the "External Force and Torque" block, one may sometimes desire to measure either dimensions, such as distances, angles, etc., or inertial parameters, e.g., mass, moments of inertia, etc., of the system. For these purposes, Simscape MBS provides the "Transform Sensor" and "Inertia Sensor" blocks, respectively. As the need to measure is not always present, the author kindly asks the reader to again consult [37, 38] for further information. Yet, our specific use case of these blocks is briefly described in Part II, Chapter 5.

2.3 Interfacing Simscape MBS with MATLAB

To conclude this chapter, let us outline how the collaboration between MATLAB and Simscape MBS works. Consider a multibody model within Simscape MBS and some MATLAB script, e.g., used to calculate joint coordinates of the revolute joints within the model. By default, these coordinates, and any variables for that matter, appear in the MATLAB workspace upon running the script.

- ▷ **From MATLAB to Simscape MBS.** Now, the user has two options on how to interface these variables into the model. Firstly, utilizing the "From Workspace" block facilitates the loading of selected variables into Simscape MBS, albeit requiring connection to an appropriate block within the model, see [36]. Alternatively, certain blocks, such as revolute joints, the "Rigid Transform" block, etc., permit direct incorporation of these variables as configuration parameters through the block configuration dialogue box. The latter approach offers a more streamlined process, enabling users to input variable names directly into the dialogue box. However, both approaches necessitate prior execution of the MATLAB script before simulation commencement.
- ▷ **From Simscape MBS to MATLAB.** Going the other way requires the "To Workspace" block, which involves assigning variable names to signals and transmitting them back to the MATLAB workspace upon initiating the simulation of the model. As the reader shall see later, our model heavily relies on this block in terms of its static gravity compensation.

3
Nonlinearity

Given most real mechanical structures, e.g., robotic systems, especially spatial ones, are often of great complexity, they exhibit highly nonlinear behaviour within their work envelope. Moreover, this nonlinearity tends to gets worse as the system gets more complicated. This phenomena restrains engineers in their attempts of finding, e.g., the equations of motion of the system, and thereby predicting the system's future movement, velocity, etc., as finding such equations requires solving nonlinear differential equations and is thereby unattainable in closed-form.

The final theoretical chapter of this thesis undertakes a formal effort to establish a fundamental connection between the state-space representation of a linearized system and its static stiffness. By delving into the foundational principles of control theory and dynamical systems, which encompass key ideas such as the Laplace transform and transfer function matrices, a theoretical framework is constructed. After thoroughly exploring these fundamental concepts, the discussion smoothly transitions into a methodical derivation of the aforementioned relationship. This derivation is meticulously carried out by employing singular value decomposition (SVD) on the transfer function of the linearized system, thus clarifying the intricate relationship between system linearity and static stiffness.

3.1 State-Space Representation

The state-space representation can be understood as a mathematical model of a physical system, defined by a set of input, output, and general variables related by first-order differential, or difference equations. These general variables, termed the *state variables*, dynamically evolve over time contingent upon the input variables of the system. While the output variables are generally dependent on the state

variables, it is noteworthy that in certain scenarios, their reliance extends to both input and state space variables of the given system. [39, 40, 41, 42]

3.1.1 Linearization

To derive the state-space representation, it is imperative to transform all higher-order differential equations governing the system into first-order equations through a process of *linearization*. This involves expanding all nonlinear terms within the system's equations using the Taylor expansion, as extensively detailed in Appendix A, and truncating the expansion at the linear term, i.e., the first order. A fundamental tenet of calculus dictates that the Taylor expansion of a function yields acceptable accuracy only in the vicinity of the expansion point. Consequently, the resultant linearized mathematical description, does not faithfully capture the system's behaviour across its entire domain. Should a change in the system's operating point be required, it becomes necessary to reiterate the linearization process to ensure consistency. As a result, for a system to be pronounced linear, it *must* satisfy Definition 3.1 in all cases. [39, 43]

Definition 3.1. A system $\mathcal{S}$ is considered *linear* if it responds to a *linear* combination of its inputs by the *same* linear combination on its output. [41] ◇

Definition 3.2. A system $\mathcal{S}$ is *time-invariant* if for any input signal $\psi(t)$, the output signal $\varsigma(t)$ satisfies the condition

$$\varsigma(t) = \mathcal{S}[\psi(t)] \equiv \varsigma(t+\tau) = \mathcal{S}[\psi(t+\tau)], \quad \forall\, \tau \in \mathbb{R},$$

i.e., the output is independent of the time at which the input came. [40, 41] ◇

Let $\boldsymbol{\xi}(t) \in \mathbb{R}^n$ represent the state vector of $\mathcal{S}$ at time t. Provided $\mathcal{S}$ is linear, time-invariant and finite-dimensional, its state-space representation reads

$$\begin{aligned} \dot{\boldsymbol{\xi}}(t) &= \boldsymbol{A}\boldsymbol{\xi}(t) + \boldsymbol{B}\boldsymbol{\psi}(t), \\ \boldsymbol{\varsigma}(t) &= \boldsymbol{C}\boldsymbol{\xi}(t) + \boldsymbol{D}\boldsymbol{\psi}(t), \end{aligned}$$

where $\boldsymbol{A} \in \mathbb{R}^{n\times n}$ represents the dynamics of the system, $\boldsymbol{B} \in \mathbb{R}^{n\times m}$ represents the effect of inputs on the system, $\boldsymbol{C} \in \mathbb{R}^{p\times n}$ represents the the relationship between the state and the output, and $\boldsymbol{D} \in \mathbb{R}^{p\times m}$ represents the direct influence of inputs on the output, for n-dimensional state space, m-dimensional input space, and p-dimensional output space. [39, 41, 43]

3.1.2 Transfer Function Matrix

Consider a state space representation of a linearized system $\mathcal{S}$. The output vector $\boldsymbol{\varsigma}(t) \in \mathbb{R}^p$ of $\mathcal{S}$ can be expressed in the form

$$\boldsymbol{\Sigma}(s) = \boldsymbol{G}(s)\boldsymbol{\Psi}(s),$$

so that

$$\boldsymbol{G}(s) = \frac{\boldsymbol{\Sigma}(s)}{\boldsymbol{\Psi}(s)},$$

where the matrix operator $\boldsymbol{G}(s)$ is referred to as the *transfer function matrix*, relating the inputs and outputs of $\mathcal{S}$ in the *Laplace domain*. The Laplace domain is a product of the *Laplace transform*, transforming a function of time t into a function of a complex variable s, termed the *Laplace operator* (see Definition 3.3), which allows for an easier analysis of differential equations as they become algebraic ones within this specific framework. [40, 41]

Definition 3.3. The *Laplace transform* of a function $f(t) : \mathbb{R}_0^+ \mapsto \mathbb{C}$ is defined as

$$\mathcal{L}\{f\}(s) \coloneqq \int_0^{+\infty} f(t) \exp(-st)\, \mathrm{d}t,$$

wherein $s = s_1 + is_2 \in \mathbb{C}$, $s_1, s_2 \in \mathbb{R}$, $i \coloneqq \sqrt{-1}$. [41, 43] ◇

Remark 3.1. The Laplace transform is defined only for a specific set of functions, referred to as the *domain of Laplace transformation*. Essentially, the function $f(t) : \mathbb{R}_0^+ \mapsto \mathbb{C}$ has to be integrable on $\mathtt{I} = [0, a]$ for all $a > 0$ and

$$\int_0^{+\infty} f(t) \exp(-st)\, \mathrm{d}t < +\infty$$

as $t \to +\infty$ for some $s \in \mathbb{C}$, i.e., the integral converges. [41] ◇

As for the state-space description, taking the Laplace transform yields

$$s\boldsymbol{\Xi}(s) - \boldsymbol{\xi}_0 = \boldsymbol{A}\boldsymbol{\Xi}(s) + \boldsymbol{B}\boldsymbol{\Psi}(s) \tag{3.1a}$$
$$\boldsymbol{\Sigma}(s) = \boldsymbol{C}\boldsymbol{\Xi}(s) + \boldsymbol{D}\boldsymbol{\Psi}(s) \tag{3.1b}$$

where $\boldsymbol{\xi}_0 \equiv \boldsymbol{\xi}(0)$ is the initial state of the system. From (3.1a) we have

$$(s\boldsymbol{I}_n - \boldsymbol{A})\boldsymbol{\Xi}(s) = \boldsymbol{\xi}_0 + \boldsymbol{B}\boldsymbol{\Psi}(s),$$

and rearranging gives us

$$\boldsymbol{\Xi}(s) = (s\boldsymbol{I}_n - \boldsymbol{A})^{-1}\boldsymbol{\xi}_0 + (s\boldsymbol{I}_n - \boldsymbol{A})^{-1}\boldsymbol{B}\boldsymbol{\Psi}(s).$$

Now, substituting $\boldsymbol{\Xi}(s)$ into (3.1b) yields

$$\boldsymbol{\Sigma}(s) = \boldsymbol{C}[(s\boldsymbol{I}_n - \boldsymbol{A})^{-1}\boldsymbol{\xi}_0 + (s\boldsymbol{I}_n - \boldsymbol{A})^{-1}\boldsymbol{B}\boldsymbol{\Psi}(s)] + \boldsymbol{D}\boldsymbol{\Psi}(s),$$

by letting $\boldsymbol{\xi}_0 = \boldsymbol{0}_n$ and distributing $\boldsymbol{C}$ we get,

$$\boldsymbol{\Sigma}(s) = \boldsymbol{C}(s\boldsymbol{I}_n - \boldsymbol{A})^{-1}\boldsymbol{B}\boldsymbol{\Psi}(s) + \boldsymbol{D}\boldsymbol{\Psi}(s),$$

and dividing by $\boldsymbol{\Psi}(s)$, we finally arrive at

$$\boldsymbol{G}(s) = \boldsymbol{D} + \boldsymbol{C}(s\boldsymbol{I}_n - \boldsymbol{A})^{-1}\boldsymbol{B}, \tag{3.2}$$

which can be rewritten to the form

$$\boldsymbol{G}(s) = \boldsymbol{D} + \boldsymbol{C}\frac{1}{\det(s\boldsymbol{I}_n - \boldsymbol{A})}\mathrm{adj}(s\boldsymbol{I}_n - \boldsymbol{A}),$$

wherein $\mathrm{adj}(s\boldsymbol{I}_n - \boldsymbol{A})$ is the *adjoint* of $[s\boldsymbol{I}_n - \boldsymbol{A}]$. The determinant $\det(s\boldsymbol{I}_n - \boldsymbol{A})$ is termed the *characteristic polynomial* of the system and its roots are used to determine the stability of $\mathcal{S}$. [40, 41, 43, 44]

3.2 Singular Value Decomposition

Singular value decomposition (SVD) is a powerful technique of linear algebra, used mainly for data analysis, where it allows for a simpler representation of a large dataset, capturing its essential features and reducing dimensionality. [32, 33]

Theorem 3.1. Let $\boldsymbol{A} \in \mathbb{R}^{m\times n}$ be a real-valued matrix such that $\mathfrak{r} = \mathrm{rank}(\boldsymbol{A})$ satisfies $\mathfrak{r} \leq \min(m, n)$. The factorization of $\boldsymbol{A}$ which reads

$$\boldsymbol{A} = \boldsymbol{U}\boldsymbol{\Lambda}\boldsymbol{V}^T$$

in which $\boldsymbol{U} \in \mathbb{R}^{m\times m}$, $\boldsymbol{V} \in \mathbb{R}^{n\times n}$ are orthonormal matrices of the form

$$\boldsymbol{U} = \begin{bmatrix} | & | & & | \\ \boldsymbol{u}_1 & \boldsymbol{u}_2 & \cdots & \boldsymbol{u}_m \\ | & | & & | \end{bmatrix} \in \mathbb{R}^{m\times m}, \quad \boldsymbol{V} = \begin{bmatrix} | & | & & | \\ \boldsymbol{v}_1 & \boldsymbol{v}_2 & \cdots & \boldsymbol{v}_n \\ | & | & & | \end{bmatrix} \in \mathbb{R}^{n\times n}$$

and $\boldsymbol{\Lambda} \in \mathbb{R}^{m\times n}$ is a *pseudo-diagonal* matrix of the form

$$\boldsymbol{\Lambda} = \begin{bmatrix} \sqrt{\lambda_1} & & & & & \\ & \ddots & & & & \\ & & \sqrt{\lambda_\mathfrak{r}} & & & \\ & & & 0 & & \\ & & & & \ddots & \\ & & & & & 0 \end{bmatrix} \in \mathbb{R}^{m\times n},$$

where $\lambda_1 \geq \lambda_2 \geq \cdots \geq \lambda_\mathfrak{r} > 0$ are the *eigenvalues* of $\boldsymbol{A}$, is referred to as the *singular value decomposition* of $\boldsymbol{A}$. The non-zero diagonal entries of $\boldsymbol{\Lambda}$ are termed the *singular values* of $\boldsymbol{A}$, the columns of $\boldsymbol{U}$ are called the *left singular vectors* of $\boldsymbol{A}$, and the rows of $\boldsymbol{V}^T$ are the *right singular vectors* of $\boldsymbol{A}$. [15, 32, 45, 46, 47] $\diamond$

Remark 3.2. In cases when $\boldsymbol{A}$ is *rectangular*, i.e., when $m \neq n$, the singular values $\sqrt{\lambda}_1, \ldots, \sqrt{\lambda}_\mathfrak{r}$ are computed from the eigenvalues of $\boldsymbol{A}^T\boldsymbol{A}$. [45, 47] ◇

Remark 3.3. The singular value decomposition concept can be extended to the complex domain, where, for any complex-valued matrix $\boldsymbol{A} \in \mathbb{C}^{m\times n}$, it reads

$$\boldsymbol{A} = \boldsymbol{U}\boldsymbol{\Lambda}\boldsymbol{V}^H,$$

wherein $\boldsymbol{U} \in \mathbb{C}^{m\times m}$ and $\boldsymbol{V} \in \mathbb{C}^{n\times n}$ are unitary matrices, $\boldsymbol{\Lambda} \in \mathbb{C}^{m\times n}$ is again a pseudo-diagonal matrix, and $\boldsymbol{V}^H$ denotes the *Hermitian transpose*, sometimes also referred to as the *conjugate transpose*, of $\boldsymbol{V}$. [45, 46, 47] ◇

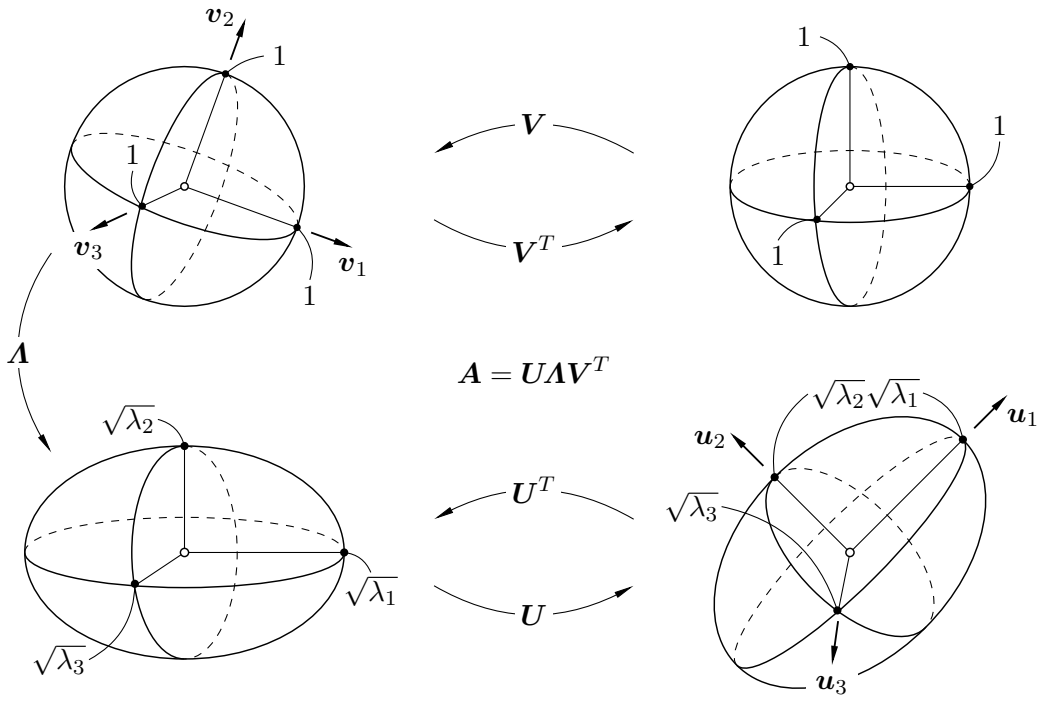


Figure 3.1 Singular value decomposition.

Definition 3.4. The *domain* of a transformation is the set of all possible input values $\boldsymbol{\psi} \in \mathbb{R}^m$ for which the transformation is defined, i.e.,

$$\mathrm{dom}(\boldsymbol{\mathcal{V}}) = \{\boldsymbol{\psi} \mid \boldsymbol{\psi} \in \mathbb{R}^m\},$$

for a linear transformation $\boldsymbol{\mathcal{V}}$. [25, 26, 47, 48] ◇

Definition 3.5. The *range* of $\boldsymbol{\mathcal{V}}$ is the set of all possible output values $\boldsymbol{\mathcal{V}}$ can yield for the input values from its domain, i.e.,

$$\mathrm{range}(\boldsymbol{\mathcal{V}}) = \{\boldsymbol{\varsigma} \mid \boldsymbol{\varsigma} = \boldsymbol{\mathcal{V}}(\boldsymbol{\psi}) \in \mathbb{R}^p,\, \boldsymbol{\psi} \in \mathrm{dom}(\boldsymbol{\mathcal{V}})\},$$

where $\boldsymbol{\varsigma} \in \mathbb{R}^p$ is the output produced by applying $\boldsymbol{\mathcal{V}}$ onto $\boldsymbol{\psi}$. [25, 26, 47, 48] ◇

The domain space of $\boldsymbol{A}$, $\mathrm{dom}(\boldsymbol{A})$, is represented by the columns of $\boldsymbol{U}$, which form the basis for the space of all possible input vectors $\boldsymbol{A}$ can operate on. On the other hand, the columns of $\boldsymbol{V}$ represent the range space of $\boldsymbol{A}$, $\mathrm{range}(\boldsymbol{A})$, and form a basis for the space spanned by all possible output vectors that can be produced by applying $\boldsymbol{A}$ to vectors from its domain space. [33, 45, 47]

The geometric intuition behind singular value decomposition in 3D space can be interpreted as shown within Figure 3.1. If we consider $\boldsymbol{U}$ and $\boldsymbol{V}$ to be square, they can be interpreted as rotation matrices, i.e., elements of the special orthogonal group $SO(3)$, under the condition that $\det(\boldsymbol{U}) = \det(\boldsymbol{V}) = 1$. If this property is not satisfied, then either $\det(\boldsymbol{U}) = -1$ or $\det(\boldsymbol{V}) = -1$ as both matrices are still orthonormal, indicating reflection, not rotation. At last, Fig. 3.1 shows the effect of $\boldsymbol{\Lambda}$, scaling each coordinate by the factor $\sqrt{\lambda_i}$. [32, 45, 46, 47]

3.2.1 Static Compliance Matrix

Finally, the time has come for us to tie all of the previously established theoretical framework to the spatial static stiffness of six-axis serial robots. Consider a robotic system, e.g., the 6R anthropomorphic arm with an attached end-effector housing a tool. By applying external forces on the tool center point, we can measure its displacement from its equilibrium position, prescribed as the input of the inverse kinematics solution. As the dynamics governing such displacement is described by higher-order differential equations, the system needs to be linearized at this juncture to derive its state-space representation. Subsequently,

$$\begin{aligned}\dot{\boldsymbol{\xi}}(t) &= \boldsymbol{A}\boldsymbol{\xi}(t) + \boldsymbol{B}\boldsymbol{f}(t),\\ \boldsymbol{\delta}(t) &= \boldsymbol{C}\boldsymbol{\xi}(t) + \boldsymbol{D}\boldsymbol{f}(t),\end{aligned}$$

for some unit input force $\boldsymbol{f}(t) \in \mathbb{R}^3$ and output TCP displacement $\boldsymbol{\delta}(t) \in \mathbb{R}^3$. Since the subject of this thesis is *static* stiffness, $\dot{\boldsymbol{\xi}}(t) = 0$ and both the input force and output displacements are *not* functions of time. As a result, (3.2) becomes

$$\boldsymbol{G}_{\mathrm{static}} = \boldsymbol{D} - \boldsymbol{C}\boldsymbol{A}^{-1}\boldsymbol{B},$$

relating the displacement of the tool center point to the input unit force,

$$\boldsymbol{\Delta} = \boldsymbol{G}_{\mathrm{static}}\boldsymbol{F},$$

effectively encompassing static compliance of the system. Taking its SVD yields

$$\boldsymbol{\Delta} = \boldsymbol{U}\boldsymbol{\Lambda}_{\mathrm{static}}\boldsymbol{V}^T\boldsymbol{F}, \tag{3.3}$$

where $\boldsymbol{\Lambda}_{\mathrm{static}} \in \mathbb{R}^{3\times 3}$ is the *static compliance matrix*. Its diagonal entries are the principal static malleabilities $\sqrt{\lambda_1}$, $\sqrt{\lambda_2}$, and $\sqrt{\lambda_3}$ $[\mathrm{m\,N^{-1}}]$, i.e., the lengths of the three semi-axes of the *compliance ellipsoid* (Figure 3.1). The center of the

compliance ellipsoid is the point the tool center point is currently located at. Further, postmultiplying both sides of (3.3) by $\boldsymbol{U}^T$ yields

$$\boldsymbol{U}^T\boldsymbol{\Delta} = \boldsymbol{\Lambda}_{\text{static}}\boldsymbol{V}^T\boldsymbol{F},$$

i.e., the system inputs and outputs in the overarching coordinate system. This effectively gives us a directional field of three vectors at each chosen point of interest, where $\boldsymbol{u}_1/\|\boldsymbol{u}_1\|$ is the unit vector pointing in the direction of the largest malleability, i.e., $\sqrt{\lambda_1}$, and vice versa for the other malleabilities, $\sqrt{\lambda_2}$ and $\sqrt{\lambda_3}$.

3.2.2 Static Stiffness Matrix

Now we know the principal static compliances and the principal direction these compliances act in, obtaining static stiffness is trivial. Since stiffness is the reciprocal of compliance, in terms of matrices, the static stiffness matrix reads

$$\boldsymbol{K}_{\text{static}} = \boldsymbol{\Lambda}_{\text{static}}^{-1} = \begin{bmatrix} 1/\sqrt{\lambda_1} & 0 & 0 \\ 0 & 1/\sqrt{\lambda_2} & 0 \\ 0 & 0 & 1/\sqrt{\lambda_3} \end{bmatrix} \in \mathbb{R}^{3\times 3}$$

and has the principal stiffnesses,

$$\kappa_i = \frac{1}{\sqrt{\lambda_i}}, \quad i = 1, 2, 3,$$

as its diagonal entries. Given that $\sqrt{\lambda_1} \geq \sqrt{\lambda_2} \geq \sqrt{\lambda_3} > 0$, the principal static stiffness values are arranged in reverse order on the diagonal of $\boldsymbol{K}_{\text{static}}$, making κ_3 effectively the maximum principal stiffness, whilst κ_1 is the minimum principal stiffness since it corresponds to the maximum principal compliance.

Geometrically, one can imagine taking the inverse of the static malleability matrix as reshaping the compliance ellipsoid in Figure 3.1, i.e., each of the three semi-axes changing its length to κ_i, forming the *stiffness ellipsoid*.

3.2.3 Stiffness Homogeneity

Besides investigating the values of principal static stiffnesses, a question of homogeneity arises. Coming back to our geometric parallel, we ideally want the stiffness ellipsoid to become a *sphere*, i.e., have all of its three semi-axes be the *same* length. Mathematically, we can express such relationship as

$$\frac{\kappa_1}{\kappa_2} = \frac{\kappa_2}{\kappa_3} = \frac{\kappa_1}{\kappa_3} = 1$$

in terms of principal static stiffnesses, or as

$$\frac{\sqrt{\lambda_1}}{\sqrt{\lambda_2}} = \frac{\sqrt{\lambda_2}}{\sqrt{\lambda_3}} = \frac{\sqrt{\lambda_1}}{\sqrt{\lambda_3}} = 1,$$

in terms of principal static compliances.

Remark 3.4. Looking forward to Part II, the median value, κ_2, holds less significance. In practical terms of robotic machining, what truly matters are the principal minima and maxima. This effectively reshapes the stiffness ellipsoid in $\mathbb{R}^3$ into a stiffness ellipse in $\mathbb{R}^3$ (see Figure 3.2).

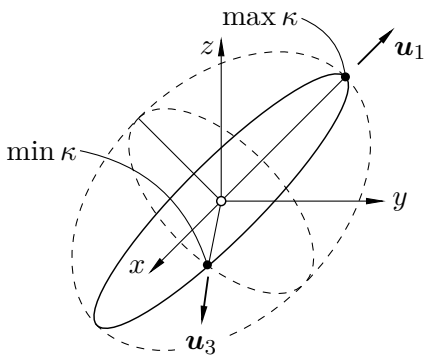


Figure 3.2 Stiffness ellipse in $\mathbb{R}^3$.

However, this ellipse isn't strictly confined to a plane because the columns of $\boldsymbol{U}^T$, representing the principal directions, can point in any direction in space as long as they maintain orthogonality, essentially forming a spatial curve. Under this framework, the scenario of stiffness homogeneity reads

$$\frac{\max \kappa}{\min \kappa} = \frac{\min \kappa}{\max \kappa} = 1,$$

where, for clarity, κ_3 and κ_1 are denoted as $\max \kappa$ and $\min \kappa$, respectively. $\diamond$

II
Simulation Model

4
Model Assembly

With the necessary theoretical framework now established, let us commence the second part of this thesis by discussing which robotic system is the most adequate for addressing the problem at hand. As elucidated in the preceding theoretical discourse, the 6R anthropomorphic arm emerges as the optimal candidate due to its capacity to streamline the inverse kinematics solution, facilitating a closed-form solution and obviating the necessity for numerical algorithms. Additionally, it is imperative that the chosen structure exhibits sufficient stiffness to endure the typical loads encountered during machining operations such as milling and drilling, thus constituting another pivotal criterion for our evaluation.

4.1 Choice of Robot

Numerous manufacturers currently provide robotic systems with the 6R anthropomorphic configuration. It is incumbent upon us to select the one that best aligns with the specified criteria. Among the arms considered by the author for inclusion in the model was the IRB 6660-100/3.3 from ABB, engineered specifically for high-performance, high-payload machining tasks. Another arm deliberated upon was the KR 120 R2700-2 from KUKA, which ultimately emerged as the chosen option and will be elaborated upon in the subsequent text.

4.1.1 KUKA KR 120 R2700-2

Revisiting our criteria, the KR 120 R2700-2 comfortably meets the stipulated requirements. For one, it is a 6R anthropomorphic arm with a spherical wrist and secondly, akin to the IRB 6660-100/3.3, is designed to excel in high-performance applications, allowing it to accommodate substantial force. Aligned with industry

norms, KUKA provides a freely accessible, concise one-page datasheet for each of its arms, featuring essential particulars like the work envelope and maximum reach. Conversely, detailed technical specifications and manuals are accessible solely through formal request. However, owing to KUKA's provision of meticulously detailed CAD models for its arms and accompanying accessories, the exhaustive specifications are deemed superfluous for our requirements.

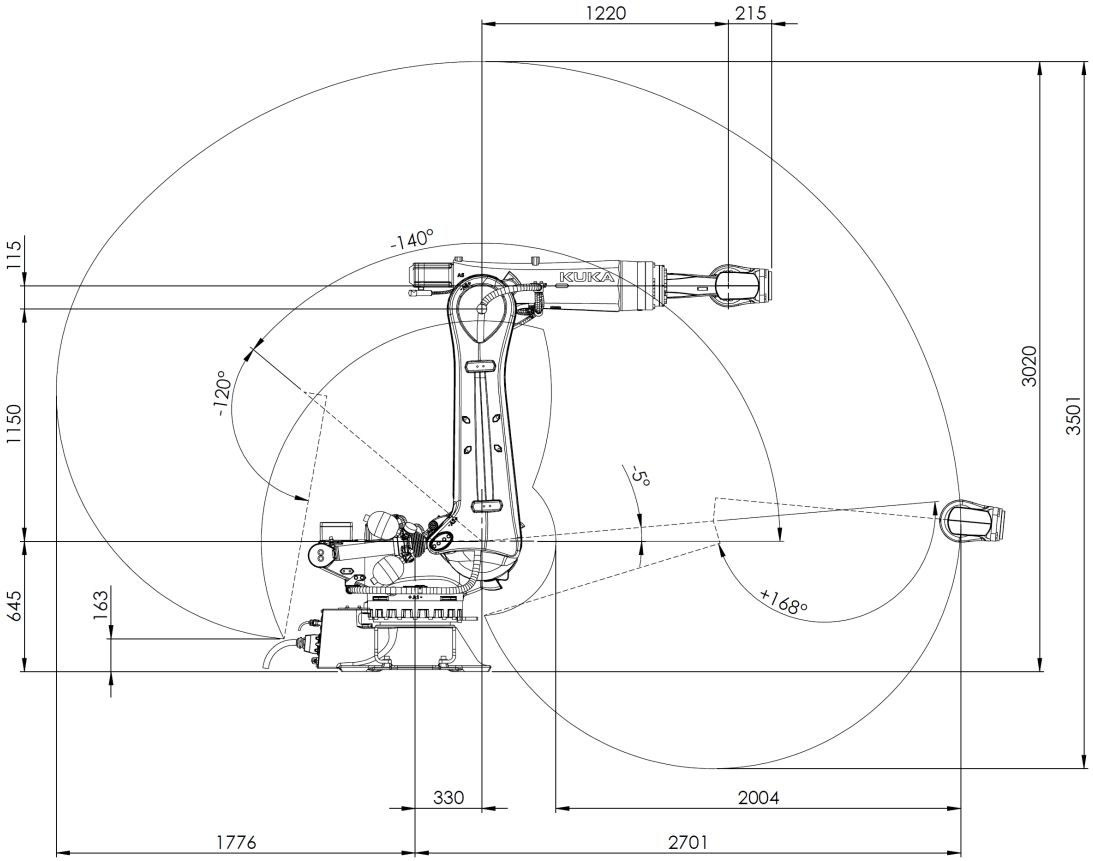


Figure 4.1 Main dimensions of the KR 120 R2700-2. Courtesy of KUKA.

Let us now proceed to detail all significant parameters of the KR 120 R2700-2, as specified in the datasheet provided by KUKA, cited under [49]. The main dimensions of the robot along with its workspace are depicted in Figure 4.1.

Parameter	Value
Robot weight [kg]	approx. 1 069
Maximum reach [mm]	2 701
Handling capacity [kg]	120

Continued on next page

Continued from previous page

Number of axes	6
Repeatability [mm]	±0.05
Installation surface	Floor
Installation surface area [mm^2]	754

Table 4.1 Technical parameters of the KR 120 R2700-2. Data from [49].

The main technical parameters of the KR 120 R2700-2 are shown within Table 4.1. Table 4.2 shows the range and angular velocity ω_i of each joint i.

(Joint)i	Range [°]		ω_i [° s^{-1}]
	min.	max.	
(J)1	−185	+185	120
(J)2	−140	−5	115
(J)3	−120	+168	120
(J)4	−350	+350	190
(J)5	−125	+125	180
(J)6	−350	+350	260

Table 4.2 KR 120 R2700-2 joint ranges and angular velocities. Data from [49].

Overall, the listed parameters suffice for all necessary computations to achieve the desired results. For additional information, please consult the KR 120 R2700-2 datasheet, or directly request the full product specification.

4.2 CAD Model within DS SolidWorks

SolidWorks, a CAD environment developed and maintained by Dassault Systèmes (DS), a leading French technology company specialising in CAD, CAM, and related software, stands out as an optimal choice for the CAD modelling phase. This is due to its capability to effortlessly import `.step` files, a format in which KUKA provides its models, ensuring a smooth and convenient workflow.

4.2.1 Link Assembly

The first step of the process is to obtain said `.step` files from the KUKA Download Center.[26] This can be done by searching "KR 120 R2700-2" in the search box of the Download Center and downloading a `.zip` archive of the desired format, i.e., `.step`. Next, upon extracting all files into a folder and opening SolidWorks,

[26] Found at `https://www.kuka.com/en-de/services/downloads?terms=Language:en:1&q=`.

one just simply opens the respective file by using `ctrl+O` and then clicking on it. SolidWorks then processes this input, parses all files, forms all surfaces and volumes, etc., and creates a new file with a `.sldprt` extension.

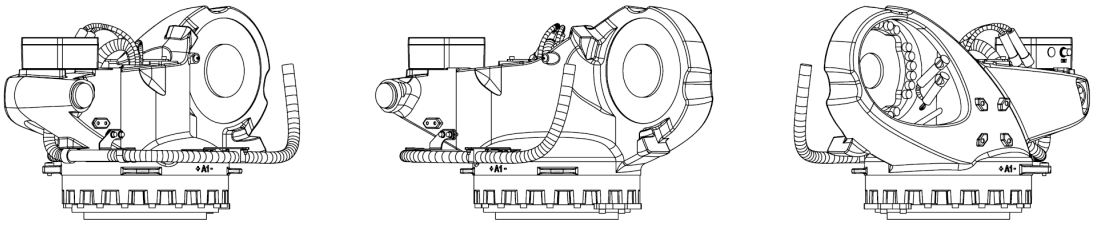


Figure 4.2 CAD model of link 2 (shoulder). Courtesy of KUKA.

Upon successfully opening and saving all constituent parts of the robot through this method, the subsequent modelling endeavour entails assembling these individual links into an assembly file with the `.sldasm` extension. This assembly process is facilitated by employing standard SolidWorks tools, such as mates. Notably, since KUKA furnishes the models with cylindrical leaders in locations where joints would conventionally be situated (refer to Figure 4.2), the mating process predominantly necessitates the application of the "coincident" and "concentric" mates. These mates then serve to precisely align the cylindrical leaders of each two respective links i and $i + 1$ at joint i.

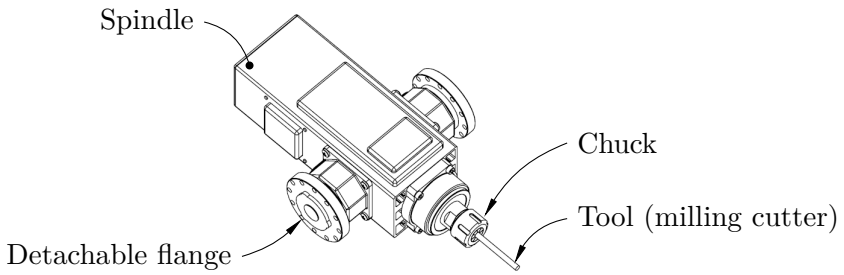


Figure 4.3 CAD model of the end-effector.

Additionally, the assembled links are complemented with a 500 millimetre booster frame, additional brackets, screws, pins, etc. KUKA directly provides the booster frame and brackets, while standardized elements such as screws (ISO 4762) are sourced from the SolidWorks Toolbox.

4.2.2 Attached End-Effector

Finally, let us turn our attention to the end-effector. Since we aim to develop a slight parallel to robotic machining, the chosen end-effector is a spindle with a milling cutter. As can be observed in Figure 4.3, the model of the spindle is very

thorough, featuring detachable flanges to mount the end-effector to the robot's flange (link 7). On the other hand, the tool is only modelled as a $\varnothing 10 \times 120$ [mm] cylinder in order to simplify the complicated geometry of milling cutters.

4.3 Simscape MBS Integration

Integrating the fully assembled model into Simscape MBS can be done using two different approaches. For one, the components can be imported to Simscape MBS individually by virtue of the "File Solid" block. All inertial parameters of the imported parts are then, by default, calculated by sourcing the density of the part from the `.sldprt` file. Further, the user has to manually define *all* coordinate systems, connect all blocks either through a revolute joint or simply as a rigid connection, etc., each step taking a considerable amount of time.

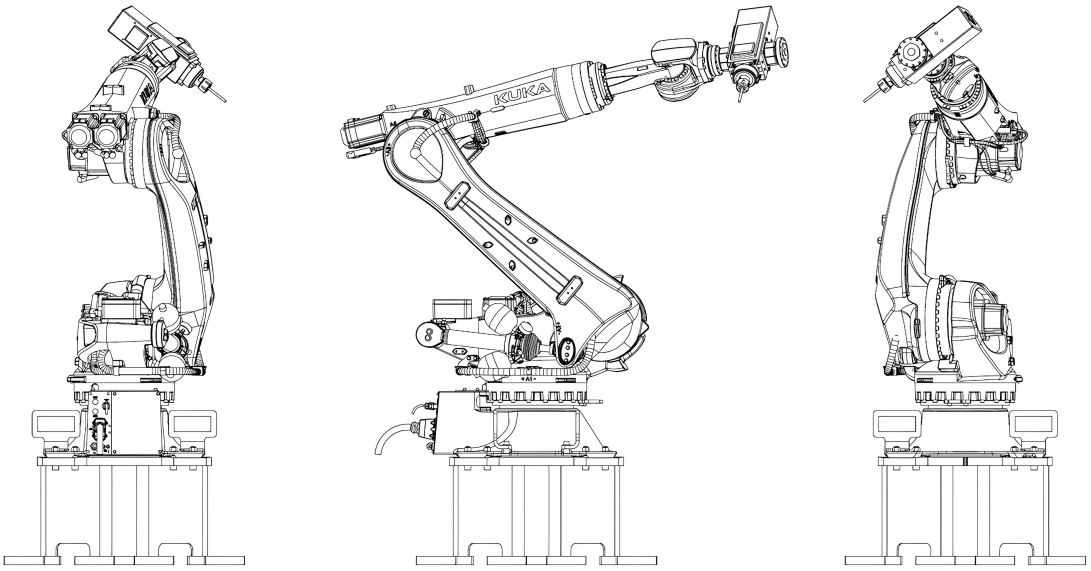


Figure 4.4 Final robot CAD assembly. Courtesy of KUKA.

It becomes apparent that streamlining this process becomes essential with larger models. To save time, there exists a handy plug-in for SolidWorks called Simscape Multibody Link. Although it significantly accelerates the block-by-block building process directly in Simulink, since it is an pre-written algorithm, the user has no control over how the assembled model behaves and looks.

4.3.1 Simscape Multibody Link for DS SolidWorks

Although the Simscape Multibody Link plug-in presents severe caveats as discussed in the previous paragraph, it is still a convenient tool, giving the user an

idea on how to wire the blocks in Simscape MBS, and also providing them with a data file containing all physical information about each component.

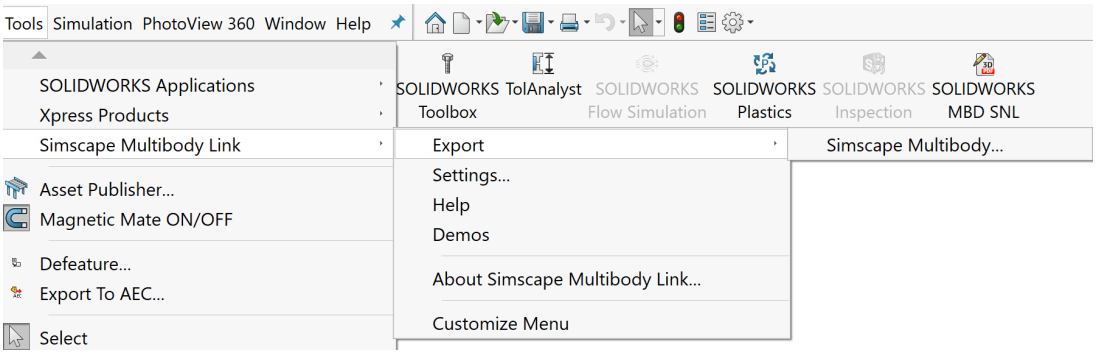


Figure 4.5 Simscape Mutlibody Link for DS SolidWorks.

The comprehensive guidelines detailing the installation and activation of the plug-in are available on the MathWorks Help Center[27] and affiliated platforms. In essence, following successful activation of the plug-in in both MATLAB and SolidWorks, the sole requisite within an *open* assembly is to run

```
Tools > Simscape Multibody Link > Export > Simscape Multibody
```

which outputs an `.xml` file with the same name as the assembly. Next, running

```
smimport('<filename>.xml');
```

in MATLAB's command window results in the instantiation of a new Simulink model derived from the SolidWorks assembly, accompanied by the creation of the `<filename>_DataFile.m` data file corresponding to the newly established model.

4.3.2 Block Rewiring and Model Reassembly

The Simulink/Simscape MBS model produced is generally suitable for smaller-scale implementations. However, in our current scenario, where we are working with a larger-scale model, the automatically generated model tends to exhibit significant clutter, with blocks densely interconnected. Moreover, the plug-in generates an excessive number of unnecessary coordinate systems, thereby exacerbating the clutter and undermining the overall clarity of the model.

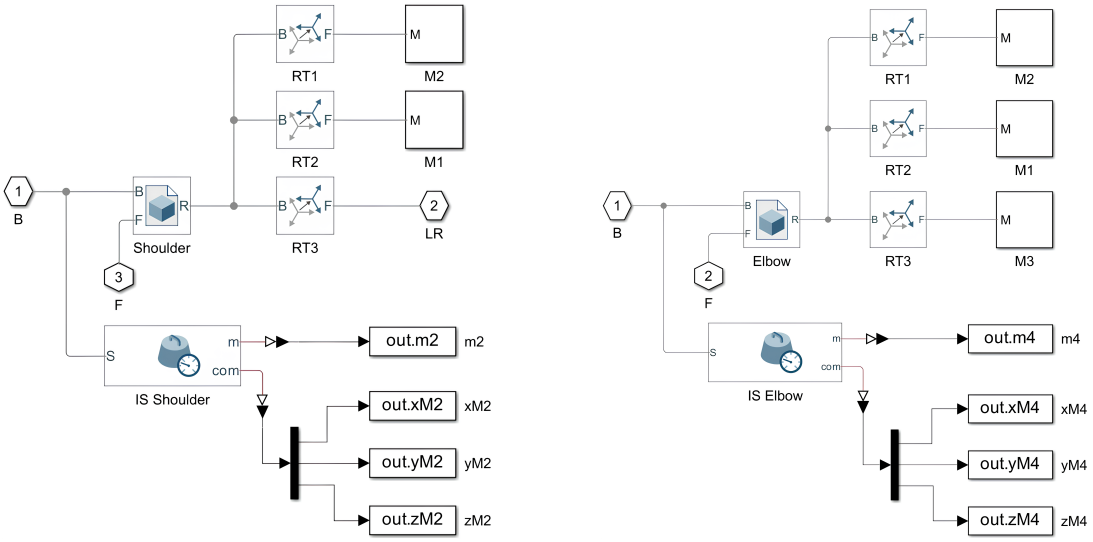
With the accompanying data file generated, we are now able to mostly disregard the existing model. Instead, we choose to initiate a fresh approach, establishing our own coordinate systems and optimizing the model. This targeted removal process is restrained, as certain "non-critical" blocks, such as motors

[27] Found at `https://www.mathworks.com/help/smlink/ref/linking-and-unlinking-simmechanics-link-software-with-solidworks.html`

and cables, are already well-defined. We specifically only eliminate some of the primary seven links, subsequently re-importing them back in and assigning their inertial properties. For instance, if the shoulder (Figure 4.2) is identified by index 28 in the data file, its inertial parameters are inputted in the form[28]

```
smiData.Solid(28).<parameter>
```

wherein `<parameter>` is, e.g., `mass` for mass m, `CoM` for the coordinates of the center of mass $\mathfrak{M}$, etc. It is important to note a significant limitation with all inertial parameters except mass — each of these properties is referenced to a predetermined coordinate system `R` (as seen in Fig. 4.6), predefined in the CAD model. As the CAD model is provided by KUKA, users are unable to relocate this system. Attaching the "Inertia Sensor" block to each rigidly connected body group and measuring the center of mass coordinates in a specific coordinate system can resolve this issue. This approach is demonstrated in Figure 4.6, where the outputs of the "Inertia Sensor" block (IS) are sent back as variables to the MATLAB workspace, such as `xM2`, using the "To Workspace" block.[29]



(a) Link 2 (shoulder) (b) Link 4 (elbow)

Figure 4.6 Link subsystems.

In addition, beyond the task of rewiring all blocks, it is expedient to construct subsystems for each assembly of rigidly interconnected bodies within the links, facilitating further improvement in the model's clarity. These subsystems are depicted in Figure 4.6, while in the comprehensive model (Figure 4.7), users do not initially observe these expanded structures.

[28]These parameters are entered-in in the left panel of the File Solid block (see Chap. 2).
[29]In Fig. 4.6, the `m` port outputs mass and the `com` port outputs center of mass coordinates.

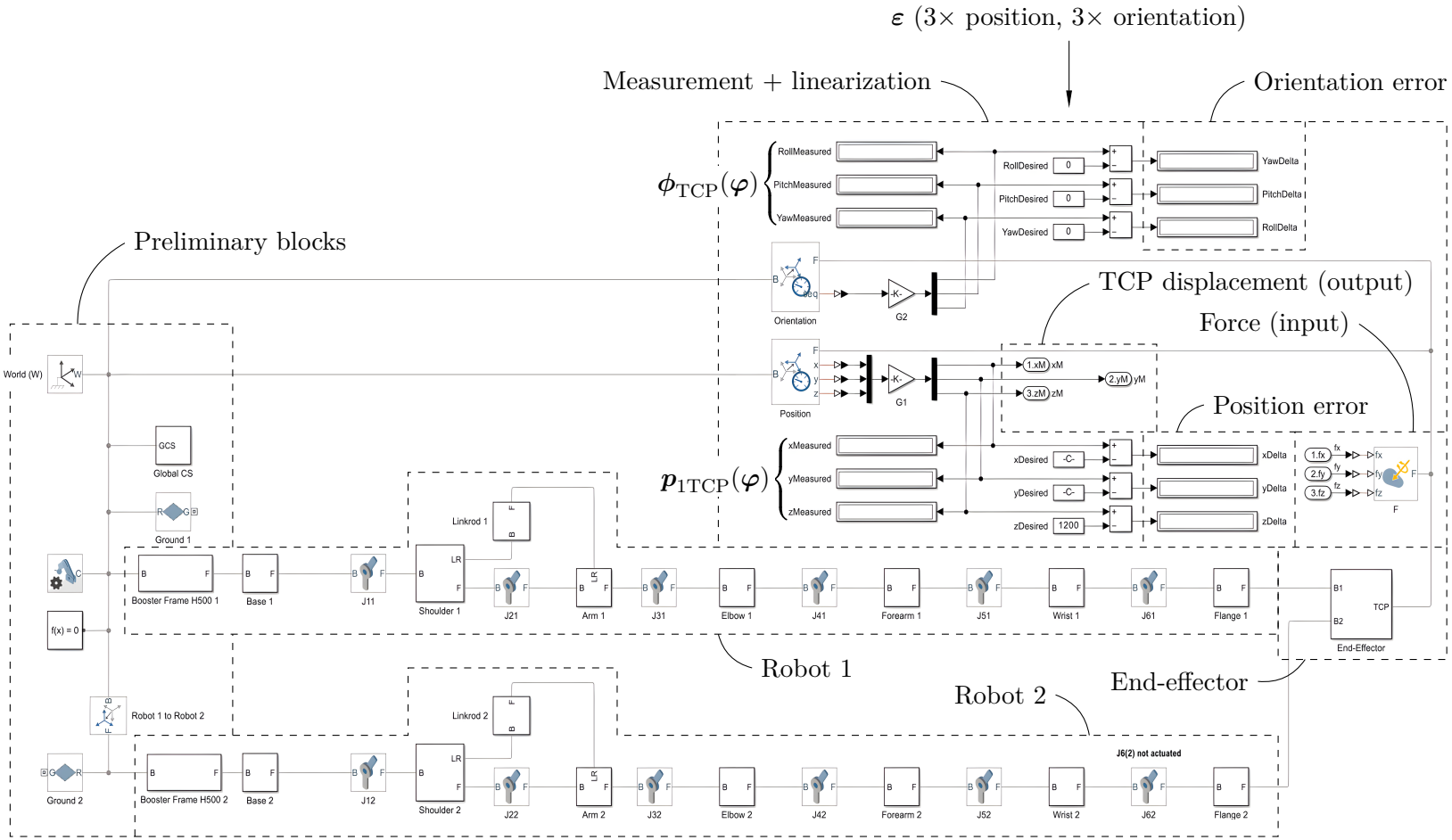


Figure 4.7 Complete Simulink/Simscape MBS model.

4.3.3 Model Walkthrough

The comprehensive layout of the KR 120 R2700-2 Simulink/Simscape MBS model is depicted in Figure 4.7 and can be segmented both horizontally and vertically. Let us describe its contents in both of these directions in the following text.

- **Horizontal Direction.** Progressing from left to right, the model commences with essential preliminary components, such as the "Mechanism Configuration" and "Solver Configuration" blocks. Moving forward, users encounter top-level subsystems that, when opened, unveil structures akin to those depicted in Figure 4.6. These subsystems are organized sequentially from the booster frame and base to the end-effector positioned on the right. The six revolute joints of the 6R anthropomorphic arm are situated between the respective links they connect, with their configurations to be elaborated upon in the subsequent chapter.
- **Vertical Direction.** The middle one of the three vertical layers belongs to the first of the two total arms. For simulation with both arms coupled, the bottom row houses the second robot and can be simply commented-out by selecting it and using `ctrl+shift+X`, effectively forming a new model with only one arm present. The topmost layer serves as a measuring and linearization station. The measuring portion is realised using two "Transform Sensor" blocks, the bottom of which measures the position, and the top one measures the orientation of the tool center point, marked as the `TCP` port on the right side of the "End-Effector" subsystem. Next, there are a total of twelve displays, displaying either the current pose of the tool center point (left column of six displays), or the error from the desired pose (right column of six displays). The measured values are fed to the first six displays through two "Gain" blocks, simply allowing for unit conversion, either from [m] to [mm] as for position, or from [rad] to [°] as for orientation. The desired positions and orientations of the TCP are entered in as constants, updating on each run of the inverse kinematics algorithm (see Chapter 5), and are subsequently being subtracted from the `<parameter>Measured` values to display the current error, as hinted by the `<parameter>Delta` labels of the right six displays. Lastly, the rightmost block of the top layer is the "External Force and Torque" block, applying force on the tool center point via three "Inport" blocks, `fx`, `fy`, and `fz`, which serve an the input of our nonlinear system for future linearization. The output of this nonlinear system are the displacements of the TCP, `xM`, `yM`, and `zM`, fed from the position "Transform Sensor" block.

5
Algorithmization

Now that the model within Simulink/Simscape MBS has been assembled, we arrive at the essence of this thesis — the algorithmization of the model, comprising the systematic computation of inverse kinematics, forward kinematics, and related procedures, culminating in the determination of spatial static stiffness, which stands as the primary focus of this research endeavor.

Notation 5.1. In this chapter, we will seamlessly alternate between the established standard notation for position vector components and its simplified form, represented by x, y, and z, in order to maintain clarity whenever necessary. $\diamond$

5.1 Chapter Organisation

Since the spatial stiffness function, $\kappa(\boldsymbol{\varepsilon}) : \mathbb{R}^6 \mapsto (0, +\infty)$, is a six variable one, dependent on the current position and orientation of the TCP, showing results graphically is impossible. The idea of the stiffness map, i.e., the output of the algorithm in Figure 5.1, is to *fix* all orientation angles, α, β, γ, and one position coordinate, e.g., z of the tool center point, letting it work only in the respective xy plane, creating a coloured contour plot of the stiffness

$$\kappa(x, y, z_k) : \mathbb{R}^3 \mapsto (0, +\infty), \quad \forall\ k \in \{0, 1, \dots\},$$

within that plane. Next, the plane is shifted one level above to

$$z_{k+1} = z_k + \Delta z, \quad \forall\ \Delta z > 0,$$

and the same process repeats, ultimately resulting in multiple coloured contour plots situated next to each other. After the completion of these xy-maps, one

fixes a different coordinate, e.g., x, and the process repeats, similarly for the y-coordinate, i.e., xz-planes. This ultimately results in stiffness maps across all three Cartesian planes, xy, yz, and xz, in various levels, giving the reader an idea on how the stiffness changes across three-dimensional space $\mathbb{E}^3$.

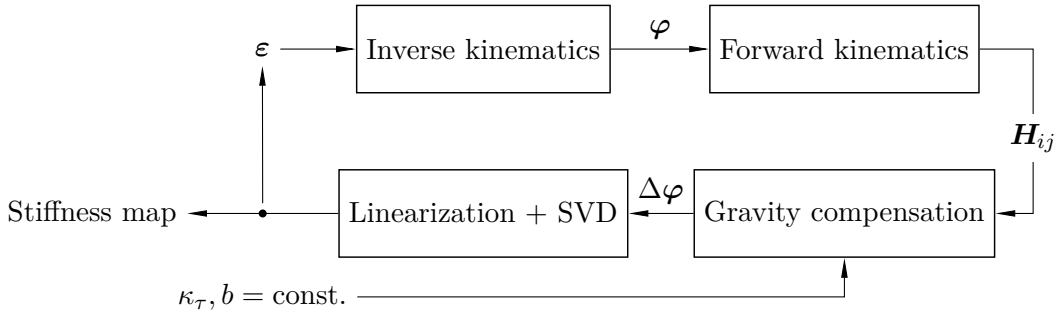


Figure 5.1 Organisation structure of Chapter 5.

To obtain accurate results, we need to divide each plane of interest into segments in both directions, i.e., create an equidistant grid of points within that plane. If we are provided with the dimensions of said plane in both coordinates, we thereby know the planar coordinates of each one of its points, with the third, spatial coordinate of these points being currently constant. If we broaden our view and consider all possible levels of this moving plane, the spatial coordinates of each point the static stiffness is being evaluated at can be stored in a 3rd order *tensor*, with the length and width of the tensor corresponding to the grid in the current plane, more specifically the planar coordinates of each of its points, and the depth of the tensor encompassing all possible planes (see Figure 5.2).

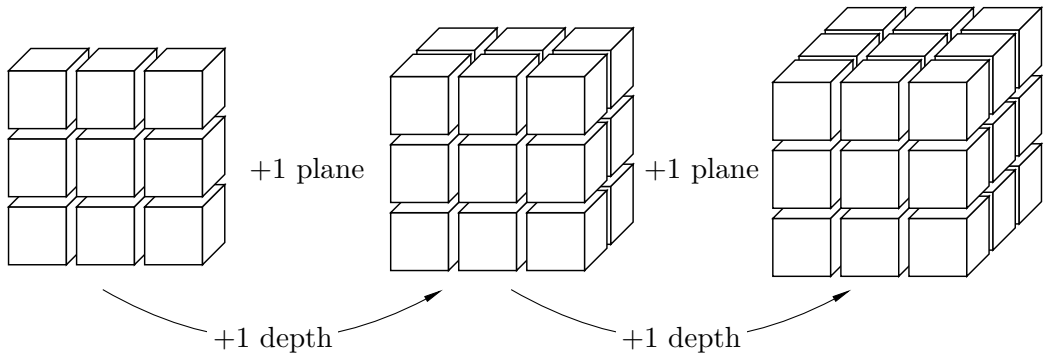


Figure 5.2 Tensor of stiffness-evaluation points.

It is because of the need to precisely reach these points we need to start with the inverse kinematics solution and proceed as showcased in Figure 5.1. This algorithm follows a loop where upon each iteration, a new point is inputted as the desired TCP position until the static stiffness has been evaluated at each point.

5.2 Closed-Form Inverse Kinematics

Being a 6R anthropomorphic arm with a spherical wrist, the KR 120 R2700-2 allows for closed-form inverse kinematics solution. Before solving for $\boldsymbol{\varphi} \in \mathbb{R}^6$, we need to topologically simplify the real robotic system (Figure 4.4), i.e., create its kinematic diagram, in order to streamline the computation process.

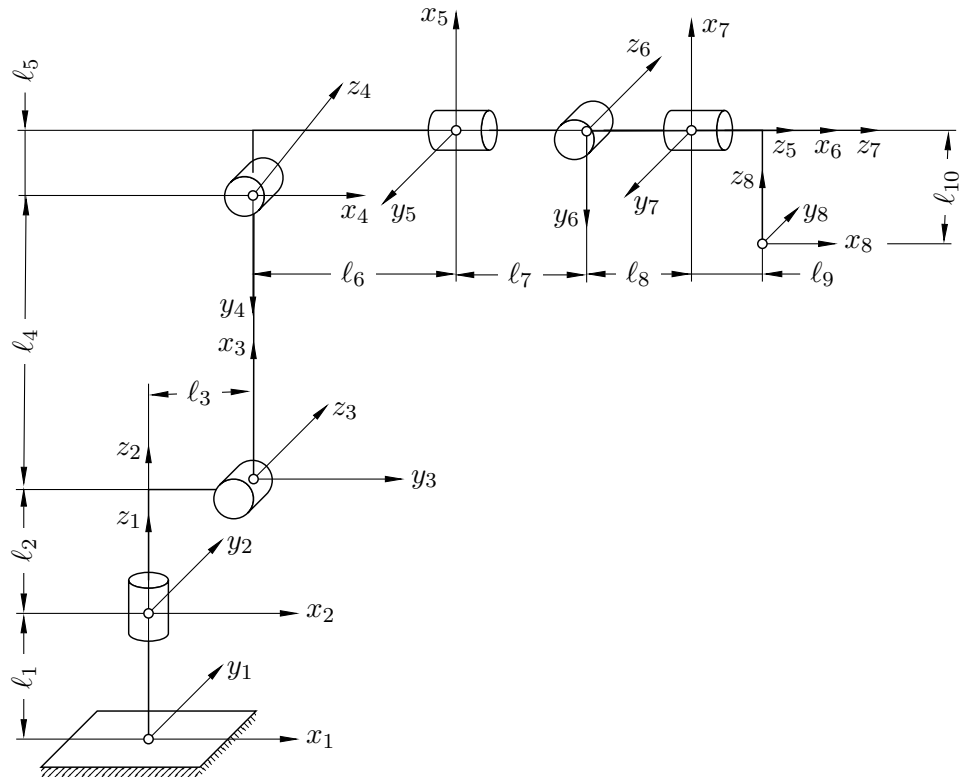


Figure 5.3 Kinematic diagram of the KR 120 R2700-2. Not to scale.

Such kinematic diagram is depicted in Fig. 5.3. The coordinate systems designated on the robot are congruent with those delineated within the Simscape MBS model. The subsequent table introduces the distances between these systems.

i	1	2	3	4	5	6	7	8	9	10
ℓ_i [mm]	735.6	409.4	330	1 150	115	868.5	351.5	170	213.5	345

Table 5.1 Coordinate system displacements.

Furthermore, the right hand rule for positive rotation [Figure 1.5(b)] can be used to determine the positive direction of angular displacement for each joint. Here, the right-hand thumb points along the joint axis, i.e., the z-axis.

5.2.1 Inverse Position Kinematics

The inverse position kinematics problem for the 6R anthropomorphic arm starts with determining the wrist center point position vector $\boldsymbol{p}_{1\mathrm{WCP}}(\boldsymbol{\varepsilon}) \in \mathbb{R}^3$. In light of (1.17) and Figure 5.3, we can write

$$\boldsymbol{p}_{1\mathrm{WCP}}(\boldsymbol{\varepsilon}) = \boldsymbol{p}_{1\mathrm{TCP}} + \boldsymbol{R}_{18}(\boldsymbol{\phi}_{\mathrm{TCP}})\Big[\ell_{10}\boldsymbol{k}_8 - (\ell_8 + \ell_9)\boldsymbol{i}_8\Big],$$

since there is no displacement in the direction of the y_8-axis. Now, φ_{12} reads

$$\varphi_{12} = \mathrm{atan2}(p_{1\mathrm{WCP}_y}, p_{1\mathrm{WCP}_x}), \tag{5.1}$$

as per (1.18). Moreover, the homogeneous transformation

$$\boldsymbol{H}_{12} = \begin{bmatrix} \cos\varphi_{12} & -\sin\varphi_{12} & 0 & 0 \\ \sin\varphi_{12} & \cos\varphi_{12} & 0 & 0 \\ 0 & 0 & 0 & \ell_1 \\ 0 & 0 & 0 & 1 \end{bmatrix} \in SE(3)$$

is known, and given

$$\begin{bmatrix} \boldsymbol{p}_{1\mathrm{WCP}} & 1 \end{bmatrix}^T = \boldsymbol{H}_{12}\begin{bmatrix} \boldsymbol{p}_{2\mathrm{WCP}} & 1 \end{bmatrix}^T,$$

we have

$$\begin{bmatrix} \boldsymbol{p}_{2\mathrm{WCP}} & 1 \end{bmatrix}^T = \boldsymbol{H}_{12}^{-1}\begin{bmatrix} \boldsymbol{p}_{1\mathrm{WCP}} & 1 \end{bmatrix}^T,$$

yielding the coordinates of the wrist center point in $O_2x_2y_2z_2$.

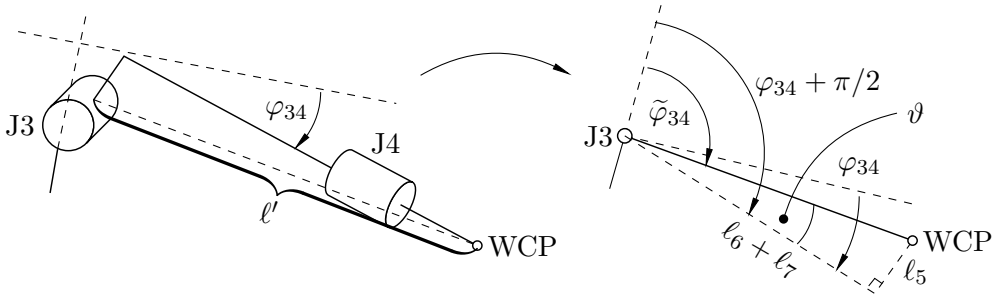


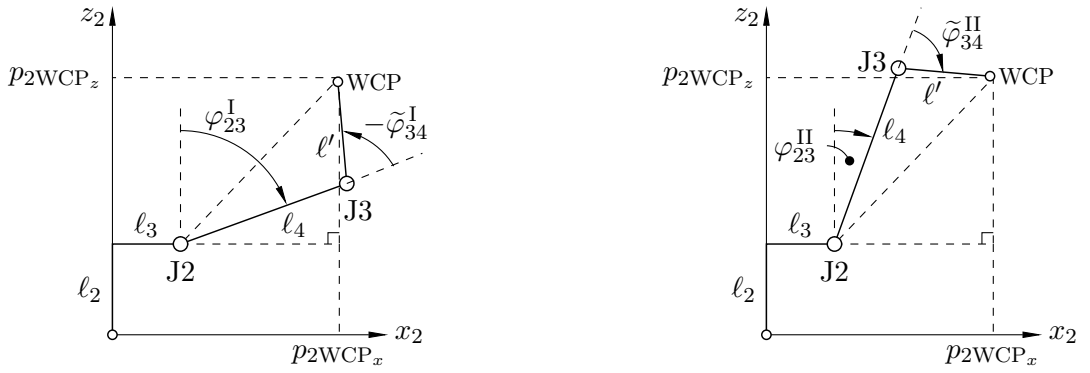
Figure 5.4 Substitution of links 4 and 5.

The determination of solutions for φ_{23} and φ_{34} necessitates their disentanglement into distinct scenarios: the elbow-up configuration problem and the elbow-down configuration problem. Ultimately, it will be revealed that both configurations converge upon similar solutions. Initially, given that joint 4 does not exert influence on the positional kinematics of the TCP, it can be temporarily

disregarded. Subsequently, we opt to substitute links 4 and 5 with a single straight virtual link of length ℓ', prioritizing clarity. It follows from Figure 5.4 that

$$\ell' = \sqrt{(\ell_6 + \ell_7)^2 + \ell_5^2},$$
$$\vartheta = \operatorname{atan2}(\ell_5, \ell_6 + \ell_7),$$

seeing as the dashed triangle is a right triangle.



(a) Elbow-up (I) (b) Elbow-down (II)

Figure 5.5 Configurations of the resulting 2R planar sub-mechanism.

What we are now left with is a 2R planar sub-mechanism of the 6R anthropomorphic arm (Figure 5.5), already discussed in the theoretical portion of this thesis (see Figure 1.12). Using the cosine theorem,

$$\cos \widetilde{\varphi}_{34}^{w} = -\frac{\ell_4^2 + \ell'^2 - [(p_{2\mathrm{WCP}_x} - \ell_3)^2 + (p_{2\mathrm{WCP}_z} - \ell_2)^2]}{2\ell_4\ell'},$$

wherein $w = \mathrm{I}, \mathrm{II}$. Further, from basic trigonometry, we obtain

$$\sin \widetilde{\varphi}_{34}^{w} = \pm\sqrt{1 - \cos^2 \widetilde{\varphi}_{34}^{w}},$$

and therefore, accounting for Figure 5.4, the solution for φ_{34}^{w} reads

$$\varphi_{34}^{w} = \operatorname{atan2}\left(\pm\sqrt{1 - \cos^2 \widetilde{\varphi}_{34}^{w}}, \cos \widetilde{\varphi}_{34}^{w}\right) + \vartheta - \pi/2. \tag{5.2}$$

As for φ_{23}^{w}, expressing the wrist center point coordinates as projections of the links lengths yields a system of nonlinear equations of the form

$$p_{2\mathrm{WCP}_x} = \ell_3 + \ell_4 \sin \varphi_{23}^{w} + \ell'(\sin \varphi_{23}^{w} \cos \widetilde{\varphi}_{34}^{w} + \cos \varphi_{23}^{w} \sin \widetilde{\varphi}_{34}^{w})$$
$$p_{2\mathrm{WCP}_z} = \ell_2 + \ell_4 \cos \varphi_{23}^{w} + \ell'(\cos \varphi_{23}^{w} \cos \widetilde{\varphi}_{34}^{w} - \sin \varphi_{23}^{w} \sin \widetilde{\varphi}_{34}^{w}),$$

but since φ_{34}^w is already known, we can solve for $\sin\varphi_{23}^w$ and $\cos\varphi_{23}^w$. Rearranging the system of equations into a linear system of the form $\boldsymbol{\Phi}_{34}^w\boldsymbol{\varphi}_{23}^w = \boldsymbol{\ell}$ results in

$$\underbrace{\begin{bmatrix}\ell_4 + \ell'\cos\widetilde{\varphi}_{34}^w & \ell'\sin\widetilde{\varphi}_{34}^w \\ -\ell'\sin\widetilde{\varphi}_{34}^w & \ell_4 + \ell'\cos\widetilde{\varphi}_{34}^w\end{bmatrix}}_{\boldsymbol{\Phi}_{34}^w}\underbrace{\begin{bmatrix}\sin\varphi_{23}^w \\ \cos\varphi_{23}^w\end{bmatrix}}_{\boldsymbol{\varphi}_{23}^w} = \underbrace{\begin{bmatrix}p_{2\mathrm{WCP}_x} - \ell_3 \\ p_{2\mathrm{WCP}_z} - \ell_2\end{bmatrix}}_{\boldsymbol{\ell}},$$

and upon computing

$$\boldsymbol{\varphi}_{23}^w = (\boldsymbol{\Phi}_{34}^w)^{-1}\boldsymbol{\ell},$$

the expression for φ_{23}^w reads

$$\varphi_{23}^w = \operatorname{atan2}(\sin\varphi_{23}^w, \cos\varphi_{23}^w), \tag{5.3}$$

finalising the inverse position kinematics problem.

5.2.2 Inverse Orientation Kinematics

The rotation matrix of the tool center point can be expressed as

$$\boldsymbol{R}_{18}(\boldsymbol{\phi}_{\mathrm{TCP}}) = \boldsymbol{R}_{14}(\varphi_{12}, \varphi_{23}^w, \varphi_{34}^w)\boldsymbol{R}_{48},$$

where $\boldsymbol{R}_{18}(\boldsymbol{\phi}_{\mathrm{TCP}}) \in SO(3)$ is given by the desired orientation of the TCP and

$$\begin{aligned}\boldsymbol{R}_{14}(\varphi_{12}, \varphi_{23}^w, \varphi_{34}^w) = {}& \boldsymbol{R}_{12}(z_1, \varphi_{12})\boldsymbol{R}_{23}(y_2, -\pi/2)\boldsymbol{R}_{23}(x_2, -\pi/2) \\ & \times \boldsymbol{R}_{23}(z_2, \varphi_{23}^w)\boldsymbol{R}_{34}(z_3, \pi/2)\boldsymbol{R}_{34}(z_3, \varphi_{34}^w),\end{aligned}$$

as per Figure 5.3 when not in its home configuration. By (1.19), we have

$$\boldsymbol{R}_{48}(\varphi_{12}, \varphi_{23}^w, \varphi_{34}^w, \boldsymbol{\phi}_{\mathrm{TCP}}) = \boldsymbol{R}_{14}^T(\varphi_{12}, \varphi_{23}^w, \varphi_{34}^w)\boldsymbol{R}_{18}(\boldsymbol{\phi}_{\mathrm{TCP}}), \tag{$\star$}$$

or alternatively, from (1.20),

$$\boldsymbol{R}'_{48}(\varphi_{45}, \varphi_{56}, \varphi_{67}) = \boldsymbol{R}_{45}(\varphi_{45})\boldsymbol{R}_{56}(\varphi_{56})\boldsymbol{R}_{67}(\varphi_{67})\boldsymbol{R}_{78},$$

wherein

$$\begin{aligned}\boldsymbol{R}_{45}(\varphi_{45}) &= \boldsymbol{R}_{45}(x_4, -\pi/2)\boldsymbol{R}_{45}(y_4, \pi/2)\boldsymbol{R}_{45}(z_4, \varphi_{45}), \\ \boldsymbol{R}_{56}(\varphi_{56}) &= \boldsymbol{R}_{56}(y_5, -\pi/2)\boldsymbol{R}_{56}(x_5, \pi/2)\boldsymbol{R}_{56}(z_5, \varphi_{56}), \\ \boldsymbol{R}_{67}(\varphi_{67}) &= \boldsymbol{R}_{67}(x_6, -\pi/2)\boldsymbol{R}_{67}(y_6, \pi/2)\boldsymbol{R}_{67}(z_6, \varphi_{67}), \\ \boldsymbol{R}_{78} &= \boldsymbol{R}_{78}(x_7, \pi)\boldsymbol{R}_{78}(y_7, \pi/2),\end{aligned}$$

and if we label the (i, j)-th element of $\boldsymbol{R}_{45}(\varphi_{45})$, $\boldsymbol{R}_{56}(\varphi_{56})$, and $\boldsymbol{R}_{67}(\varphi_{67})$ a_{ij}, b_{ij}, and c_{ij}, respectively, we get a conglomerate rotation matrix of the form

$$\boldsymbol{R}'_{48}(\varphi_{45}, \varphi_{56}, \varphi_{67}) = \begin{bmatrix} b_{31} & -b_{32}c_{22} & b_{32}c_{21} \\ a_{21}b_{11} & a_{22}c_{32} - a_{21}b_{12}c_{22} & a_{21}b_{12}c_{21} - a_{22}c_{31} \\ a_{31}b_{11} & a_{32}c_{32} - a_{31}b_{12}c_{22} & a_{31}b_{12}c_{21} - a_{32}c_{31}\end{bmatrix} \in SO(3),$$

ultimately yielding

$$\varphi_{45} = \operatorname{atan2}\left[\frac{r_{31}}{\sin(\arccos r_{11})}, \frac{r_{21}}{\sin(\arccos r_{11})}\right], \tag{5.4}$$

$$\varphi_{56} = \arccos r_{11}, \tag{5.5}$$

$$\varphi_{67} = \operatorname{atan2}\left[\frac{r_{12}}{\sin(\arccos r_{11})}, \frac{r_{13}}{\sin(\arccos r_{11})}\right], \tag{5.6}$$

when juxtaposed with $(\star)$, denoting the (i, j)-th element of $(\star)$ as r_{ij}.

5.2.3 Implementation

Implementing the inverse kinematics solution into Simscape MBS first requires writing a MATLAB function of the form

```
function [phi] = ik6Rarm1(x,y,z,roll,pich,yaw)
        <inverse kinematics solution>
```

and running it to obtain $\boldsymbol{\varphi} \in \mathbb{R}^6$, comprised of solutions (5.1 — 5.6). With the vector of joint coordinates loaded in the MATLAB workspace, interfacing this solution with Simscape MBS is as simple as typing `phi(`i`)` into the "State Targets" option in the block configuration dialogue box of the respective joint i.

5.3 Forward Kinematics

Upon the completion of the inverse kinematics solution, the next step in the algorithmization of the KR 120 R2700-2 is the forward kinematics solution, which outputs *all* transformation matrices $\boldsymbol{H}_{ij} \in SE(3)$, later to be inputted into the gravity compensation algorithm (see Figure 5.1 for reference).

Since the computation of the respective homogeneous transformations is trivial and its showcase would only prolong the thesis, the author kindly asks the reader to refer to (1.12, 1.14) and their surrounding discussion.

5.3.1 Implementation

Implementing the forward kinematics solution into Simscape MBS is not necessary as it has no direct influence on the behaviour of the model. Instead, we only need to write a MATLAB function of the form

```
function [H12,H23,H34,H45,H56,H67,H78] = fk6Rarm1(phi)
        <forward kinematics solution>
```

and run it to save all transformations into the MATLAB workspace.

5.4 Static Gravity Compensation

The demonstration of the static gravity compensation algorithm, i.e., the solution for $\Delta\boldsymbol{\varphi} \in \mathbb{R}^6$, shall be somewhat of a middle ground between the forward and inverse kinematics solutions, i.e., its extent will be neither too long nor brief. Firstly, we have already showcased the "Inertia Sensor" block, thanks to which we are able to measure the coordinates of the center of mass $\mathfrak{M}$ in a chosen coordinate system. The setup for such measurements can be seen in Figure 4.6 and its implementation back to MATLAB workspace comprises of running

```
out = sim('KR_120_R2700_2.slx');
```

in the MATLAB command window. Since the variables fed back to MATLAB workspace, e.g., `xM2` [Fig. 4.6(a)], are being measured at each discrete point in time the simulation is running, these variables are loaded as arrays and not as scalars. Yet, given their constant character (the center of mass is measured within a system tightly bound to the given link), we can confidently extract *any* value from this array, knowing it will be correct. For instance, to extract the very last element of a variable array, one runs the command

```
out.<variable>(end,end)
```

in the MATALAB command window, where `<variable>` is, e.g., `xM2`, `m2`, etc. Upon viewing the value of this variable, storing it is as simple as copying and pasting it somewhere within a new MATLAB function or script.

As for the actual computation of $\Delta\boldsymbol{\varphi} \in \mathbb{R}^6$, we further need all rotation matrices and position vectors of the origins of each coordinate system as demonstrated in Chapter 1. We ultimately have both at our disposal as all homogeneous transformations $\boldsymbol{H}_{ij} \in SE(3)$ are now stored within the MATLAB workspace since the `fk6Rarm1.m` function has already been executed. Finally, the last two inputs of the static gravity compensation algorithm are the torsion spring stiffness κ_τ and the damping coefficient b. We elect to set

```
kappa_tau = 2.5e4; % [N.m/rad]
b         = kappa_tau/100; % [N.m/(rad/s)]
```

as the KR 120 R2700-2 is considerably large and thereby heavy.[30] Now, the stage is set and one is able to compute the vector of joint compensations. As the specifics of the analytical solution of $\Delta\boldsymbol{\varphi} \in \mathbb{R}^6$ for the 6R anthropomorphic arm have already been discussed, and the solution is simply iterative, the author refers the reader to Section 1.4 of Chapter 1. What follows is, again, an outline on the implementation of this solution into MATLAB and subsequently Simscape MBS, allowing for subsequent linearization and singular value decomposition.

[30] The heaviest link of the KR 120 R2700-2 is link 3 (arm) with a mass of approx. 304 [kg].

5.4.1 Implementation

The last user-defined function reads

```
function [dphi] =
↪  comp6Rarm1(H12,H23,H34,H45,H56,H67,H78,kappa_tau,b)
        <gravity compensation solution>
```

which, upon running, saves the vector of joint compensations into MATLAB workspace. To implement such solution into the model within Simscape MBS, one now has to change more parameters within the block configuration dialogue box of the "Revolute Joint" block. Under the "Internal Mechanics" option, the "Equilibrium Position" of the torsion spring attached to joint i is to be set to `phi(`i`) + dphi(`i`)`, with the "Spring Stiffness" parameter being `kappa_tau` and "Damping Coefficient" being `b`. At last, under "Actuation", the "Torque" is set to "None" and "Motion" to "Automatically Computed". These settings, combined with the already set parameters from the inverse kinematics solution, ensure the error from the desired TCP pose is no more than approximately $1 \cdot 10^{-11}$ [mm] as for position, and approximately $1 \cdot 10^{-12}$ [°] as for orientation.

5.5 Robot Coupling

It was already established that the Simscape MBS model of the final robotic system houses both manipulators at once and commenting out the bottom rigid body tree results in an effectively new model with only one arm present.

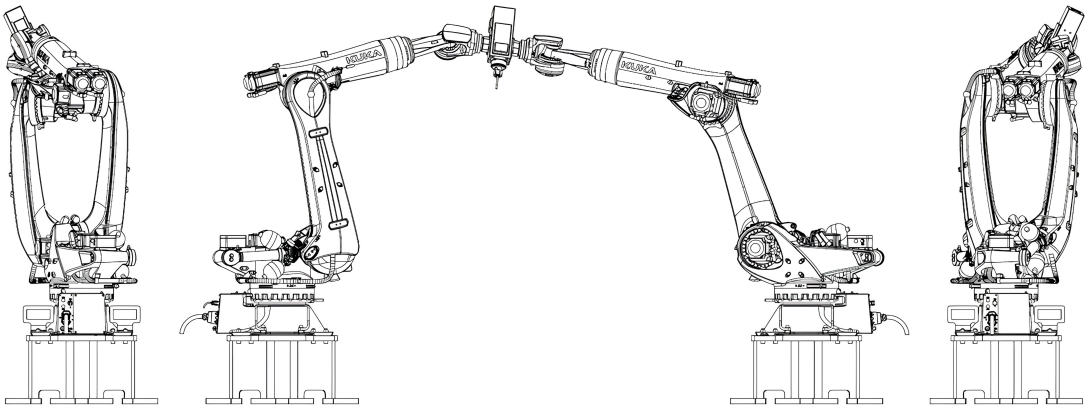


Figure 5.6 Two coupled robots working together. Courtesy of KUKA.

Given both robots are essentially identical kinematic structures, the algorithms developed for a solitary manipulator apply for the case of coupling two manipulators as well. Yet, due to the mirroring of the second manipulator, some things have to be altered in order for the algorithms to work correctly.

5.5.1 Closed-Form Inverse Kinematics

The inverse kinematics solution for the second robot is altered in two way. For one, since the second robot is mirrored,

$$\alpha = \alpha', \quad \beta = -\beta', \quad \gamma = -\gamma',$$

in the local coordinate system of the second robot, $O'_1x'_1y'_1z'_1$, as $\langle \boldsymbol{i}'_1, \boldsymbol{i}_1 \rangle = -1$, $\langle \boldsymbol{j}'_1, \boldsymbol{j}_1 \rangle = -1$, and $\langle \boldsymbol{k}'_1, \boldsymbol{k}_1 \rangle = 1$, i.e., the positive direction of the x- and y- axes are reversed, which can also be observed if one mirrors Fig. 5.3. Secondly, as the closed-form solution requires the position vector of the wrist center point, we need to alter this position vector so it originates from O'_1. The respective homogeneous transformation from $O_1x_1y_1z_1$ to $O'_1x'_1y'_1z'_1$ is of the form

$$\boldsymbol{H}_{11'} = \begin{bmatrix} -1 & 0 & 0 & D_{11'} \\ 0 & -1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \in SE(3),$$

where $D_{11'} = 3\,867$ [mm] is the distance between O_1 and O'_1, calculated so that the two robots meet precisely at the end-effector when both in their home configuration. By virtue of this transformation, we have

$$\begin{bmatrix} \boldsymbol{p}'_{1\mathrm{TCP}} & 1 \end{bmatrix}^T = \boldsymbol{H}_{11'}^{-1} \begin{bmatrix} \boldsymbol{p}_{1\mathrm{TCP}} & 1 \end{bmatrix}^T,$$

and the following solution is the same as for a solitary manipulator.

As for implementation into MATLAB and Simscape MBS, one writes as very similar function to the `ik6Rarm1.m` function. It is defined as

```
function [phidash] = ik6Rarm2(x,y,z,roll,pitch,yaw)
         <altered inverse kinematics solution>
```

and upon running such function, the vector of joint coordinates for the second arm, $\boldsymbol{\varphi}' \in \mathbb{R}^6$, is saved to MATLAB's workspace. Within the model (Figure 4.7), the settings for the bottom six revolute joints, corresponding to the second arm, are essentially the same as for the first arm. The only difference is that, this time, under "State Targets", one inputs `phidash(`i'`)` for joint i'.

Now, instead of having to run two separate functions to solve for $\boldsymbol{\varphi}$ and $\boldsymbol{\varphi}' \in \mathbb{R}^6$, it is convenient to create a third function, encapsulating both `ik6Rarm1.m` and `ik6Rarm2.m`, as $\boldsymbol{\varepsilon} \in \mathbb{R}^6$ is always defined within $O_1x_1y_1z_1$. This function reads

```
function [phi,phidash] = ik6R(x,y,z,roll,pitch,yaw)
         <setting desired pose constants>
         [phi]     = ik6Rarm1(x,y,z,roll,pitch,yaw);
         [phidash] = ik6Rarm2(x,y,z,roll,pitch,yaw);
```

and *always* solves for both $\boldsymbol{\varphi}$ and $\boldsymbol{\varphi}' \in \mathbb{R}^6$ simultaneously, regardless of if the second arm is commented out or not. Furthermore, when walking the reader through the model, we have mentioned the six constants representing the desired pose of the tool center point. These constants update automatically on each run of the `ik6R.m` function *before* `ik6Rarm1.m` and `ik6Rarm2.m` are executed. Under `<setting desired pose constants>` within the previous code,

```
% position
set_param('KR120_R2700_2/xDesired','Value',string(x));
set_param('KR120_R2700_2/yDesired','Value',string(y));
set_param('KR120_R2700_2/zDesired','Value',string(z));
% orientation
set_param('KR120_R2700_2/RollDesired','Value',string(roll));
set_param('KR120_R2700_2/PitchDesired','Value',string(pitch));
set_param('KR120_R2700_2/YawDesired','Value',string(yaw));
```

is written, making sure the desired pose remains correct at all times, providing a correct error if subtracted from the current TCP pose.

5.5.2 Forward Kinematics

Given all coordinate systems are defined in similar fashion for both robots, the only alteration in the forward kinematics solution is the prior transformation from $O_1x_1y_1z_1$ to $O_1'x_1'y_1'z_1'$, before following the *same* solution to a solitary arm. Similar to the inverse kinematics solution, the function for the forward kinematics of the second manipulator is written in the form

```
function [H12dash,H23dash,...,H78dash] = fk6Rarm2(phidash)
         <altered forward kinematics solution>
```

and outputs *all* $\boldsymbol{H}_{ij}' \in SO(3)$ to MATLAB's workspace.

Again, creating a master forward kinematics function remains convenient, so

```
function [H12,...,H78,H12dash,...,H78dash] = fk6R(phi,phidash)
         [H12,H23,...,H78] = fk6Rarm1(phi);
         [H12dash,H23dash,...,H78dash] = fk6Rarm2(phidash);
```

outputs both $\boldsymbol{H}_{ij}$ and $\boldsymbol{H}_{ij}' \in SO(3)$ at once, regardless of if the second arm is commented out or not, in similar fashion to `ik6R.m`.

5.5.3 Static Gravity Compensation

What concerns the static gravity compensation algorithm, as the flanges (links 7 and 7′) of *both* robots are now rigidly connected with the end-effector, the combined mass of this link increases and a shift in the coordinates of the center

of mass can be observed. To account for this phenomena, we feed these altered values back to MATLAB's workspace in a manner similar to the static gravity compensation algorithm for a solitary arm. When it comes to the mass distribution, we choose to distribute this new mass *evenly* between the two robots, i.e.,

$$m_7 = m_7/2,$$
$$m_7' = m_7/2,$$

since the actual mass distribution is unknown.

Remark 5.1. One can feel free to choose a different mass distribution, e.g.,

$$m_7 = m_7/3,$$
$$m_7' = 2m_7/3,$$

so long as the actual updated mass of the end-effector remains *unchanged*. ◇

As for the center of mass coordinates, one simply updates $\boldsymbol{p}_{7\mathfrak{M}_7}$ and $\boldsymbol{p}'_{7\mathfrak{M}_7} \in \mathbb{R}^3$ from the variables fed back to the MATLAB workspace.

Upon making the aforementioned alterations, we move on to the final master function, housing static compensation algorithms for both manipulators. It reads

```
function [dphi,dphidash] =
↪  comp6R(H12,...,H78,H12dash,...,H78dash,kappa_tau,b)
       [dphi] = comp6Rarm1(H12,...,H78,kappa_tau,b);
       [dphidash] =
       ↪  comp6Rarm2(H12dash,...,H78dash,kappa_tau,b);
```

and outputs both $\Delta\boldsymbol{\varphi}$ and $\Delta\boldsymbol{\varphi}' \in \mathbb{R}^6$ simultaneously, regardless of if the second arm is commented out or not. Implementing the solutions for the second manipulator to Simscape MBS is similar to a solitary arm, only for the bottom rigid body tree (see Figure 4.7 for reference). Again, these solutions ensure the divergence from the desired pose is no more than approximately $1 \cdot 10^{-11}$ [mm] as for position, and approximately $1 \cdot 10^{-12}$ [°] as for orientation.

5.6 Linearization and SVD

Finally, we arrive at linearization and singular value decomposition, tying the model and its static stiffness together. The reason we have written previous solutions as MATLAB functions and not as scripts is because of our ability to call them all in one linearization + SVD script, without needing to run what would be their respective scripts separately.

5.6.1 Preliminary Definitions

Before commencing the static stiffness evaluation itself, we need to define the model from which the data will be sourced, along with the number of divisions and planes. The robot is defined and loaded as

```
robot = 'KR_120_R2700_2';
load_system(robot);
```

and because Simscape MBS opens a visualization of the system by default,

```
set_param(robot,'SimMechanicsOpenEditorOnUpdate','off');
```

needs to be set so the visualization is not present at all, accelerating result generation. Next, the simulation time, i.e., the time for linearization, the number of divisions of each coordinate, and the number of levels (planes) is set using

```
simTime  = 1;          % simulation time
divNum   = <integer1>; % number of divisions
sliceNum = <integer2>; % number of levels (slices)
```

where `<integer1>`, `<integer2>` are important choices as the model needs to simulate at each point, thereby massively influencing the script's runtime. For instance, simulating the model at 900 distinct points takes approximately 36 minutes. On the other hand, simulating the model 22 500 times adds up to 15 hours of runtime, exhibiting exponential behaviour.

5.6.2 Grid and Level Definitions

Next, the grid and plane levels are simply defined using the `divNum` and `sliceNum` parameters. As for grids, these definitions read

```
%% Division of axes
% x-axis division
xmin  = <number1>; xmax  = <number2>;
Xgrid = linspace(xmin,xmax,divNum);
% y-axis division
ymin  = <number3>; ymax  = <number4>;
Ygrid = linspace(ymin,ymax,divNum);
% z-axis division
zmin  = <number5>; zmax  = <number6>;
Zgrid = linspace(zmin,zmax,divNum);

%% Grids
[XmeshXY,YmeshXY] = meshgrid(Xgrid,Ygrid); % xy-planes
[YmeshYZ,ZmeshYZ] = meshgrid(Ygrid,Zgrid); % yz-planes
[XmeshXZ,ZmeshXZ] = meshgrid(Xgrid,Zgrid); % xz-planes
```

and as for the plane levels, it is

```
Zslice = linspace(zmin,zmax,sliceNum);      % xy-planes
Xslice = linspace(xmin,xmax,sliceNum);      % yz-planes
Yslice = linspace(ymin,ymax,sliceNum);      % xz-planes
```

i.e., `sliceNum` is our step size in terms of levels, Δz, Δx, or Δy.

5.6.3 Execution

In terms of execution, the script's main portion is written as a triplet of three nested `for` loops, computing static stiffness in all Cartesian planes, i.e., xy, yz, and xz. Coming back to our tensor parallel (Figure 5.2), the topmost loop is used to set the current plane level, i.e., goes through the *depth* of the tensor, and the two nested loops perform the loop found in Figure 5.1, i.e., go through the rows and columns of the tensor. In MATLAB code, this approach reads

```
for i = 1 : 1 : sliceNum
    for j = 1 : 1 : divNum
        for k = 1 : 1 : divNum
            try
               <ik6R.m>
               <fk6R.m>
               <comp6R.m>
            catch
               continue
            end
            LsysXY = linearize(robot,simTime);
            GsysXY = LsysXY.D -
             ↪  LsysXY.C*(inv(LsysXY.A))*LsysXY.B;
            [UXY,LXY,VXY] = svd(GsysXY);

            saveUxy(:,:,j,k,i) = UXY;
            saveLxy(:,j,k,i)   = diag(LXY);
            saveVxy(:,:,j,k,i) = VXY;
        end
    end
end
save finalXY saveUxy saveLxy saveVxy
```

as for the xy-planes, and vice-versa for the yz- and xz-planes. The inclusion of the `try-catch` construct within the code primarily serves the purpose of preempting potential script failures when the `ik6R.m` function encounters difficulty in determining a solution for the current desired pose of the tool center point. In

such scenarios, program execution seamlessly transitions to the designated `catch` block, facilitating continuity from the subsequent iteration. Conversely, when the `ik6R.m` function successfully computes a solution, the robotic system assumes that position, which is then linearized using MATLAB's `linearize` function. Post-linearization, the system's transfer function `Gsys<plane>` undergoes singular value decomposition, with the resulting matrices stored within either a five-dimensional tensor or a four-dimensional equivalent. Following evaluation at all points, the script generates a `final<plane>.mat` file, housing all three tensors.[31]

5.6.4 Simulation Parameters

Given the exponential behaviour of the runtime of the script, balance needs to be found between the accuracy of the results and the actual time it takes to generate them. For such reasons, we choose to perform *every* simulation with the parameters presented within the following table.

	Parameter	
	`divNum`	`sliceNum`
Value	30	3
Number of Points	$30 \times 30 \times 3 \times 3 = 8\,100$	
Estimated runtime [h]	approx. 5.6	

Table 5.2 Simulation parameters.

On the other hand, the coordinate range, i.e., the values of `xmin`, `xmax`, `ymin`, `ymax`, `zmin`, and `zmax`, have been chosen so that the "ik6R.m" function yields as many solutions as possible, which in turn results in a larger number of points being evaluated. For these variables, we set the values shown in Table 5.3.

Variable	`xmin`	`xmax`	`ymin`	`ymax`	`zmin`	`zmax`
Value [mm]	1 700	2 200	−500	500	1 200	1 800

Table 5.3 Simulation work envelope limits.

As the selected work envelope decreases, the proximity of adjacent points increases, thereby enhancing the understanding of static stiffness distribution. Ideally, the entire work envelope of the robotic system should be evaluated. However, due to the necessity for significantly larger values of `divNum` and `sliceNum` in this scenario, the runtime of the script would become prohibitively long, precluding the acquisition of results within a reasonable timeframe.

[31] These data files are subsequently loaded for result generation.

6
Results and Discussion

Finally, we arrive at the outcomes of our investigation into static stiffness. To facilitate visualization, a distinct script was developed alongside the data acquisition script. Subsequently, the `.mat` files were loaded into this script using

```
load('final<plane>.mat');
```

for each of the three respective data files. Then, after arranging and reformatting the gathered data into manageable arrays, stiffness maps — previously discussed in the preceding chapter — were generated as a fusion of MATLAB's `quiver` and `contour` plots. Henceforth, we shall refer to them as *quiver-contour* plots. Additionally, with each instance of result generation, the minimum, maximum, and median values were recorded for both minimum and maximum principal stiffnesses. This was done to precisely assess the influence of robot coupling on the values directly, as well as on stiffness homogeneity (see further).

Capitalizing on the spatial nature of the problem, we explored two potential orientations of the tool center point. In one scenario, the tool center point faces downwards, with Euler XYZ angles set to $\boldsymbol{\phi} = [0 \quad 0 \quad 0]^T \in \mathbb{R}^3$. Conversely, in the other scenario, the tool center point approached the *same* set of points with an arbitrary, non-zero orientation. In such instances, the Euler XYZ angles were set to the values $\boldsymbol{\phi} = [10 \quad 5 \quad 20]^T \in \mathbb{R}^3$ degrees.

While investigating two possible approaches of the tool center point further enhances our understanding of the behaviour of the static stiffness function. Yet, the vast array of potential cases renders comprehensive explanation impractical. Instead, a dedicated page precedes each set of results, offering clarity and orientation for the reader. Generally, each case is covered by a quadruplet of pages, with the first one giving the reader an introduction into the specific case, whilst the other three showcase all of the gathered results during simulation.

6.1 **Case s-C1.** Solitary Robot Operating Vertically in xy-Planes

Parameters	Configuration	Solitary
	Plane(s)	xy (z-levels)
	Euler XYZ angles	$\boldsymbol{\phi} = [0 \quad 0 \quad 0]^T \in \mathbb{R}^3$

		Description
Contents	Minima quiver-contour plot for Case s-C1	Static stiffness minima, minima directional field.
	Maxima quiver-contour plot for Case s-C1	Static stiffness maxima, maxima directional field.
	Results table for Case s-C1	Minima and maxima values, static stiffness homogeneity.

The directional field vectors in Figures 6.5 and 6.6 are uniformly scaled.

Stiffness homogeneity taken as the ratio between maximum principal stiffness and minimum principal stiffness medians.

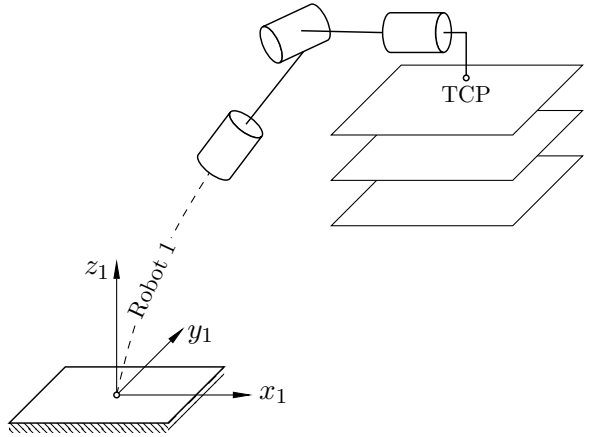


Figure 6.1 Illustration of Case s-C1. Not to scale.

All other simulation parameters remain the same to those disclosed in Tables 5.2 and 5.3.

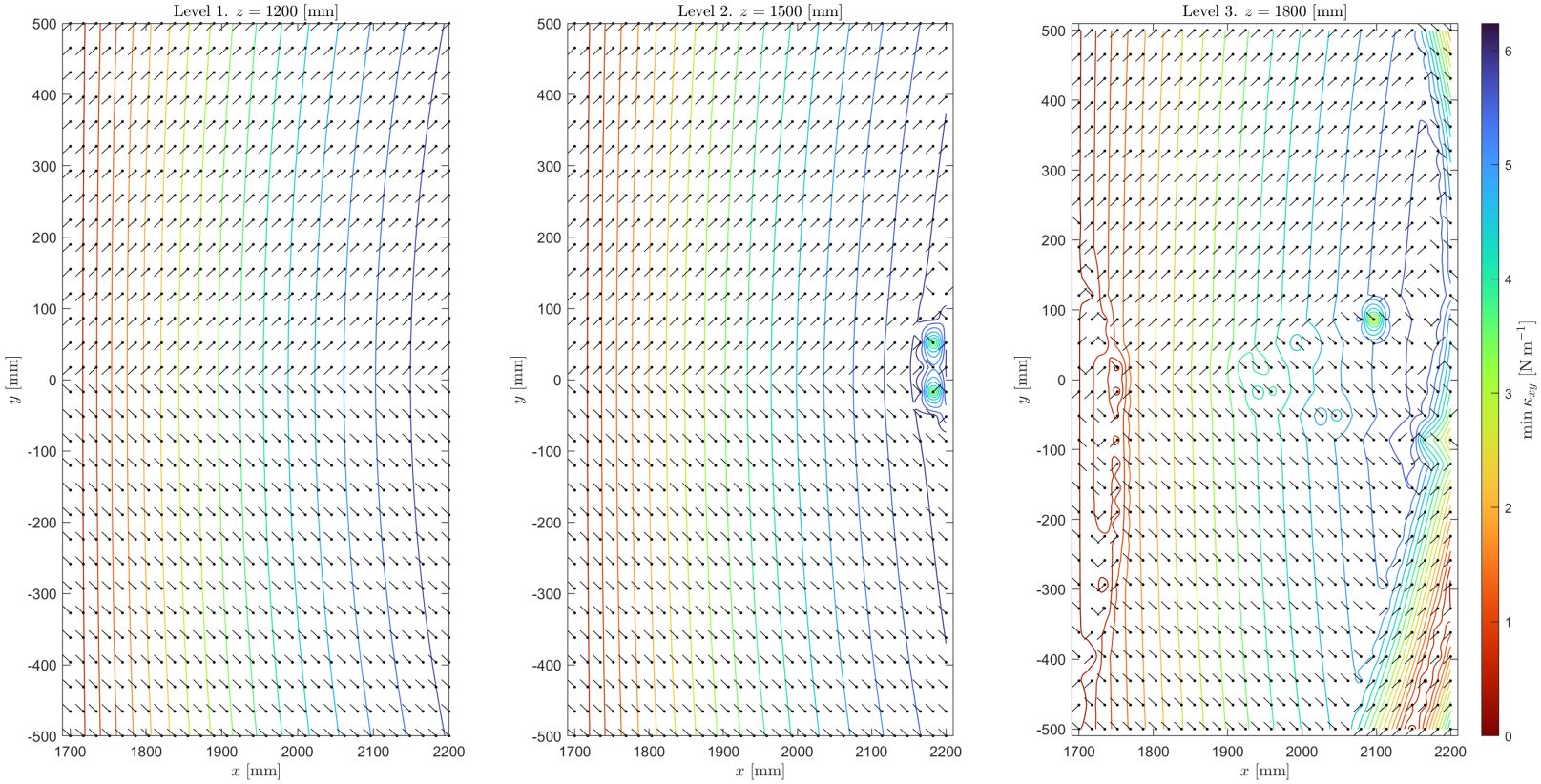


Figure 6.2 Minima quiver-contour plot for Case s-C1. Solitary robot, xy-planes, $\boldsymbol{\phi} = [0 \quad 0 \quad 0]^T \in \mathbb{R}^3$.

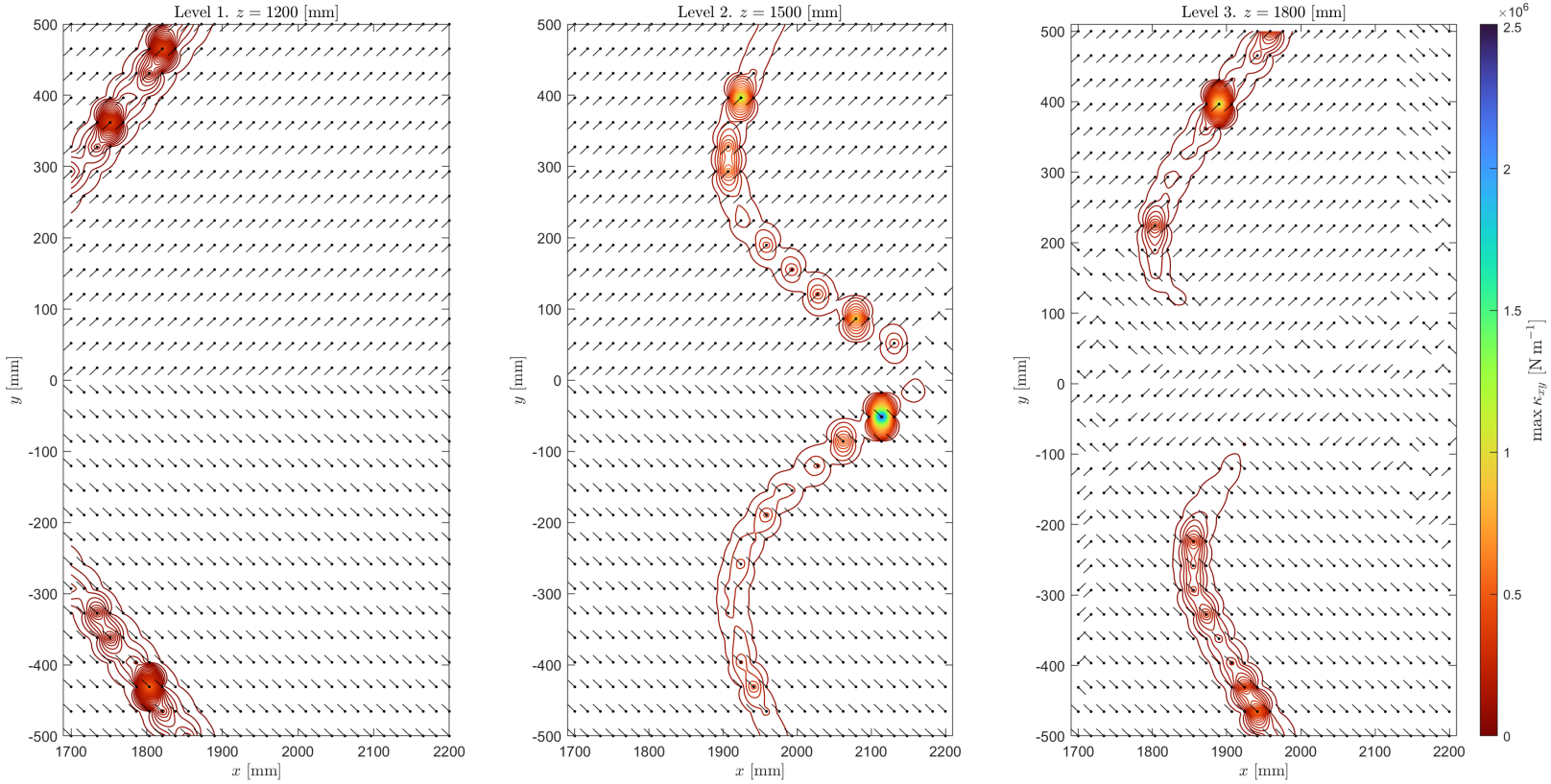


Figure 6.3 Maxima quiver-contour plot for Case s-C1. Solitary robot, xy-planes, $\boldsymbol{\phi} = [0 \quad 0 \quad 0]^T \in \mathbb{R}^3$.

Level 1. $z = 1\,200$ [mm]					
Minimum principal stiffness xy [$\mathrm{N\,m^{-1}}$]		Maximum principal stiffness xy [$\mathrm{N\,m^{-1}}$]			
min	0.317	min	2 905.725		
max	6.080	max	602 210.840		
med	3.900	med	4 934.055	homogeneity	1 265.810
Level 2. $z = 1\,500$ [mm]					
Minimum principal stiffness xy [$\mathrm{N\,m^{-1}}$]		Maximum principal stiffness xy [$\mathrm{N\,m^{-1}}$]			
min	0.317	min	734.450		
max	6.237	max	2 508 995.890		
med	3.885	med	9 278.310	homogeneity	2 388.365
Level 3. $z = 1\,800$ [mm]					
Minimum principal stiffness xy [$\mathrm{N\,m^{-1}}$]		Maximum principal stiffness xy [$\mathrm{N\,m^{-1}}$]			
min	0.001	min	10.800		
max	5.960	max	1 248 840.670		
med	3.570	med	4 652.400	homogeneity	1 304.300

Table 6.1 Results table for Case s-C1.

6.2 **Case s-C2.** Solitary Robot Operating Vertically in yz-Planes

Parameters	Configuration	Solitary
	Plane(s)	yz (x-levels)
	Euler XYZ angles	$\boldsymbol{\phi} = [0 \quad 0 \quad 0]^T \in \mathbb{R}^3$

		Description
Contents	Minima quiver-contour plot for Case s-C2	Static stiffness minima, minima directional field.
	Maxima quiver-contour plot for Case s-C2	Static stiffness maxima, maxima directional field.
	Results table for Case s-C2	Minima and maxima values, static stiffness homogeneity.

The directional field vectors in Figures 6.5 and 6.6 are uniformly scaled.

Stiffness homogeneity taken as the ratio between maximum principal stiffness and minimum principal stiffness medians.

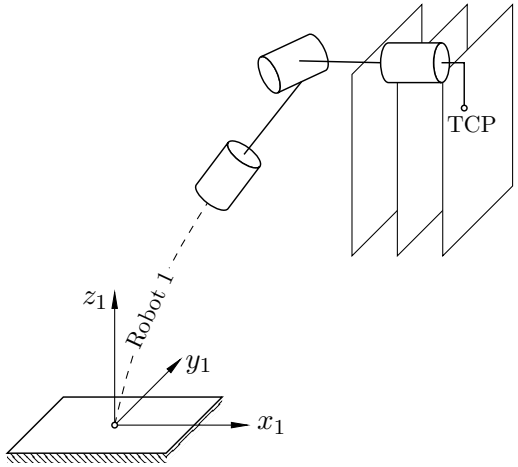


Figure 6.4 Illustration of Case s-C2. Not to scale.

All other simulation parameters remain the same to those disclosed in Tables 5.2 and 5.3.

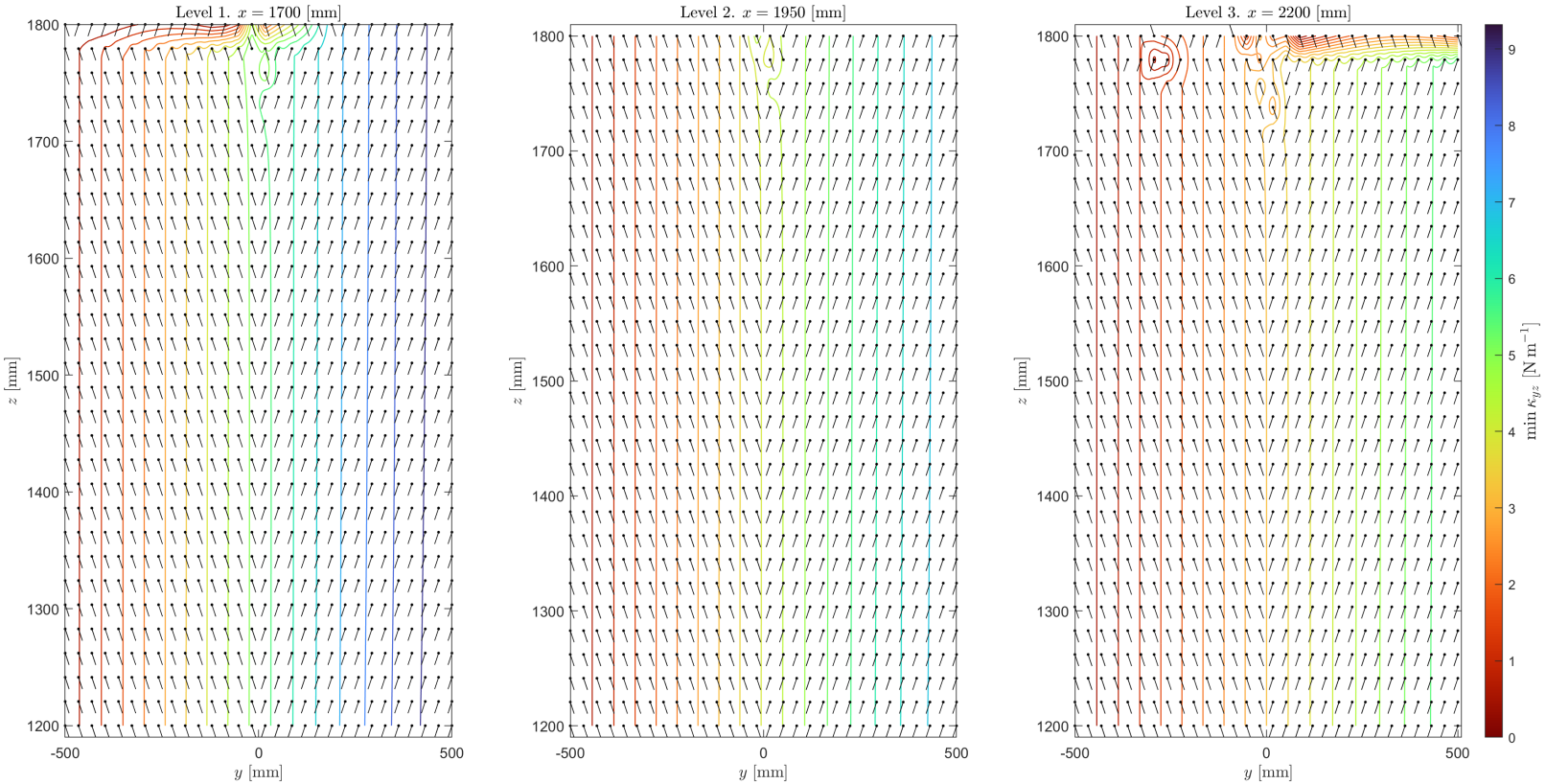


Figure 6.5 Minima quiver-contour plot for Case s-C2. Solitary robot, yz-planes, $\boldsymbol{\phi} = [0 \quad 0 \quad 0]^T \in \mathbb{R}^3$.

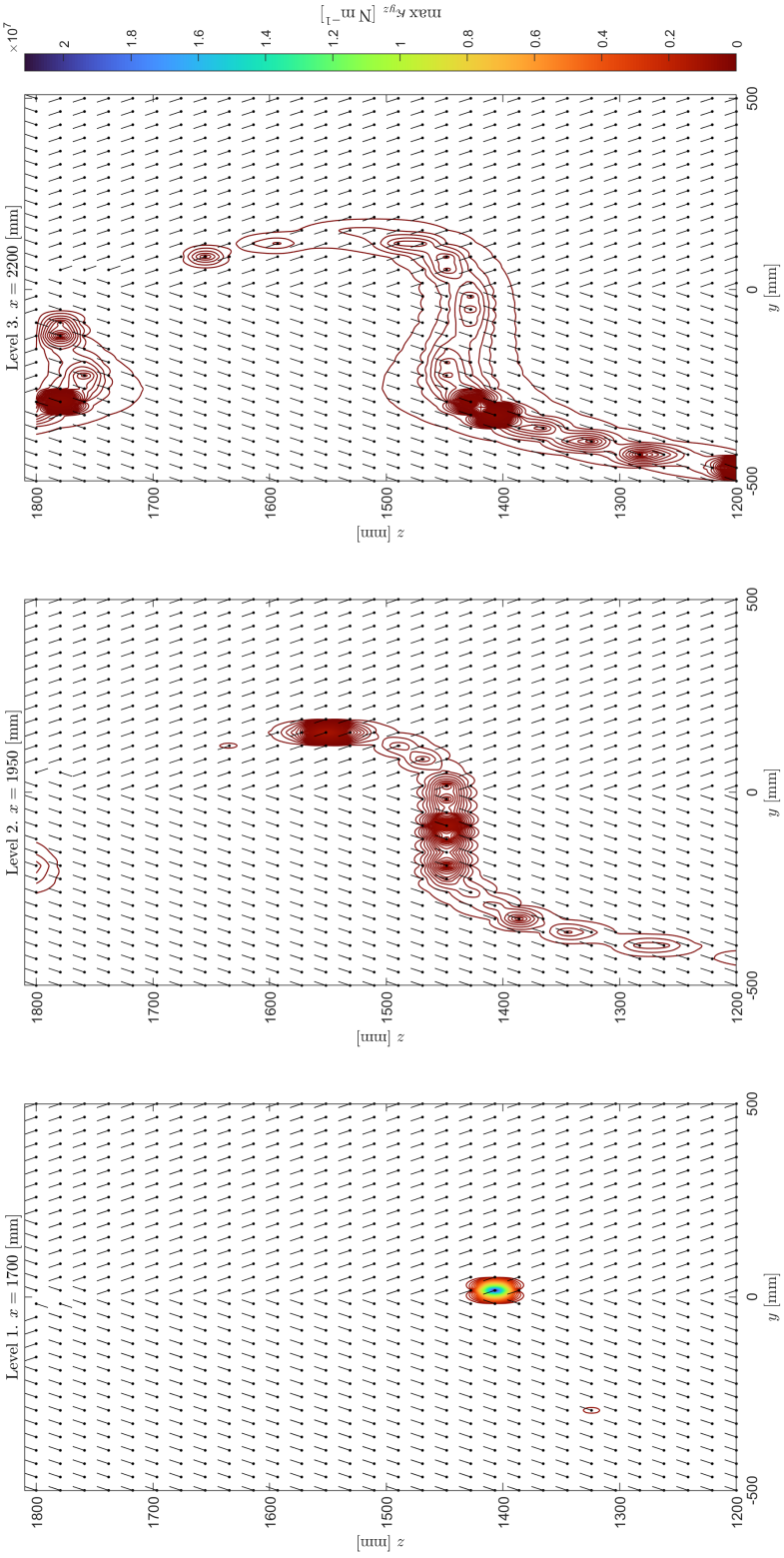


Figure 6.6 Maxima quiver-contour plot for Case s-C2. Solitary robot, yz-planes, $\boldsymbol{\phi} = [0 \quad 0 \quad 0]^T \in \mathbb{R}^3$.

Level 1. $x = 1\,700$ [mm]					
Minimum principal stiffness yz [$\mathrm{N\,m^{-1}}$]		Maximum principal stiffness yz [$\mathrm{N\,m^{-1}}$]			
min	0.120	min	79.105		
max	9.320	max	21 161 447.300		
med	5.085	med	8 650.430	homogeneity	1 701.250
Level 2. $x = 1\,950$ [mm]					
Minimum principal stiffness yz [$\mathrm{N\,m^{-1}}$]		Maximum principal stiffness yz [$\mathrm{N\,m^{-1}}$]			
min	0.245	min	120.575		
max	7.250	max	1 338 879.380		
med	4.000	med	6 808.800	homogeneity	1 704.500
Level 3. $x = 2\,200$ [mm]					
Minimum principal stiffness yz [$\mathrm{N\,m^{-1}}$]		Maximum principal stiffness yz [$\mathrm{N\,m^{-1}}$]			
min	0.195	min	26.940		
max	5.785	max	500 358.420		
med	3.050	med	5 340.130	homogeneity	1 750.500

Table 6.2 Results table for Case s-C2.

6.3 **Case s-C3.** Solitary Robot Operating Vertically in xz-Planes

Parameters	Configuration	Solitary
	Plane(s)	xz (y-levels)
	Euler XYZ angles	$\boldsymbol{\phi} = [0 \quad 0 \quad 0]^T \in \mathbb{R}^3$
		Description
Contents	Minima quiver-contour plot for Case s-C3	Static stiffness minima, minima directional field.
	Maxima quiver-contour plot for Case s-C3	Static stiffness maxima, maxima directional field.
	Results table for Case s-C3	Minima and maxima values, static stiffness homogeneity.

The directional field vectors in Figures 6.8 and 6.9 are uniformly scaled.

Stiffness homogeneity taken as the ratio between maximum principal stiffness and minimum principal stiffness medians.

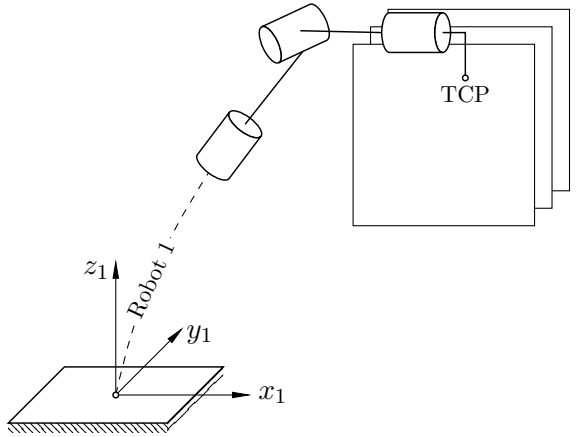


Figure 6.7 Illustration of Case s-C3. Not to scale.

All other simulation parameters remain the same to those disclosed in Tables 5.2 and 5.3.

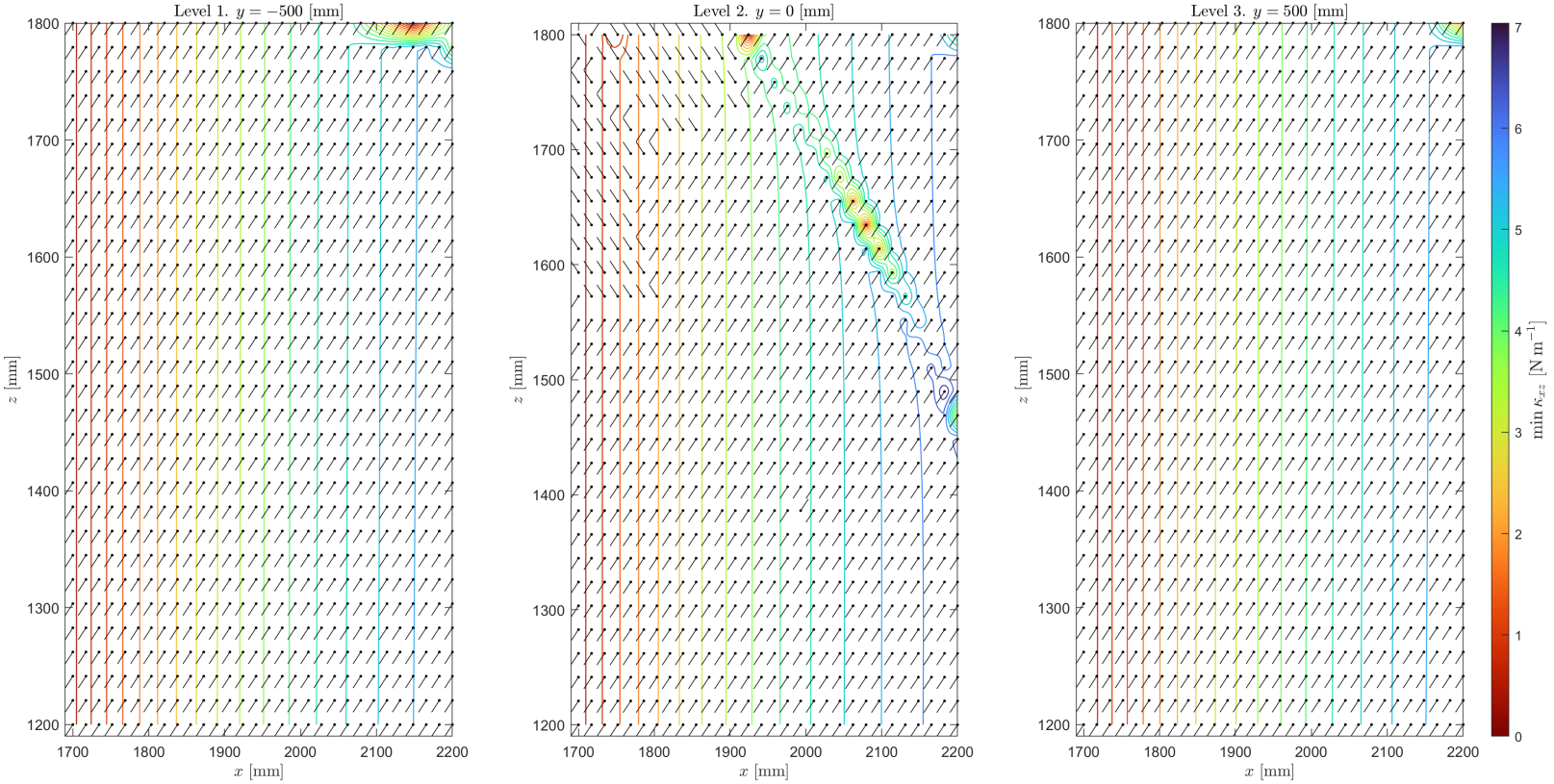


Figure 6.8 Minima quiver-contour plot for Case s-C3. Solitary robot, xz-planes, $\boldsymbol{\phi} = [0 \quad 0 \quad 0]^T \in \mathbb{R}^3$.

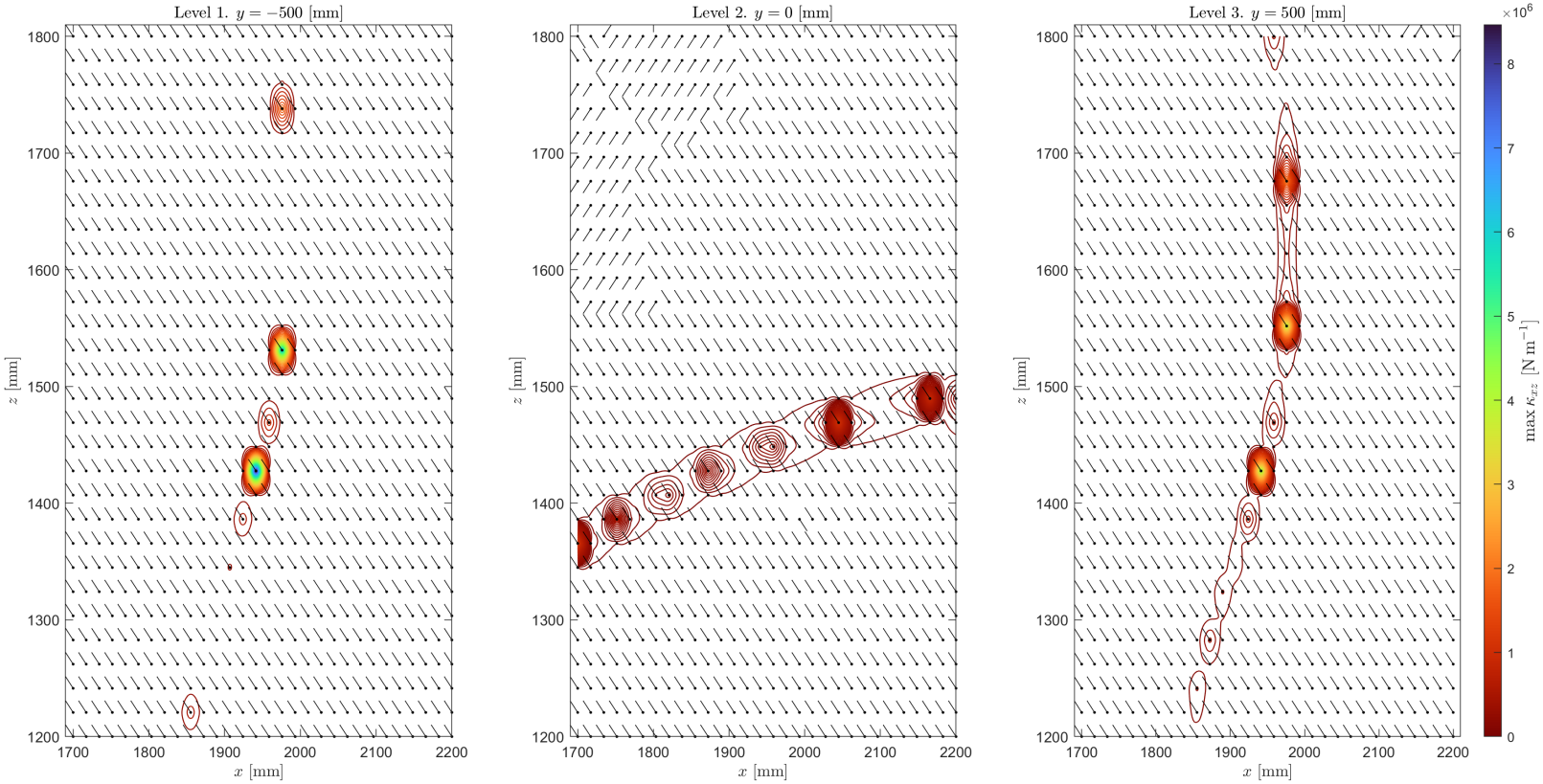


Figure 6.9 Maxima quiver-contour plot for Case s-C3. Solitary robot, xz-planes, $\boldsymbol{\phi} = [0 \quad 0 \quad 0]^T \in \mathbb{R}^3$.

Level 1. $y = -500$ [mm]					
Minimum principal stiffness xz [$\mathrm{N\,m^{-1}}$]		Maximum principal stiffness xz [$\mathrm{N\,m^{-1}}$]			
min	0.040	min	1 322.000		
max	5.800	max	8 466 771.300		
med	3.660	med	7 699.725	homogeneity	2 106.000
Level 2. $y = 0$ [mm]					
Minimum principal stiffness xz [$\mathrm{N\,m^{-1}}$]		Maximum principal stiffness xz [$\mathrm{N\,m^{-1}}$]			
min	0.065	min	87.765		
max	7.000	max	1 133 691.160		
med	4.000	med	4 470.700	homogeneity	1 145.380
Level 3. $y = 500$ [mm]					
Minimum principal stiffness xz [$\mathrm{N\,m^{-1}}$]		Maximum principal stiffness xz [$\mathrm{N\,m^{-1}}$]			
min	0.320	min	1 325.875		
max	5.785	max	4 303 868.955		
med	3.660	med	7 527.873	homogeneity	2 056.000

Table 6.3 Results table for Case s-C3.

6.4 **Case c-C1.** Coupled Robots Operating Vertically in xy-Planes

Parameters	Configuration	Coupled
	Plane(s)	xy (z-levels)
	Euler XYZ angles	$\boldsymbol{\phi} = [0 \quad 0 \quad 0]^T \in \mathbb{R}^3$

		Description
Contents	Minima quiver-contour plot for Case c-C1	Static stiffness minima, minima directional field.
	Maxima quiver-contour plot for Case c-C1	Static stiffness maxima, maxima directional field.
	Results table for Case c-C1	Minima and maxima values, static stiffness homogeneity.

The directional field vectors in Figures 6.11 and 6.12 are uniformly scaled.

Stiffness homogeneity taken as the ratio between maximum principal stiffness and minimum principal stiffness medians.

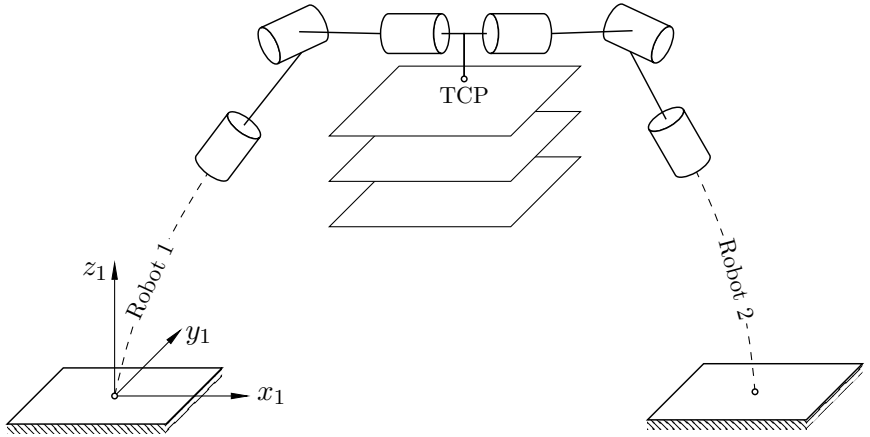


Figure 6.10 Illustration of Case c-C1. Not to scale.

All other simulation parameters remain the same to those disclosed in Tables 5.2 and 5.3.

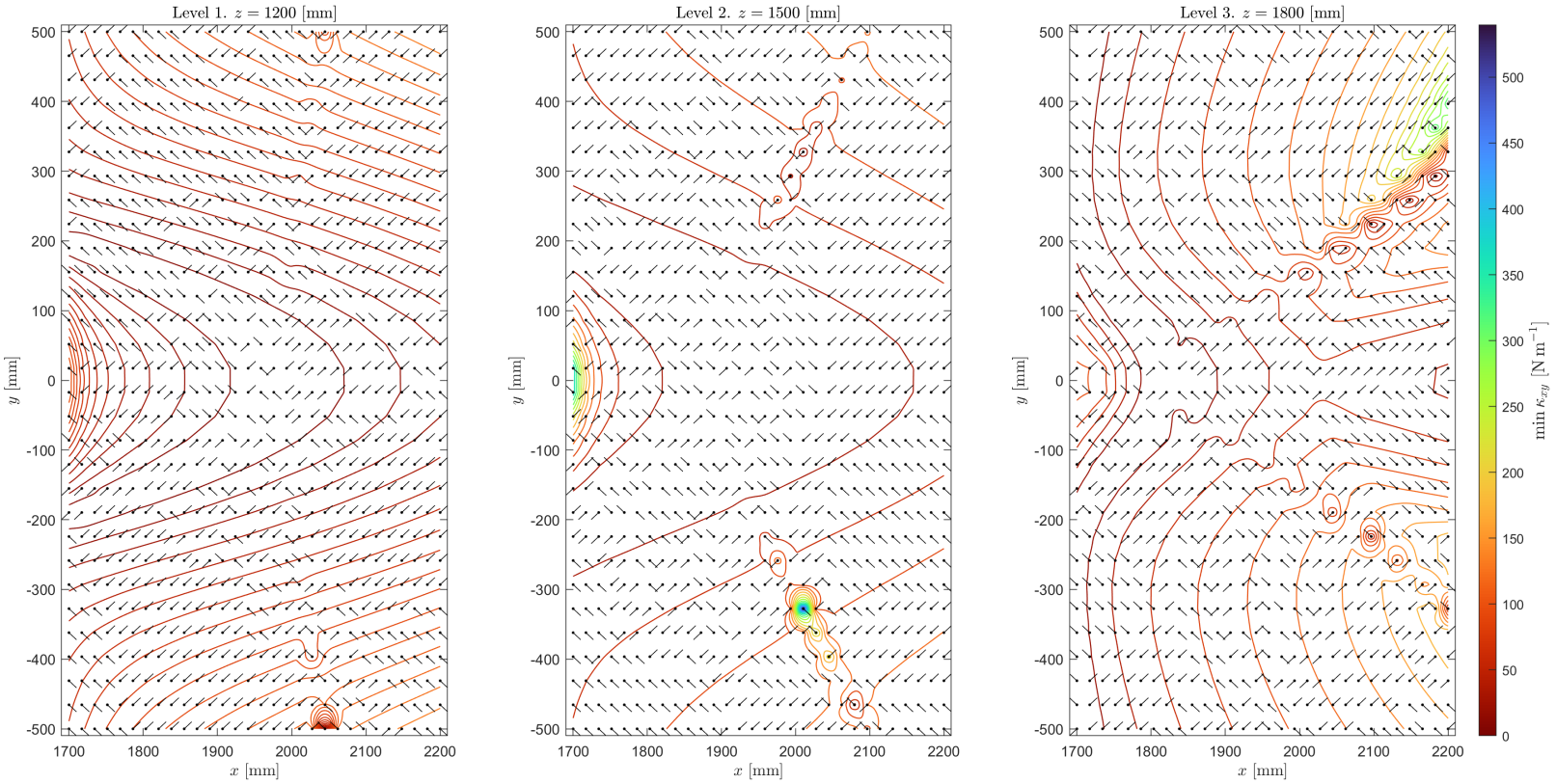


Figure 6.11 Minima quiver-contour plot for Case c-C1. Coupled robots, xy-planes, $\boldsymbol{\phi} = [0 \quad 0 \quad 0]^T \in \mathbb{R}^3$.

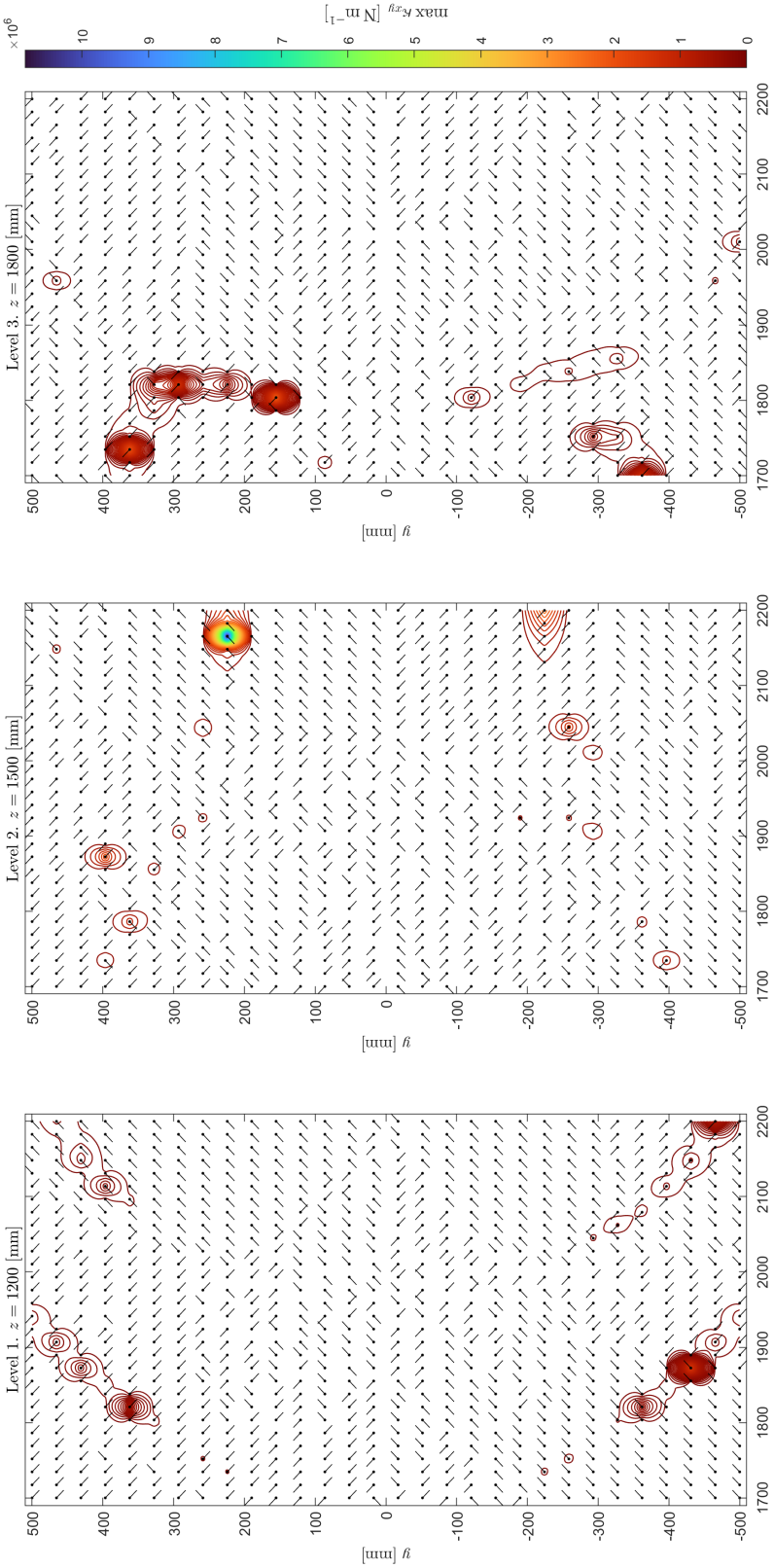


Figure 6.12 Maxima quiver-contour plot for Case c-C1. Coupled robots, xy-planes, $\boldsymbol{\phi} = [0 \quad 0 \quad 0]^T \in \mathbb{R}^3$.

Level 1. $z = 1\,200$ [mm]					
Minimum principal stiffness xy [$\mathrm{N\,m^{-1}}$]		Maximum principal stiffness xy [$\mathrm{N\,m^{-1}}$]			
min	0.160	min	2 061.160		
max	141.550	max	1 496 764.725		
med	47.330	med	5 548.445	homogeneity	117.230
Level 2. $z = 1\,500$ [mm]					
Minimum principal stiffness xy [$\mathrm{N\,m^{-1}}$]		Maximum principal stiffness xy [$\mathrm{N\,m^{-1}}$]			
min	0.110	min	1 502.500		
max	540.000	max	10 857 505.600		
med	58.710	med	16 350.000	homogeneity	278.500
Level 3. $z = 1\,800$ [mm]					
Minimum principal stiffness xy [$\mathrm{N\,m^{-1}}$]		Maximum principal stiffness xy [$\mathrm{N\,m^{-1}}$]			
min	0.520	min	106.000		
max	333.430	max	2 279 293.300		
med	74.245	med	6 699.660	homogeneity	90.238 954

Table 6.4 Results table for Case c-C1.

6.5 **Case c-C2.** Coupled Robots Operating Vertically in yz-Planes

Parameters	Configuration	Coupled
	Plane(s)	yz (x-levels)
	Euler XYZ angles	$\boldsymbol{\phi} = [0 \quad 0 \quad 0]^T \in \mathbb{R}^3$

		Description
Contents	Minimum quiver-contour plot for Case c-C2	Static stiffness minima, minima directional field.
	Maxima quiver-contour plot for Case c-C2	Static stiffness maxima, maxima directional field.
	Results table for Case c-C2	Minima and maxima values, static stiffness homogeneity.

The directional field vectors in Figures 6.14 and 6.15 are uniformly scaled.

Stiffness homogeneity taken as the ratio between maximum principal stiffness and minimum principal stiffness medians.

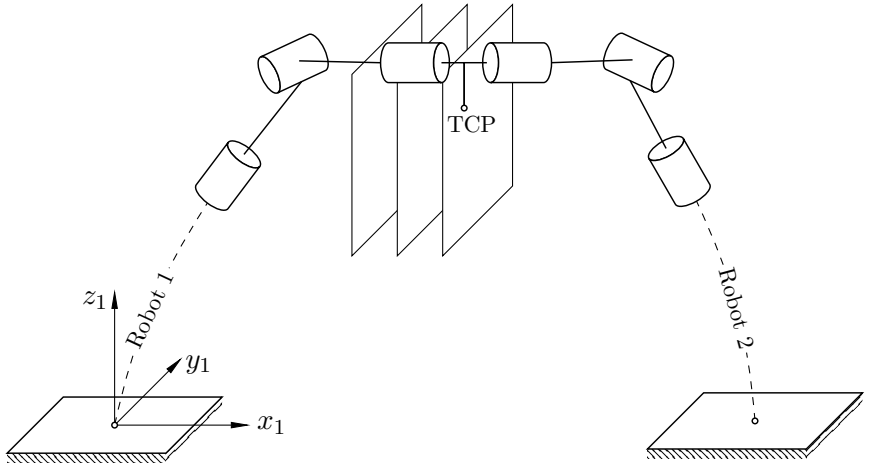


Figure 6.13 Illustration of Case c-C2. Not to scale.

All other simulation parameters remain the same to those disclosed in Tables 5.2 and 5.3.

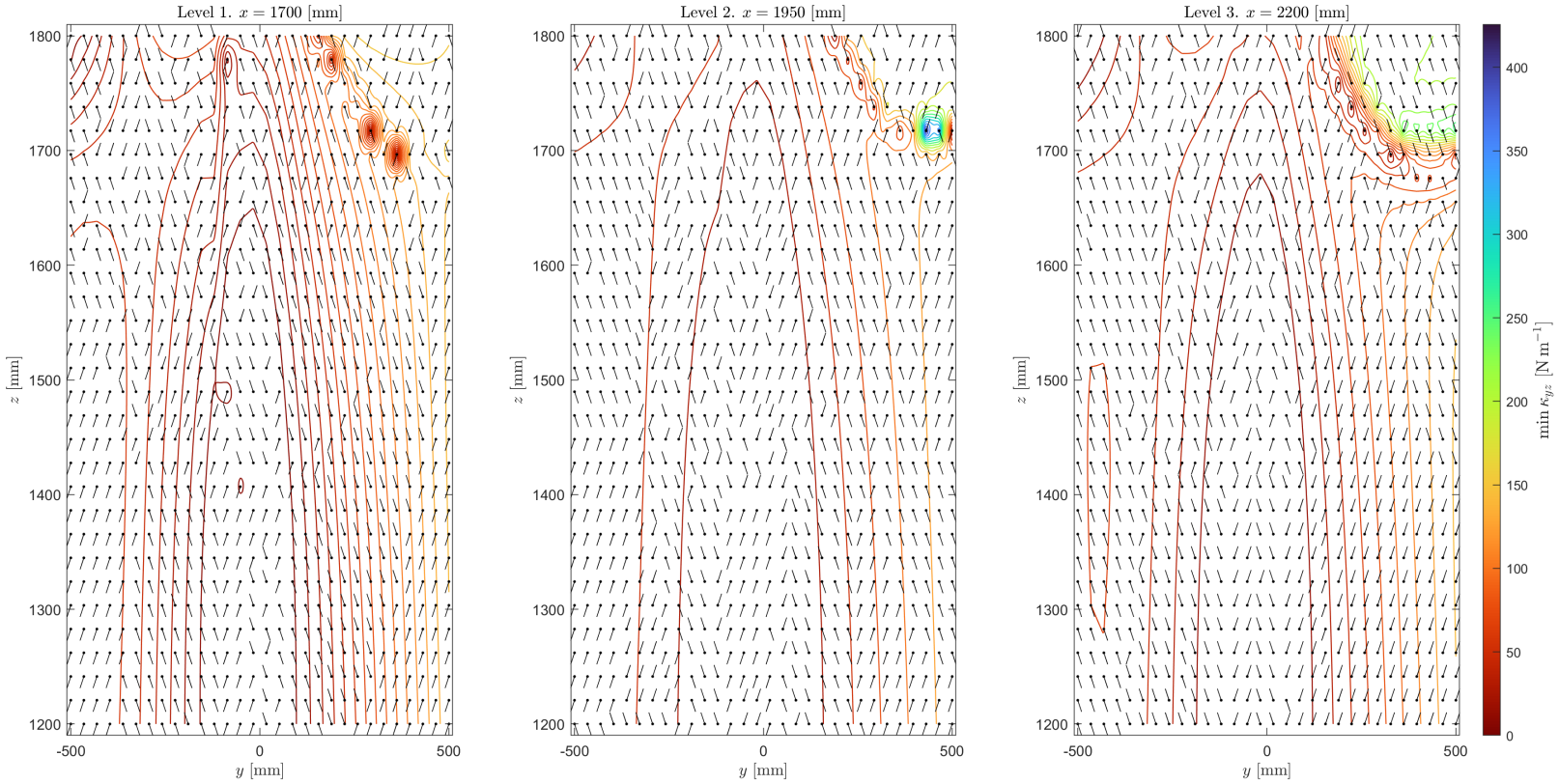


Figure 6.14 Minima quiver-contour plot for Case c-C2. Coupled robots, yz-planes, $\boldsymbol{\phi} = [0 \quad 0 \quad 0]^T \in \mathbb{R}^3$.

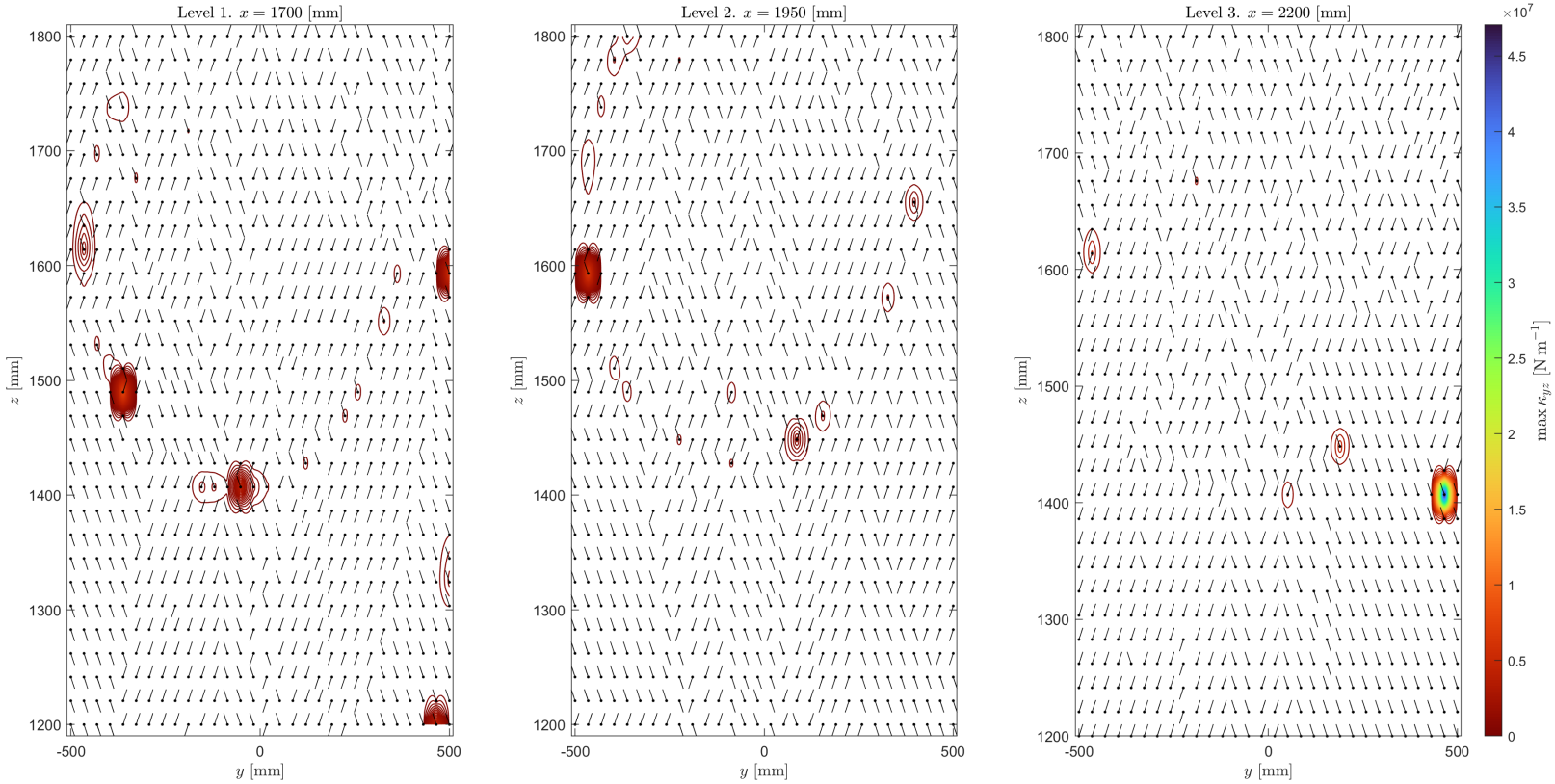


Figure 6.15 Maxima quiver-contour plot for Case c-C2. Coupled robots, yz-planes, $\boldsymbol{\phi} = [0 \quad 0 \quad 0]^T \in \mathbb{R}^3$.

Level 1. $x = 1\,700$ [mm]					
Minimum principal stiffness yz [$\mathrm{N\,m^{-1}}$]		Maximum principal stiffness yz [$\mathrm{N\,m^{-1}}$]			
min	0.115	min	133.165		
max	161.510	max	7 970 399.000		
med	53.145	med	10 153.580	homogeneity	191.000
Level 2. $x = 1\,950$ [mm]					
Minimum principal stiffness yz [$\mathrm{N\,m^{-1}}$]		Maximum principal stiffness yz [$\mathrm{N\,m^{-1}}$]			
min	0.175	min	135.670		
max	425.710	max	7 013 218.070		
med	51.670	med	9 401.940	homogeneity	181.860
Level 3. $x = 2\,200$ [mm]					
Minimum principal stiffness yz [$\mathrm{N\,m^{-1}}$]		Maximum principal stiffness yz [$\mathrm{N\,m^{-1}}$]			
min	0.055	min	105.625		
max	270.000	max	47 072 430.852		
med	51.700	med	11 039.920	homogeneity	213.440

Table 6.5 Results table for Case c-C2.

6.6 **Case c-C3.** Coupled Robots Operating Vertically in xz-Planes

Parameters	Configuration	Coupled
	Plane(s)	xz (y-levels)
	Euler XYZ angles	$\boldsymbol{\phi} = [0 \quad 0 \quad 0]^T \in \mathbb{R}^3$

		Description
Contents	Minima quiver-contour plot for Case c-C3	Static stiffness minima, minima directional field.
	Maxima quiver-contour plot for Case c-C3	Static stiffness maxima, maxima directional field.
	Results table for Case c-C3	Minima and maxima values, static stiffness homogeneity.

The directional field vectors in Figures 6.17 and 6.18 are uniformly scaled.

Stiffness homogeneity taken as the ratio between maximum principal stiffness and minimum principal stiffness medians.

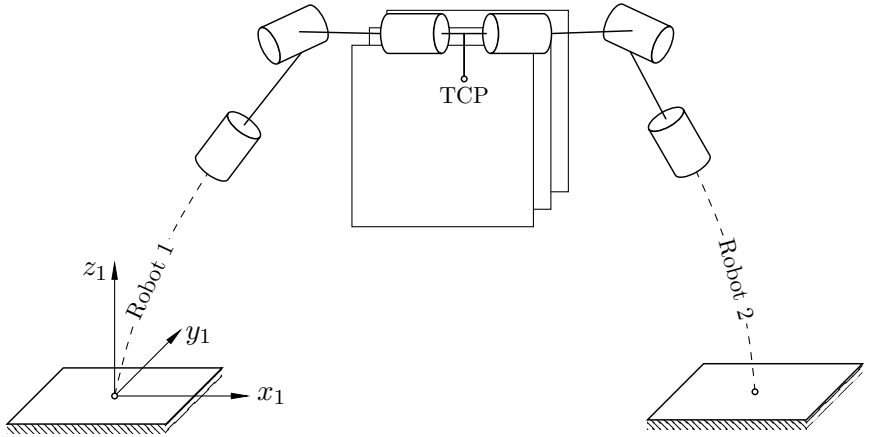


Figure 6.16 Illustration of Case c-C3. Not to scale.

All other simulation parameters remain the same to those disclosed in Tables 5.2 and 5.3.

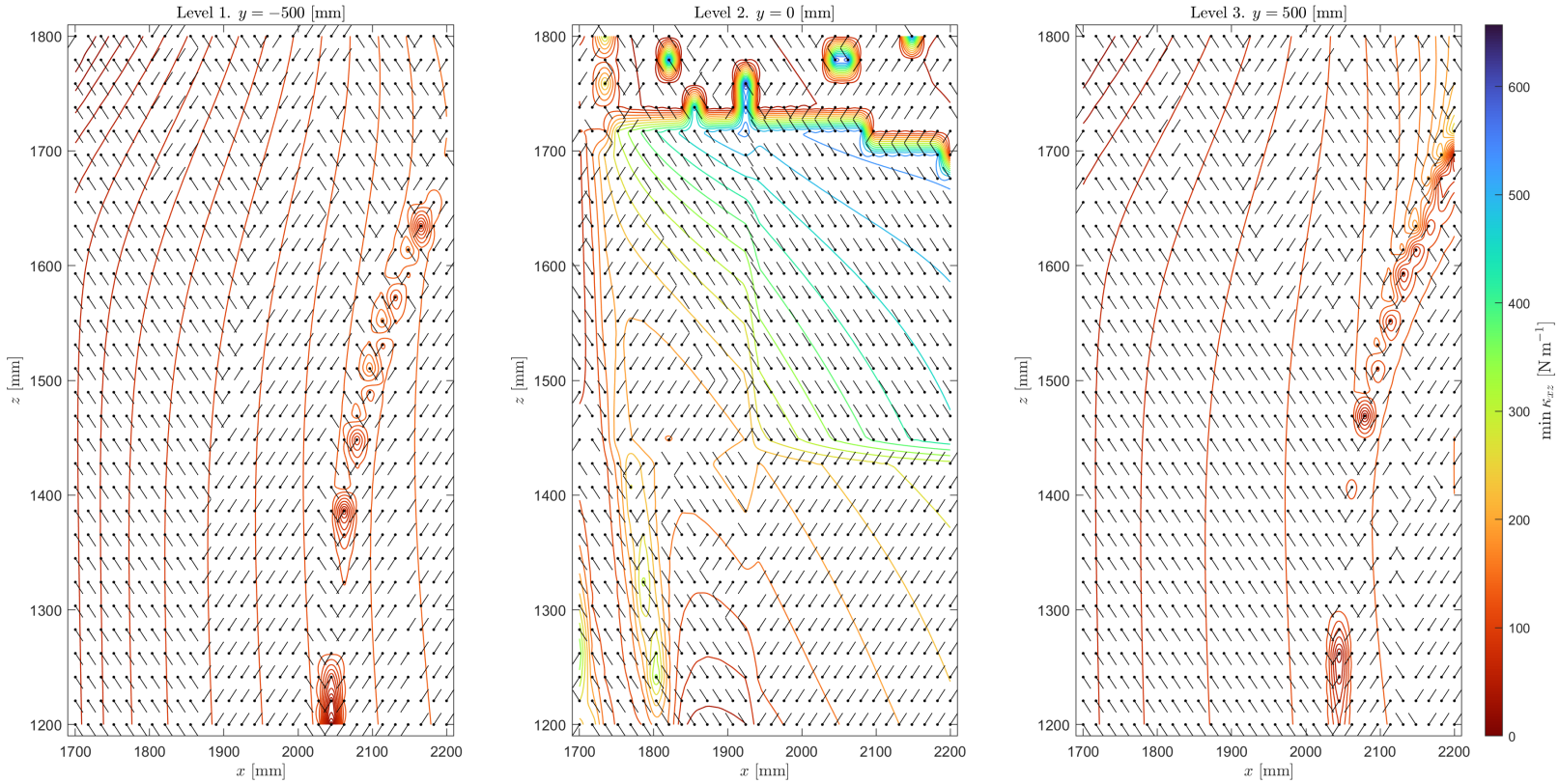


Figure 6.17 Minima quiver contour plot for Case c-C3. Coupled robots, xz-planes, $\boldsymbol{\phi} = [0 \quad 0 \quad 0]^T \in \mathbb{R}^3$.

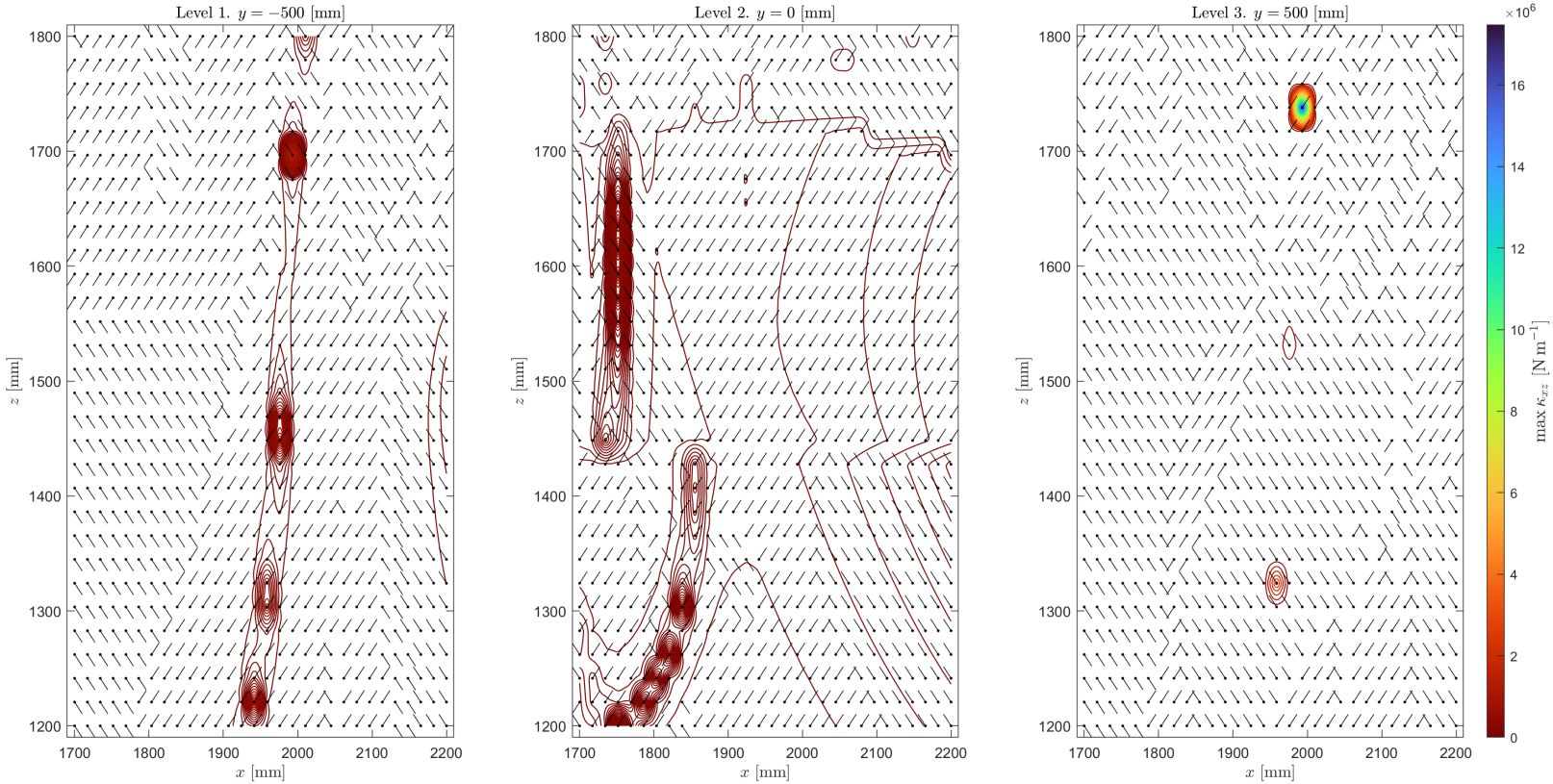


Figure 6.18 Maxima quiver-contour plot for Case c-C3. Coupled robots, xz-planes, $\boldsymbol{\phi} = [0 \quad 0 \quad 0]^T \in \mathbb{R}^3$.

Level 1. $y = -500$ [mm]					
Minimum principal stiffness xz [$\mathrm{N\,m^{-1}}$]		Maximum principal stiffness xz [$\mathrm{N\,m^{-1}}$]			
min	4.275	min	2 891.200		
max	167.300	max	1 417 744.760		
med	106.053	med	10 572.840	homogeneity	100.000
Level 2. $y = 0$ [mm]					
Minimum principal stiffness xz [$\mathrm{N\,m^{-1}}$]		Maximum principal stiffness xz [$\mathrm{N\,m^{-1}}$]			
min	0.580	min	207.333		
max	657.000	max	403 027.331		
med	204.900	med	20 698.050	homogeneity	101.000
Level 3. $y = 500$ [mm]					
Minimum principal stiffness xz [$\mathrm{N\,m^{-1}}$]		Maximum principal stiffness xz [$\mathrm{N\,m^{-1}}$]			
min	4.380	min	2 876.855		
max	270.000	max	17 475 843.842		
med	105.582	med	9 825.760	homogeneity	93.065

Table 6.6 Results table for Case c-C3.

6.7 Comparison of Cases s-C1-3 and c-C1-3

		Description
Contents	Comparison of Cases c-C1 and s-C1	Ratios between values for Case c-C1 and Case s-C1
	Comparison of Cases c-C2 and s-C2	Ratios between values for Case c-C2 and Case s-C2
	Comparison of Cases c-C3 and s-C3	Ratios between values for Case c-C3 and Case s-C3
Legend	<number>	Improvement after coupling.
	<number>	Deterioration after coupling.
	<number>	No change after coupling.

- **Improvement After Coupling.** As for <number> static stiffness values, larger ratio means greater improvement, i.e., the coupled configuration exhibits <number>-times larger the respective value. As for <number> stiffness homogeneities, smaller ratio means greater improvement, i.e., the coupled configuration's stiffness homogeneity is <number>-times the stiffness homogeneity of a solitary robot, resulting in the coupled configuration showing an overall increase in static stiffness homogeneity by a factor of 1/<number> over the solitary configuration.
- **Deterioration After Coupling.** As for <number> static stiffness values, smaller ratio means greater deterioration, i.e., the coupled configuration exhibits <number>-times smaller the respective value. As for <number> stiffness homogeneities, larger ratio means greater deterioration, i.e., the coupled configuration's stiffness homogeneity is <number>-times the stiffness homogeneity of a solitary robot, resulting in the coupled configuration showing an overall decrease in static stiffness homogeneity by a factor of 1/<number> over the solitary configuration.

Level 1. $z = 1\,200$ [mm]					
Minimum principal stiffness xy ratio		Maximum principal stiffness xy ratio			
min	0.495	min	0.710		
max	23.280	max	2.485		
med	12.140	med	1.125	homogeneity ratio	0.090
Level 2. $z = 1\,500$ [mm]					
Minimum principal stiffness xy ratio		Maximum principal stiffness xy ratio			
min	0.340	min	2.045		
max	86.546	max	4.330		
med	15.112	med	1.765	homogeneity ratio	0.116
Level 3. $z = 1\,800$ [mm]					
Minimum principal stiffness xy ratio		Maximum principal stiffness xy ratio			
min	598.180	min	9.826		
max	55.900	max	1.825		
med	20.810	med	1.440	homogeneity ratio	0.070

Table 6.7 Comparison of Cases c-C1 and s-C1. Ratios taken as c-C1/s-C1.

Level 1. $x = 1\,700$ [mm]					
Minimum principal stiffness yz ratio		Maximum principal stiffness yz ratio			
min	0.970	min	1.683		
max	17.325	max	0.375		
med	10.450	med	1.173	homogeneity ratio	0.112
Level 2. $x = 1\,950$ [mm]					
Minimum principal stiffness yz ratio		Maximum principal stiffness yz ratio			
min	0.719	min	1.125		
max	58.733	max	5.238		
med	12.936	med	1.381	homogeneity ratio	0.107
Level 3. $x = 2\,200$ [mm]					
Minimum principal stiffness yz ratio		Maximum principal stiffness yz ratio			
min	0.280	min	3.920		
max	46.677	max	94.077		
med	16.955	med	2.070	homogeneity ratio	0.122

Table 6.8 Comparison of Cases c-C2 and s-C2. Ratios takes as c-C2/s-C2.

Level 1. $y = -500$ [mm]					
Minimum principal stiffness xz ratio		Maximum principal stiffness xz ratio			
min	117.760	min	2.185		
max	28.953	max	0.167		
med	29.000	med	1.373	homogeneity ratio	0.047
Level 2. $y = 0$ [mm]					
Minimum principal stiffness xz ratio		Maximum principal stiffness xz ratio			
min	8.910	min	2.362		
max	93.360	max	0.355		
med	52.494	med	4.630	homogeneity ratio	0.088
Level 3. $y = 500$ [mm]					
Minimum principal stiffness xz ratio		Maximum principal stiffness xz ratio			
min	13.800	min	2.170		
max	46.677	max	4.060		
med	28.836	med	1.305	homogeneity ratio	0.045

Table 6.9 Comparison of Cases c-C3 and s-C3. Ratios taken as c-C3/s-C3.

6.8 **Case s-C4.** Solitary Robot Operating Angularly in xy-Planes

Parameters	Configuration	Solitary
	Plane(s)	xy (z-levels)
	Euler XYZ angles	$\boldsymbol{\phi} = [10 \quad 5 \quad 20]^T \in \mathbb{R}^3$

		Description
Contents	Minima quiver-contour plot for Case s-C4	Static stiffness minima, minima directional field.
	Maxima quiver-contour plot for Case s-C4	Static stiffness maxima, maxima directional field.
	Results table for Case s-C4	Minima and maxima values, static stiffness homogeneity.

The directional field vectors in Figures 6.20 and 6.21 are uniformly scaled.

Stiffness homogeneity taken as the ratio between maximum principal stiffness and minimum principal stiffness medians.

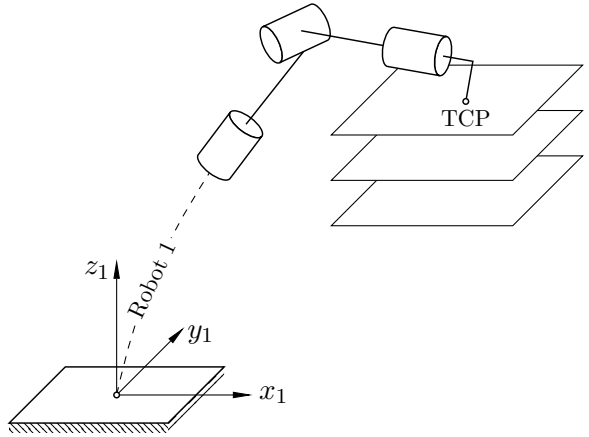


Figure 6.19 Illustration of Case s-C4. Not to scale.

All other simulation parameters remain the same to those disclosed in Tables 5.2 and 5.3.

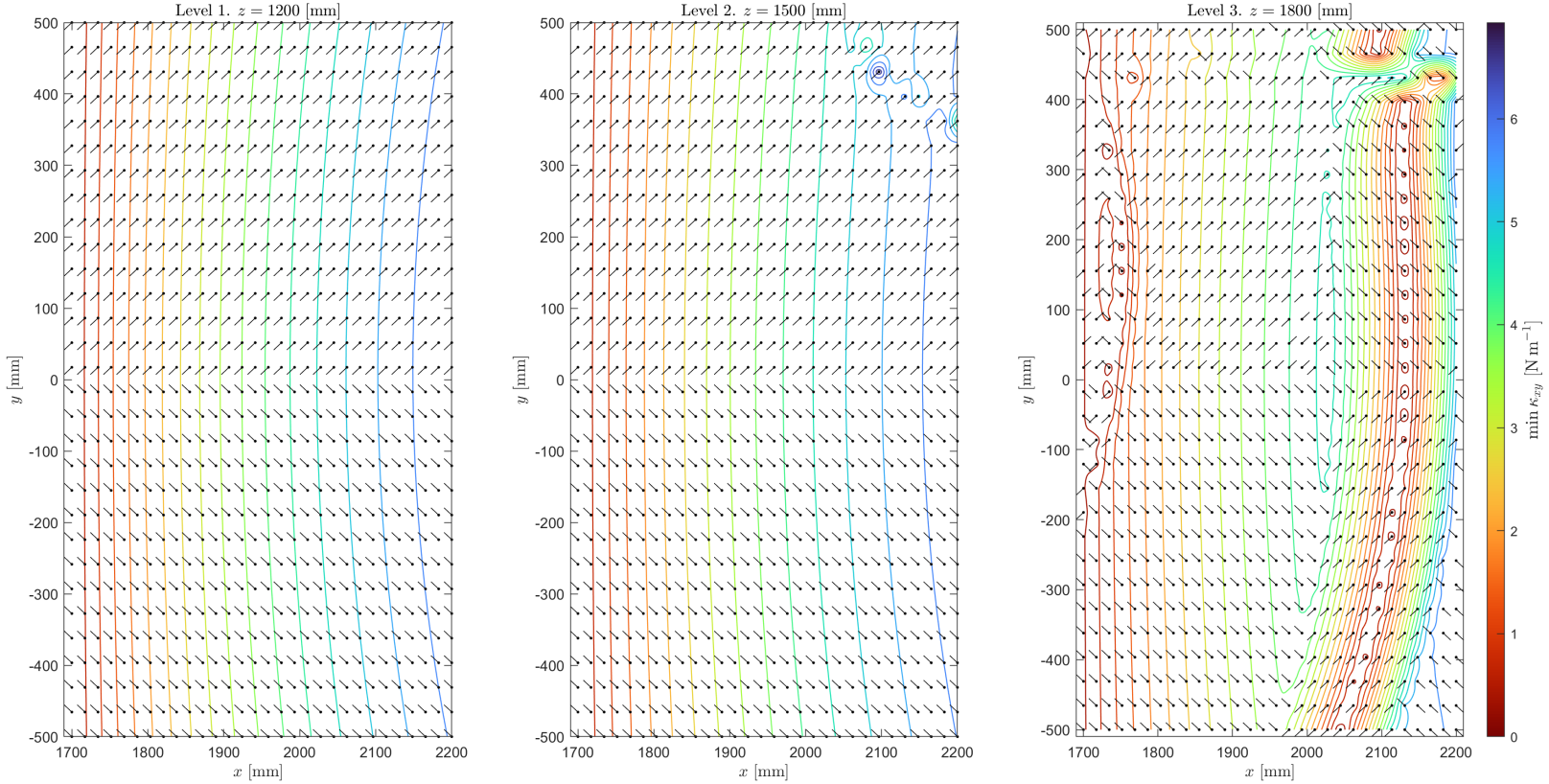


Figure 6.20 Minima quiver-contour plot for Case s-C4. Solitary robot, xy-planes, $\boldsymbol{\phi} = [10 \quad 5 \quad 20]^T \in \mathbb{R}^3$.

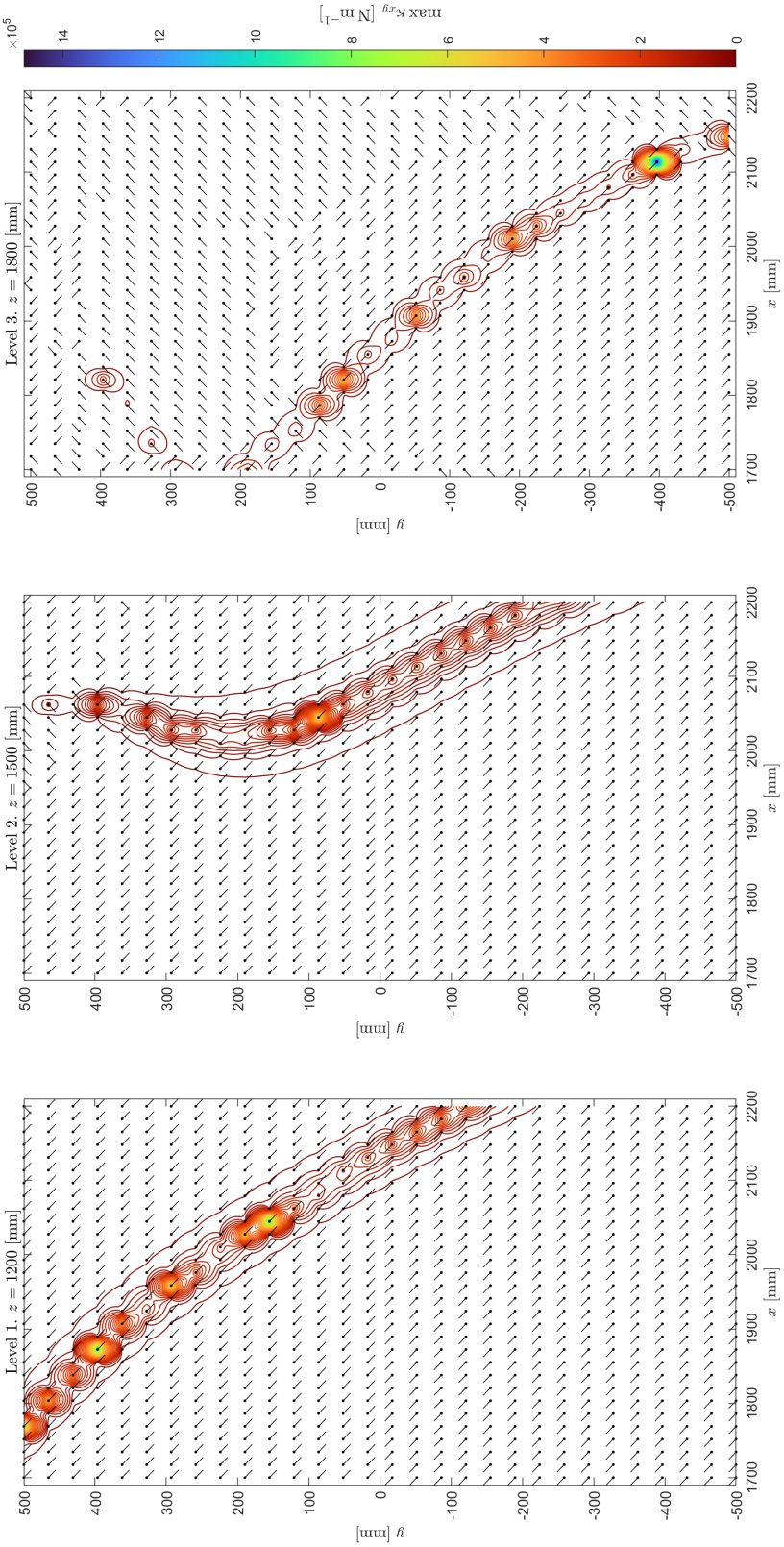


Figure 6.21 Maxima quiver-contour plot for Case s-C4. Solitary robot, xy-planes, $\boldsymbol{\phi} = [10 \quad 5 \quad 20]^T \in \mathbb{R}^3$.

Level 1. $z = 1\,200$ [mm]					
Minimum principal stiffness xy [$\mathrm{N\,m^{-1}}$]		Maximum principal stiffness xy [$\mathrm{N\,m^{-1}}$]			
min	0.317	min	329.225		
max	6.070	max	889 262.000		
med	3.900	med	5 423.870	homogeneity	1 391.890
Level 2. $z = 1\,500$ [mm]					
Minimum principal stiffness xy [$\mathrm{N\,m^{-1}}$]		Maximum principal stiffness xy [$\mathrm{N\,m^{-1}}$]			
min	0.317	min	142.890		
max	6.933	max	606 038.560		
med	3.890	med	3 841.590	homogeneity	987.770
Level 3. $z = 1\,800$ [mm]					
Minimum principal stiffness xy [$\mathrm{N\,m^{-1}}$]		Maximum principal stiffness xy [$\mathrm{N\,m^{-1}}$]			
min	0.001	min	17.085		
max	6.010	max	1 480 637.690		
med	2.853	med	4 881.520	homogeneity	1 711.100

Table 6.10 Results table for Case s-C4.

6.9 **Case s-C5.** Solitary Robot Operating Angularly in yz-Planes

Parameters	Configuration	Solitary
	Plane(s)	yz (x-levels)
	Euler XYZ angles	$\boldsymbol{\phi} = [10 \quad 5 \quad 20]^T \in \mathbb{R}^3$

		Description
Contents	Minima quiver-contour plot for Case s-C5	Static stiffness minima, minima directional field.
	Maxima quiver-contour plot for Case s-C5	Static stiffness maxima, maxima directional field.
	Results table for Case s-C5	Minima and maxima values, static stiffness homogeneity.

The directional field vectors in Figures 6.23 and 6.24 are uniformly scaled.

Stiffness homogeneity taken as the ratio between maximum principal stiffness and minimum principal stiffness medians.

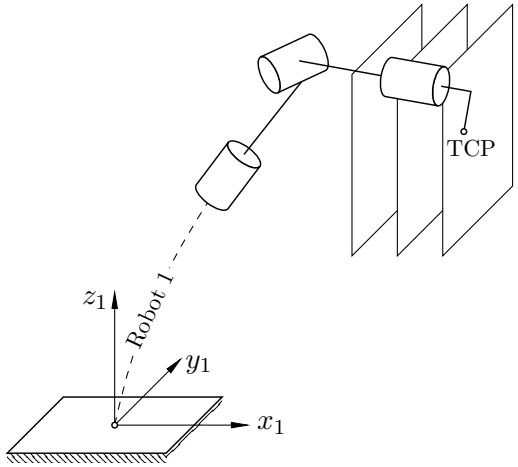


Figure 6.22 Illustration of Case s-C5. Not to scale.

All other simulation parameters remain the same to those disclosed in Tables 5.2 and 5.3.

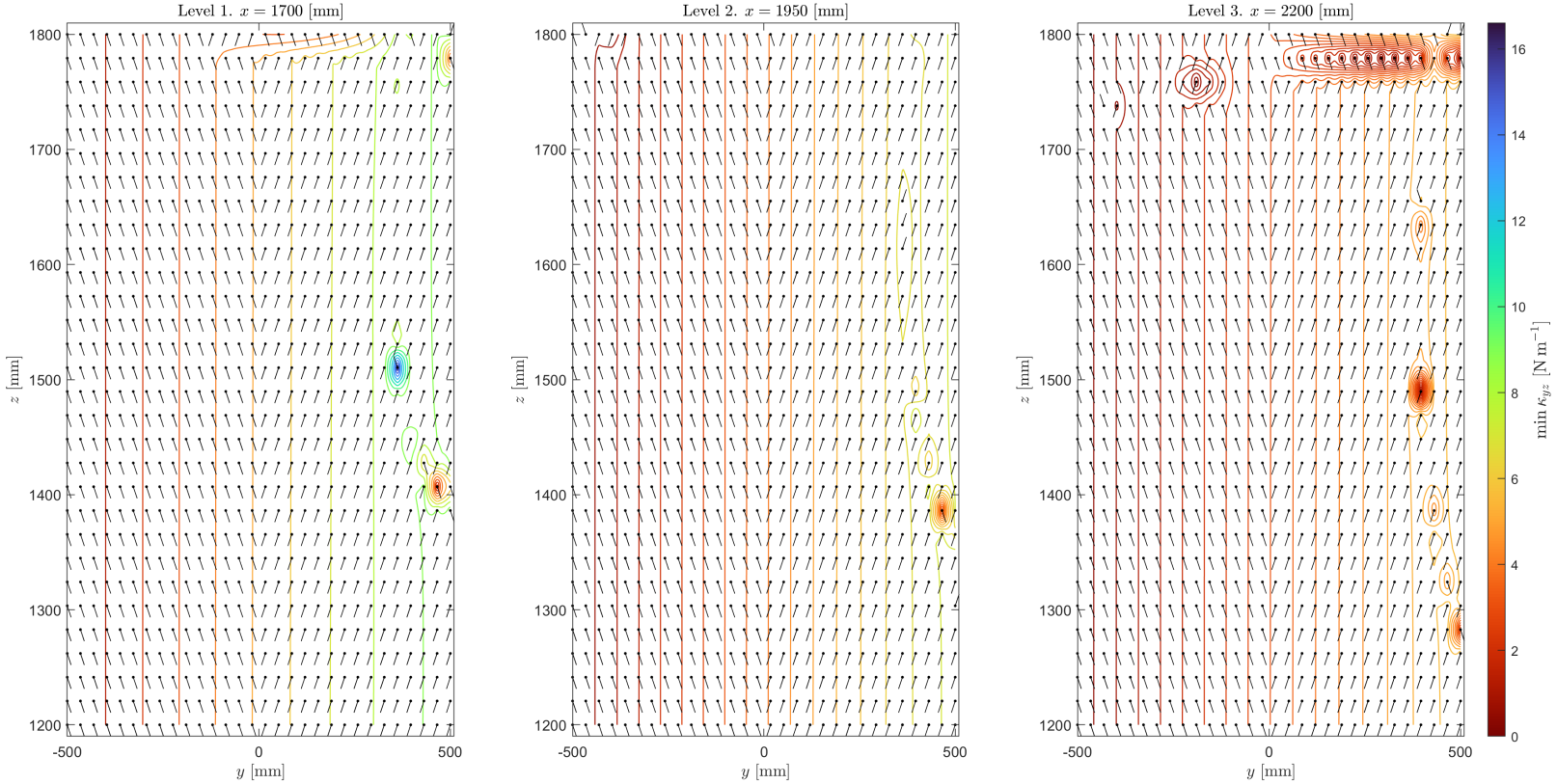


Figure 6.23 Minima quiver-contour plot for Case s-C5. Solitary robot, yz-planes, $\boldsymbol{\phi} = [10 \quad 5 \quad 20]^T \in \mathbb{R}^3$.

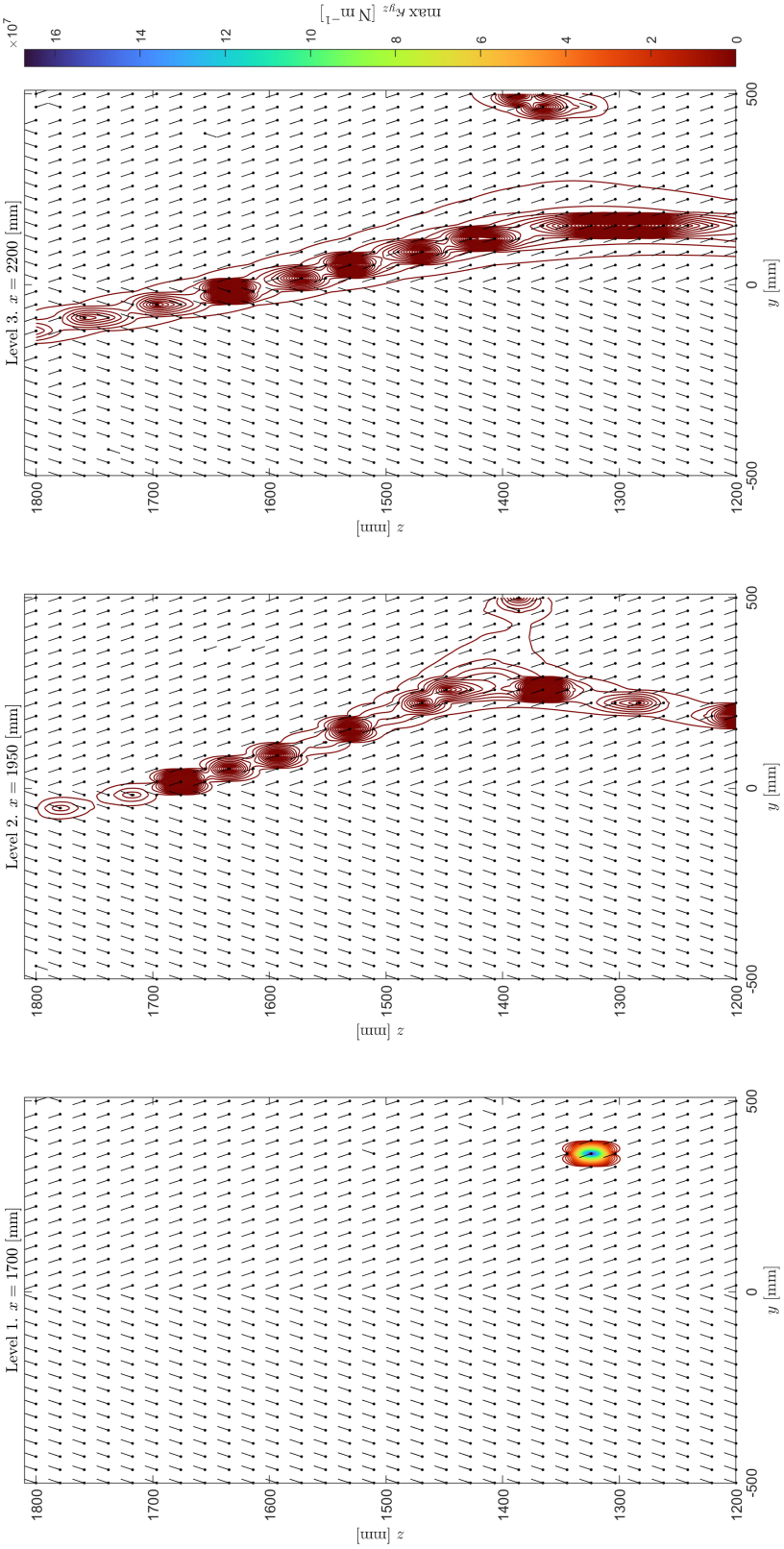


Figure 6.24 Maxima quiver-contour plot for Case s-C5. Solitary robot, yz-planes, $\boldsymbol{\phi} = [10 \quad 5 \quad 20]^T \in \mathbb{R}^3$.

Level 1. $x = 1\,700$ [mm]					
Minimum principal stiffness yz [$\mathrm{N\,m^{-1}}$]		Maximum principal stiffness yz [$\mathrm{N\,m^{-1}}$]			
min	0.317	min	39.640		
max	16.600	max	167 200 597.420		
med	5.045	med	3 289.190	homogeneity	651.750
Level 2. $x = 1\,950$ [mm]					
Minimum principal stiffness yz [$\mathrm{N\,m^{-1}}$]		Maximum principal stiffness yz [$\mathrm{N\,m^{-1}}$]			
min	0.246	min	98.836		
max	7.545	max	1 963 030.230		
med	3.867	med	3 866.395	homogeneity	1000.000
Level 3. $x = 2\,200$ [mm]					
Minimum principal stiffness yz [$\mathrm{N\,m^{-1}}$]		Maximum principal stiffness yz [$\mathrm{N\,m^{-1}}$]			
min	0.025	min	67.686		
max	5.900	max	589 019.037		
med	3.045	med	4 136.833	homogeneity	1 358.745

Table 6.11 Results table for Case s-C5.

6.10 **Case s-C6.** Solitary Robot Operating Angularly in xz-Planes

Parameters	Configuration	Solitary
	Plane(s)	xz (y-levels)
	Euler XYZ angles	$\boldsymbol{\phi} = [10 \quad 5 \quad 20]^T \in \mathbb{R}^3$

		Description
Contents	Minima quiver-contour plot for Case s-C6	Static stiffness minima, minima directional field.
	Maxima quiver-contour plot for Case s-C6	Static stiffness maxima, maxima directional field.
	Results table for Case s-C6	Minima and maxima values, static stiffness homogeneity.

The directional field vectors in Figures 6.26 and 6.27 are uniformly scaled.

Stiffness homogeneity taken as the ratio between maximum principal stiffness and minimum principal stiffness medians.

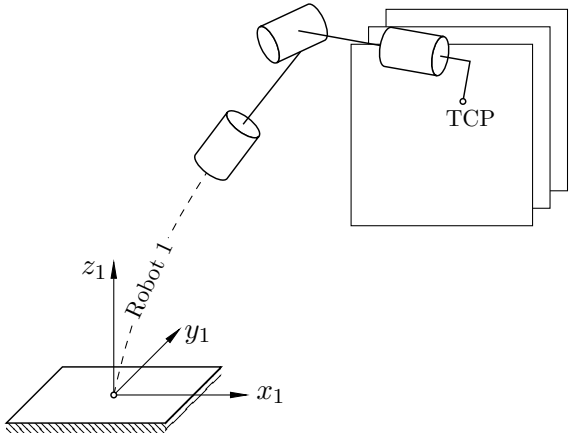


Figure 6.25 Illustration of Case s-C6. Not to scale.

All other simulation parameters remain the same to those disclosed in Tables 5.2 and 5.3.

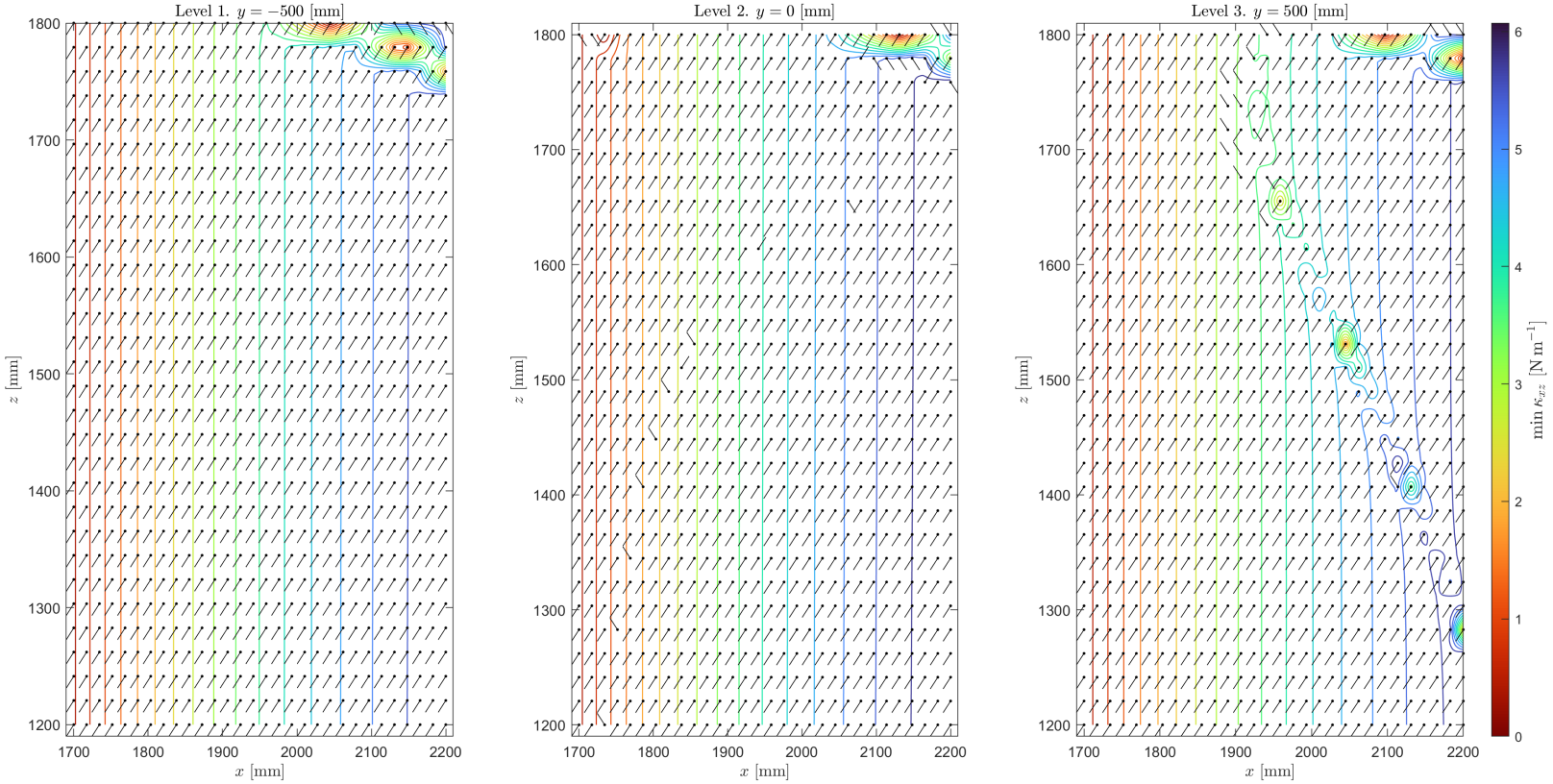


Figure 6.26 Minima quiver-contour plot for Case s-C6. Solitary robot, xz-planes, $\boldsymbol{\phi} = [10 \quad 5 \quad 20]^T \in \mathbb{R}^3$.

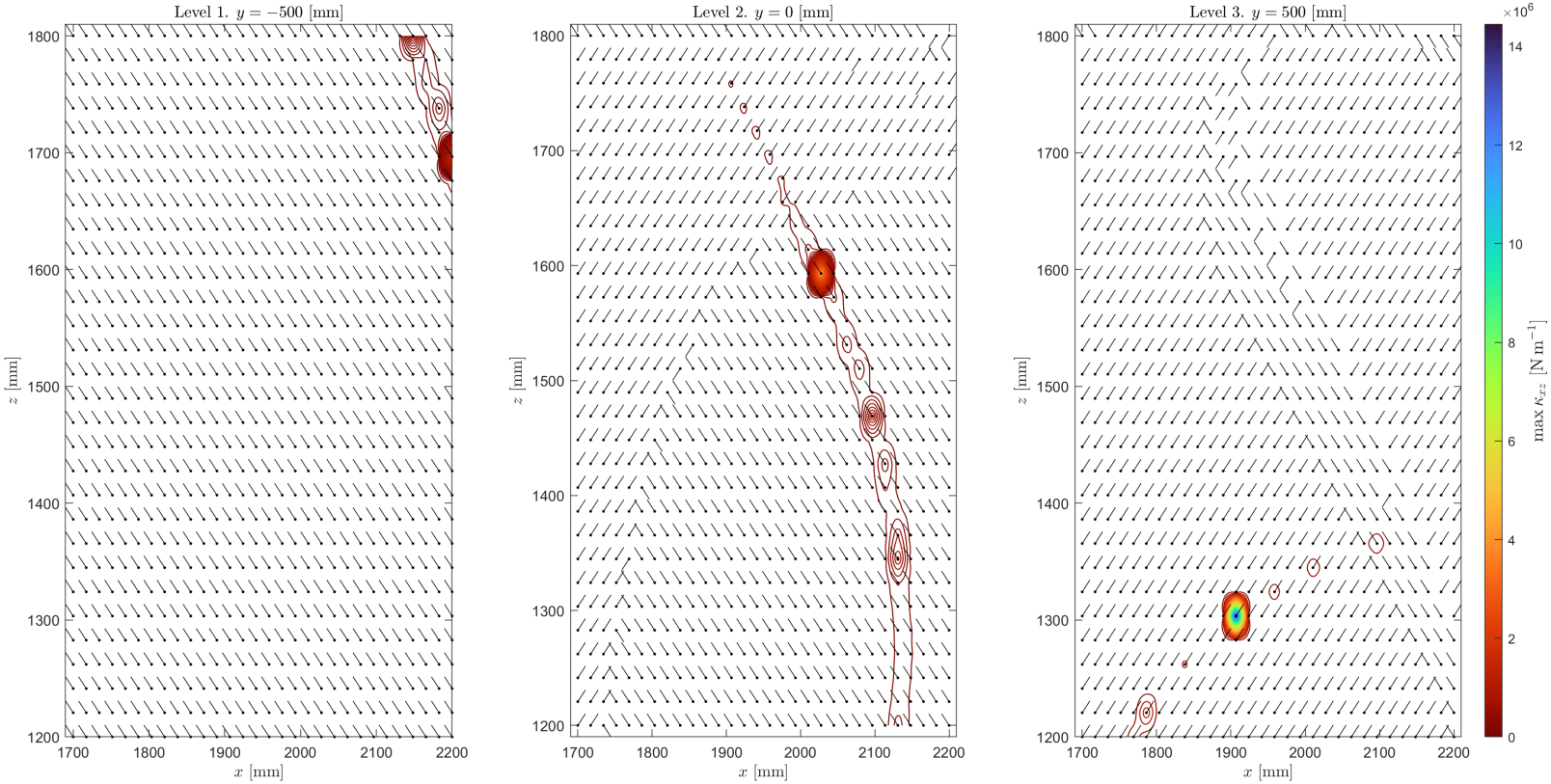


Figure 6.27 Minima quiver-contour plot for Case s-C6. Solitary robot, xz-planes, $\boldsymbol{\phi} = [10 \quad 5 \quad 20]^T \in \mathbb{R}^3$.

Level 1. $y = -500$ [mm]					
Minimum principal stiffness xz [$\mathrm{N\,m^{-1}}$]		Maximum principal stiffness xz [$\mathrm{N\,m^{-1}}$]			
min	0.001	min	329.225		
max	5.771	max	1 575 544.430		
med	3.657	med	1 596.450	homogeneity	436.530
Level 2. $y = 0$ [mm]					
Minimum principal stiffness xz [$\mathrm{N\,m^{-1}}$]		Maximum principal stiffness xz [$\mathrm{N\,m^{-1}}$]			
min	0.075	min	883.170		
max	6.073	max	4 122 020.575		
med	3.895	med	6 972.120	homogeneity	1 789.615
Level 3. $y = 500$ [mm]					
Minimum principal stiffness xz [$\mathrm{N\,m^{-1}}$]		Maximum principal stiffness xz [$\mathrm{N\,m^{-1}}$]			
min	0.173	min	17.085		
max	6.015	max	14 450 212.625		
med	3.670	med	4 039.083	homogeneity	1 100.923

Table 6.12 Results table for Case s-C6.

6.11 **Case c-C4.** Coupled Robots Operating Angularly in xy-Planes

Parameters	Configuration	Coupled
	Plane(s)	xy (z-levels)
	Euler XYZ angles	$\boldsymbol{\phi} = [10 \quad 5 \quad 20]^T \in \mathbb{R}^3$

Contents		*Description*
	Minima quiver-contour plot for Case c-C4	Static stiffness minima, minima directional field.
	Maxima quiver-contour plot for Case c-C4	Static stiffness maxima, maxima directional field.
	Results table for Case c-C4	Minima and maxima values, static stiffness homogeneity.

The directional field vectors in Figures 6.29 and 6.30 are uniformly scaled.

Stiffness homogeneity taken as the ratio between maximum principal stiffness and minimum principal stiffness medians.

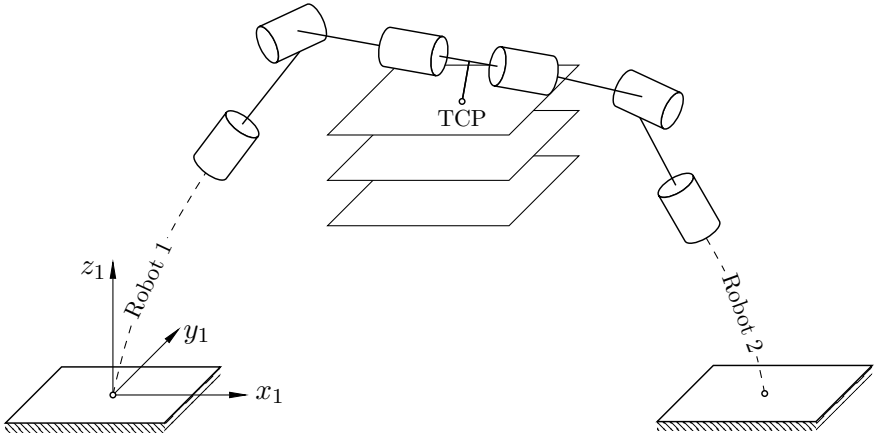


Figure 6.28 Illustration of Case c-C4. Not to scale.

All other simulation parameters remain the same to those disclosed in Tables 5.2 and 5.3.

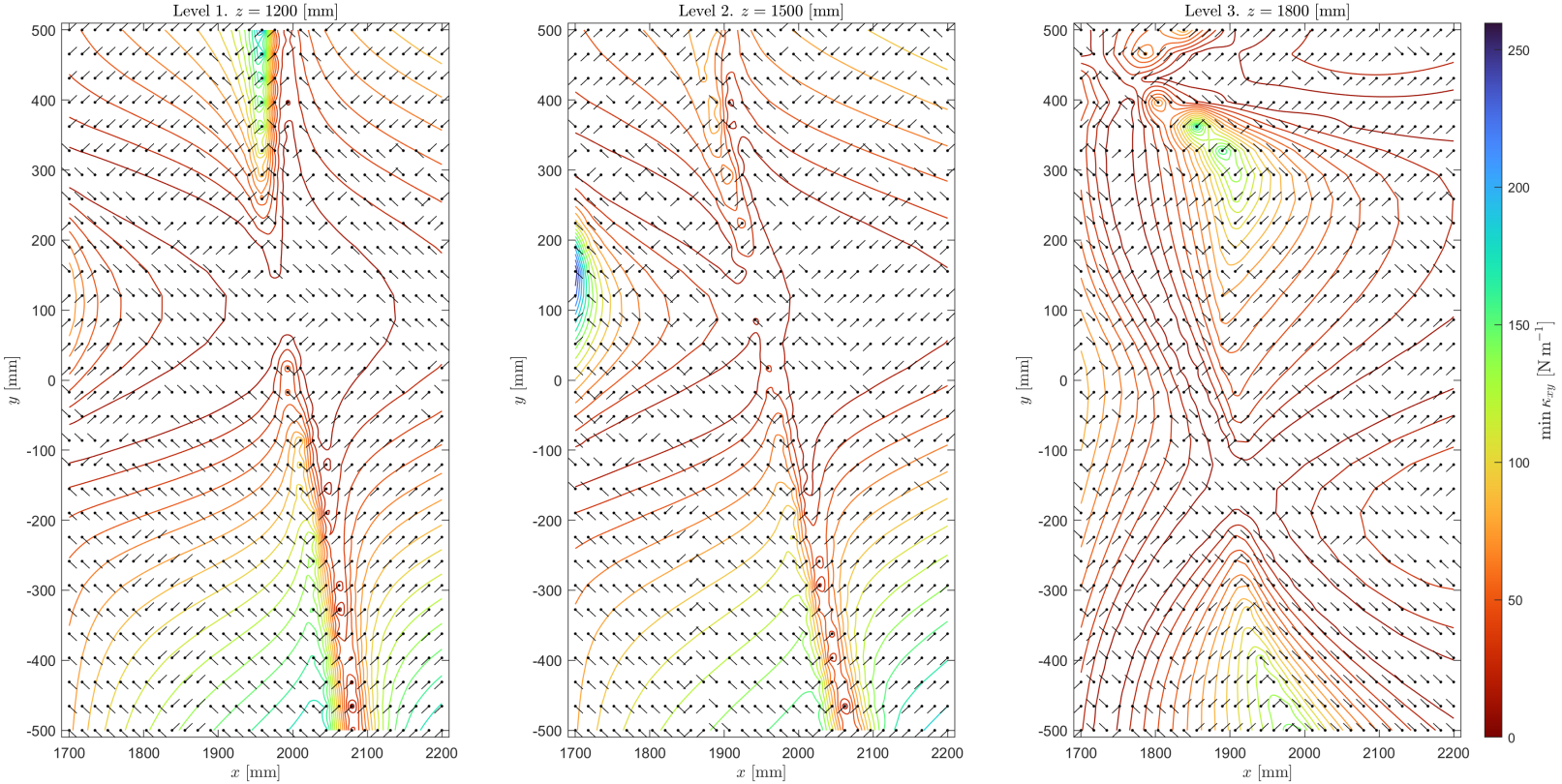


Figure 6.29 Minima quiver-contour plot for Case c-C4. Coupled robots, xy-planes, $\boldsymbol{\phi} = [10 \quad 5 \quad 20]^T \in \mathbb{R}^3$.

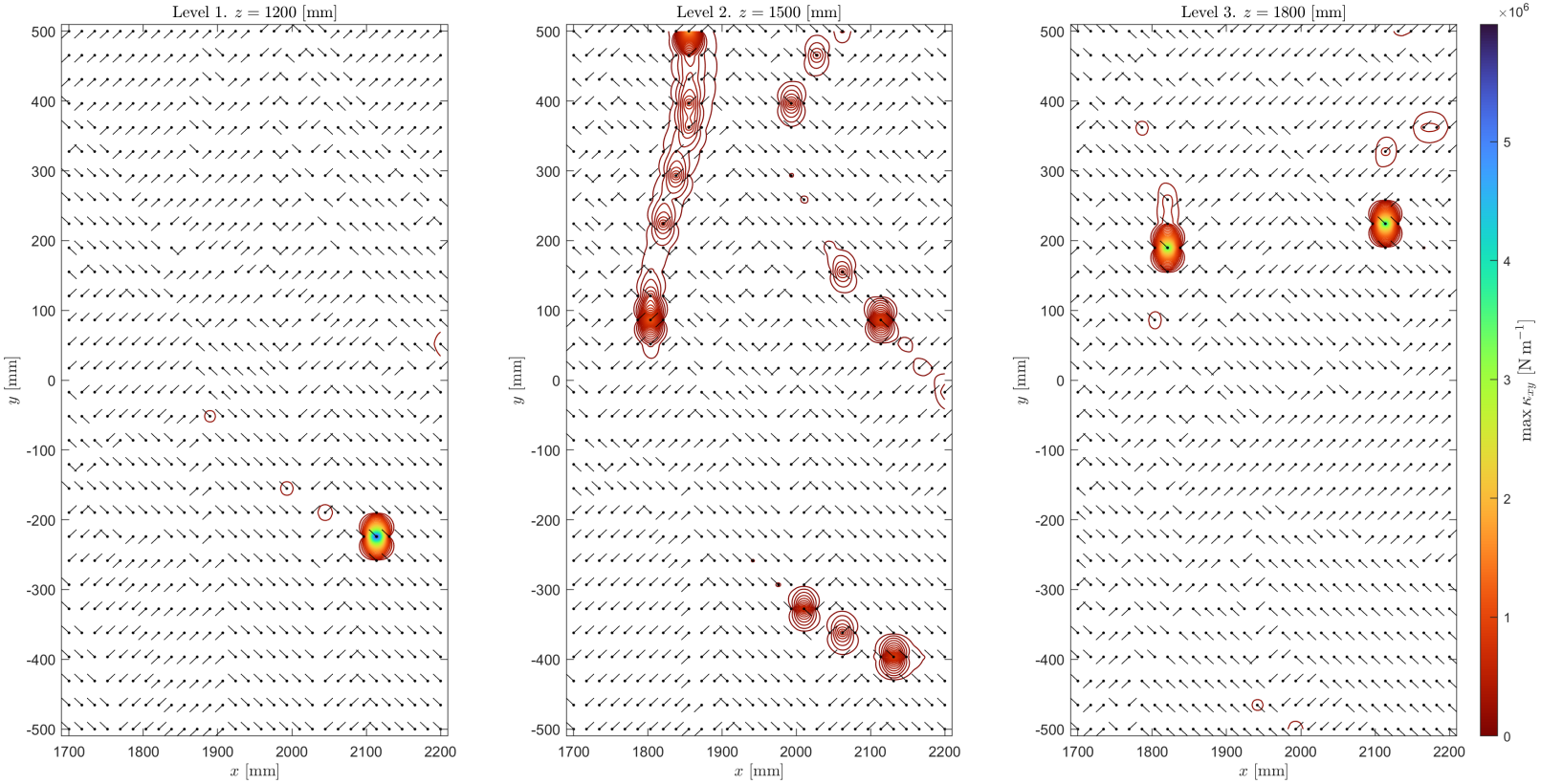


Figure 6.30 Maxima quiver-contour plot for Case c-C4. Coupled robots, xy-planes, $\boldsymbol{\phi} = [10 \quad 5 \quad 20]^T \in \mathbb{R}^3$.

Level 1. $z = 1\,200$ [mm]					
Minimum principal stiffness xy [$\mathrm{N\,m^{-1}}$]		Maximum principal stiffness xy [$\mathrm{N\,m^{-1}}$]			
min	0.020	min	1 040.222		
max	211.525	max	5 987 573.784		
med	44.692	med	4 740.550	homogeneity	106.075
Level 2. $z = 1\,500$ [mm]					
Minimum principal stiffness xy [$\mathrm{N\,m^{-1}}$]		Maximum principal stiffness xy [$\mathrm{N\,m^{-1}}$]			
min	0.023	min	749.455		
max	259.700	max	1 917 246.230		
med	50.236	med	6 529.990	homogeneity	129.985
Level 3. $z = 1\,800$ [mm]					
Minimum principal stiffness xy [$\mathrm{N\,m^{-1}}$]		Maximum principal stiffness xy [$\mathrm{N\,m^{-1}}$]			
min	0.008	min	632.231		
max	176.567	max	4 145 477.705		
med	27.165	med	3 953.360	homogeneity	145.532

Table 6.13 Results table for Case c-C4.

6.12 **Case c-C5.** Coupled Robots Operating Angularly in yz-Planes

Parameters	Configuration	Coupled
	Plane(s)	yz (x-levels)
	Euler XYZ angles	$\boldsymbol{\phi} = [10 \quad 5 \quad 20]^T \in \mathbb{R}^3$

		Description
Contents	Minima quiver-contour plot for Case c-C5	Static stiffness minima, minima directional field.
	Maxima quiver-contour plot for Case c-C5	Static stiffness maxima, maxima directional field.
	Results table for Case c-C5	Minima and maxima values, static stiffness homogeneity.

The directional field vectors in Figures 6.32 and 6.33 are uniformly scaled.

Stiffness homogeneity taken as the ratio between maximum principal stiffness and minimum principal stiffness medians.

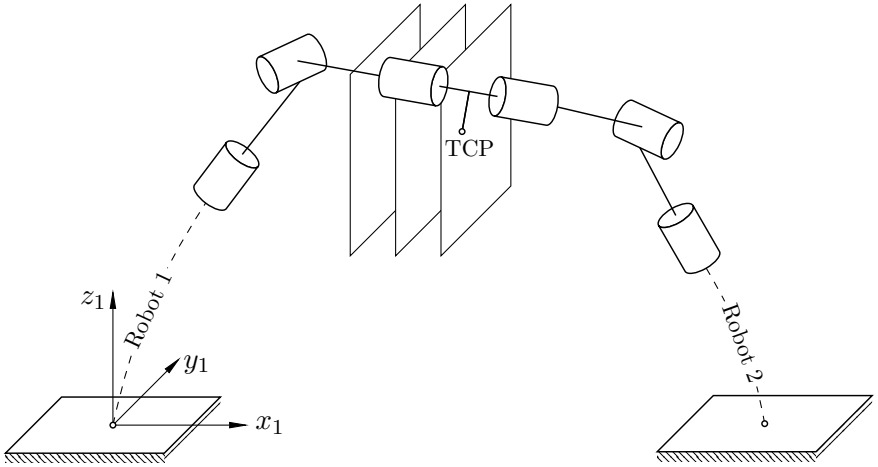


Figure 6.31 Illustration of Case c-C5. Not to scale.

All other simulation parameters remain the same to those disclosed in Tables 5.2 and 5.3.

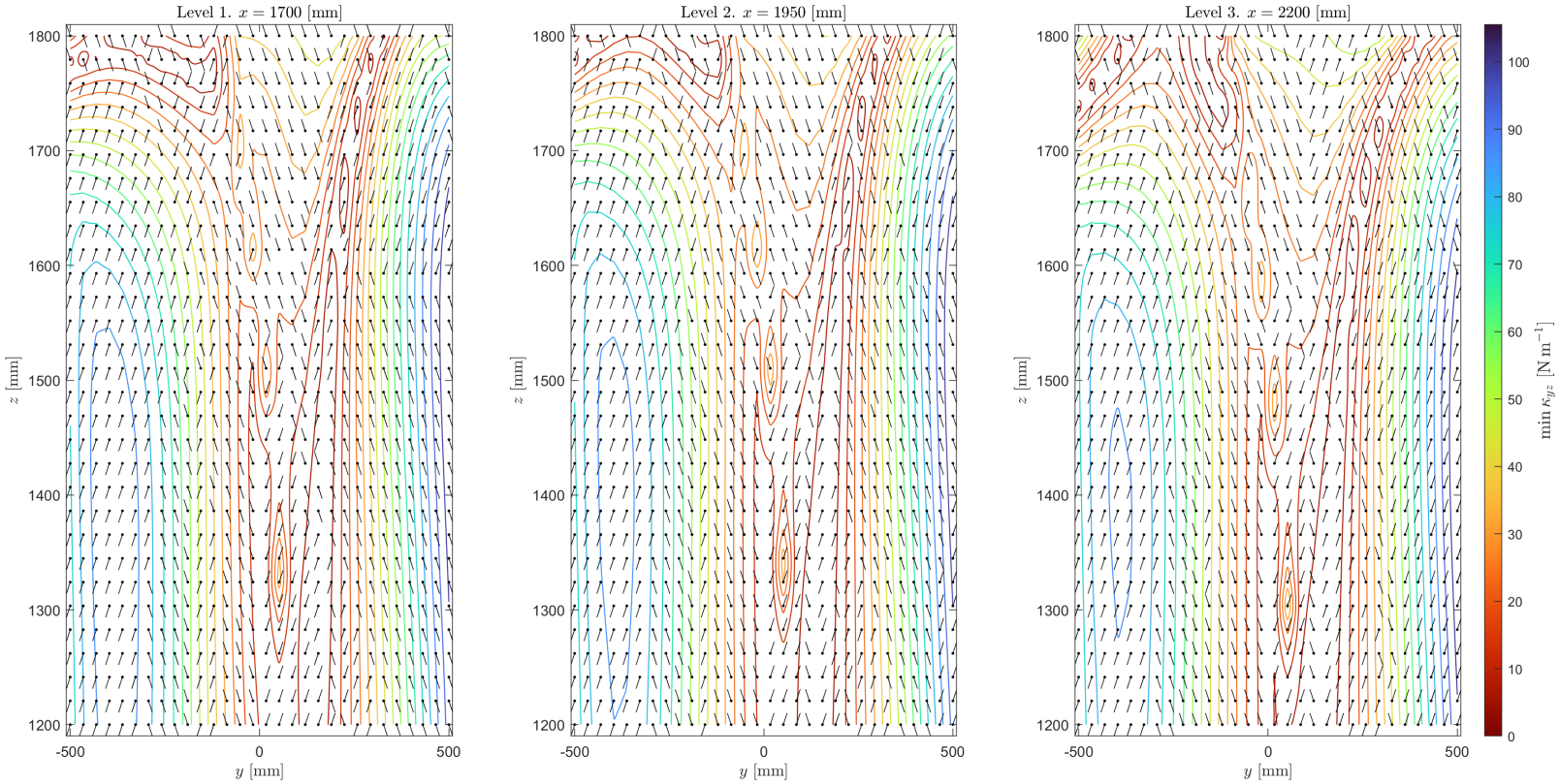


Figure 6.32 Minima quiver-contour plot for Case c-C5. Coupled robots, yz-planes, $\boldsymbol{\phi} = [10 \quad 5 \quad 20]^T \in \mathbb{R}^3$.

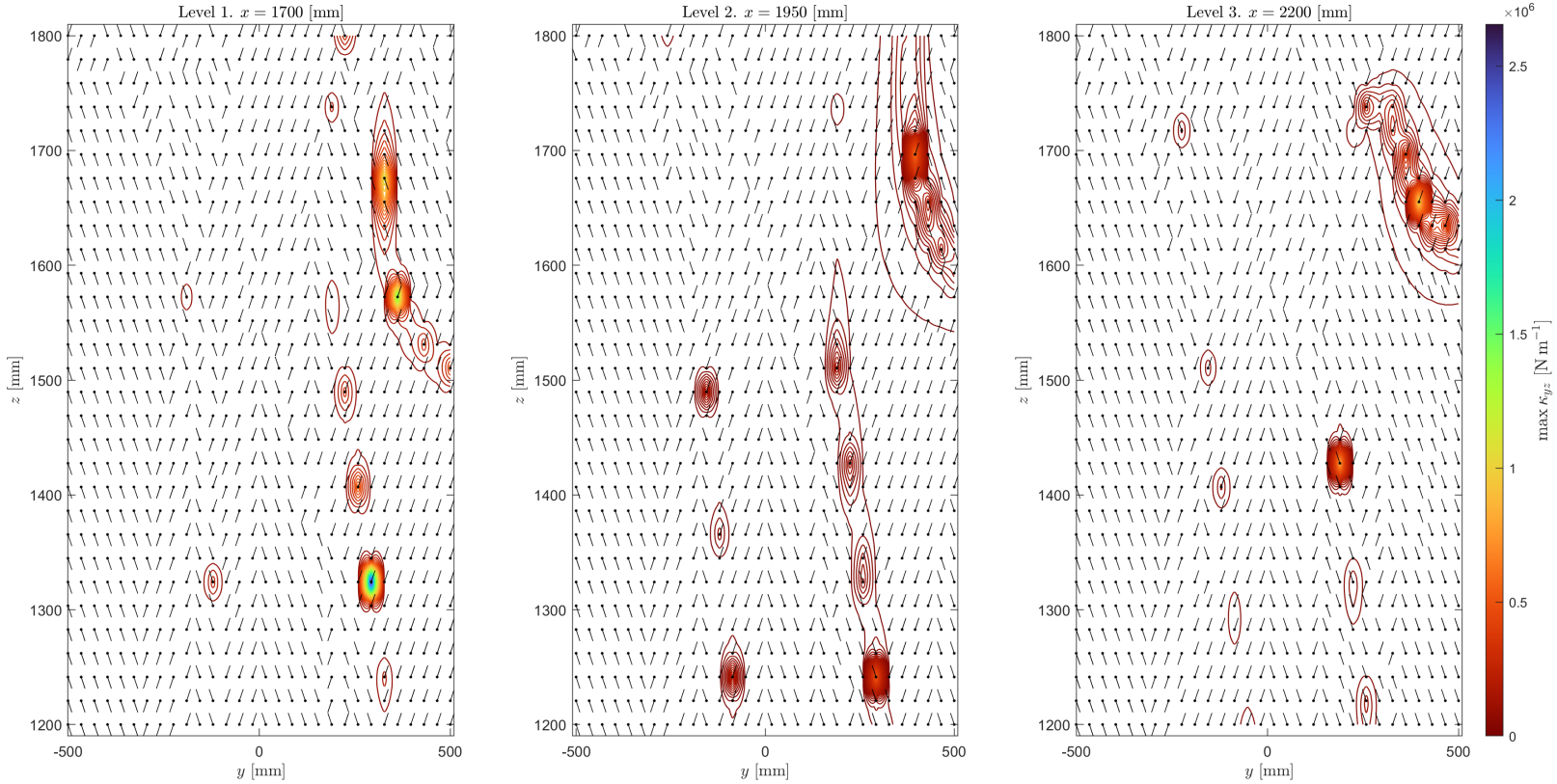


Figure 6.33 Maxima quiver-contour plot for Case c-C5. Coupled robots, yz-planes, $\boldsymbol{\phi} = [10 \quad 5 \quad 20]^T \in \mathbb{R}^3$.

Level 1. $x = 1\,700$ [mm]					
Minimum principal stiffness yz [$\mathrm{N\,m^{-1}}$]		Maximum principal stiffness yz [$\mathrm{N\,m^{-1}}$]			
min	0.027	min	1 700.517		
max	104.935	max	2 656 238.510		
med	40.625	med	3 398.753	homogeneity	83.660
Level 2. $x = 1\,950$ [mm]					
Minimum principal stiffness yz [$\mathrm{N\,m^{-1}}$]		Maximum principal stiffness yz [$\mathrm{N\,m^{-1}}$]			
min	0.225	min	1 705.020		
max	103.953	max	592 342.345		
med	40.380	med	3 938.157	homogeneity	97.527
Level 3. $x = 2\,200$ [mm]					
Minimum principal stiffness yz [$\mathrm{N\,m^{-1}}$]		Maximum principal stiffness yz [$\mathrm{N\,m^{-1}}$]			
min	0.010	min	1 702.850		
max	105.560	max	1 000 740.182		
med	39.185	med	5 019.235	homogeneity	128.090

Table 6.14 Results table for Case c-C5.

6.13 **Case c-C6.** Coupled Robots Operating Angularly in xz-Planes

Parameters	Configuration	Coupled
	Plane(s)	xz (y-levels)
	Euler XYZ angles	$\boldsymbol{\phi} = [10 \quad 5 \quad 20]^T \in \mathbb{R}^3$

		Description
Contents	Minima quiver-contour plot for Case c-C6	Static stiffness minima, minima directional field.
	Maxima quiver-contour plot for Case c-C6	Static stiffness maxima, maxima directional field.
	Results table for Case c-C6	Minima and maxima values, static stiffness homogeneity.

The directional field vectors in Figures 6.35 and 6.36 are uniformly scaled.

Stiffness homogeneity taken as the ratio between maximum principal stiffness and minimum principal stiffness medians.

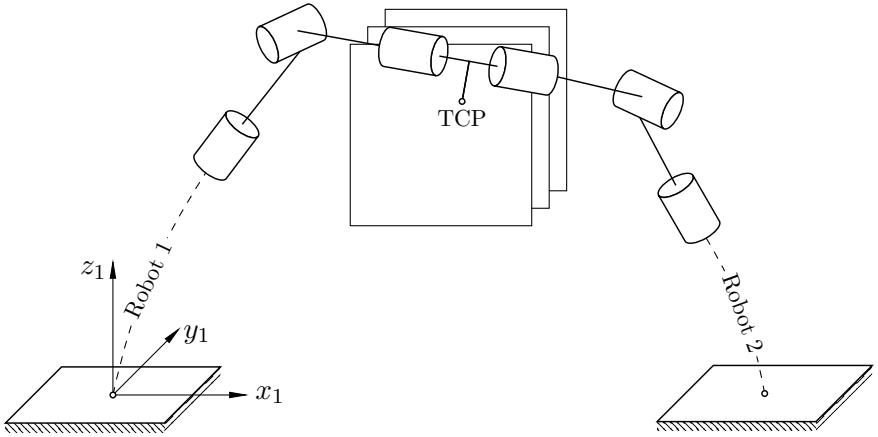


Figure 6.34 Illustration of Case c-C6. Not to scale.

All other simulation parameters remain the same to those disclosed in Tables 5.2 and 5.3.

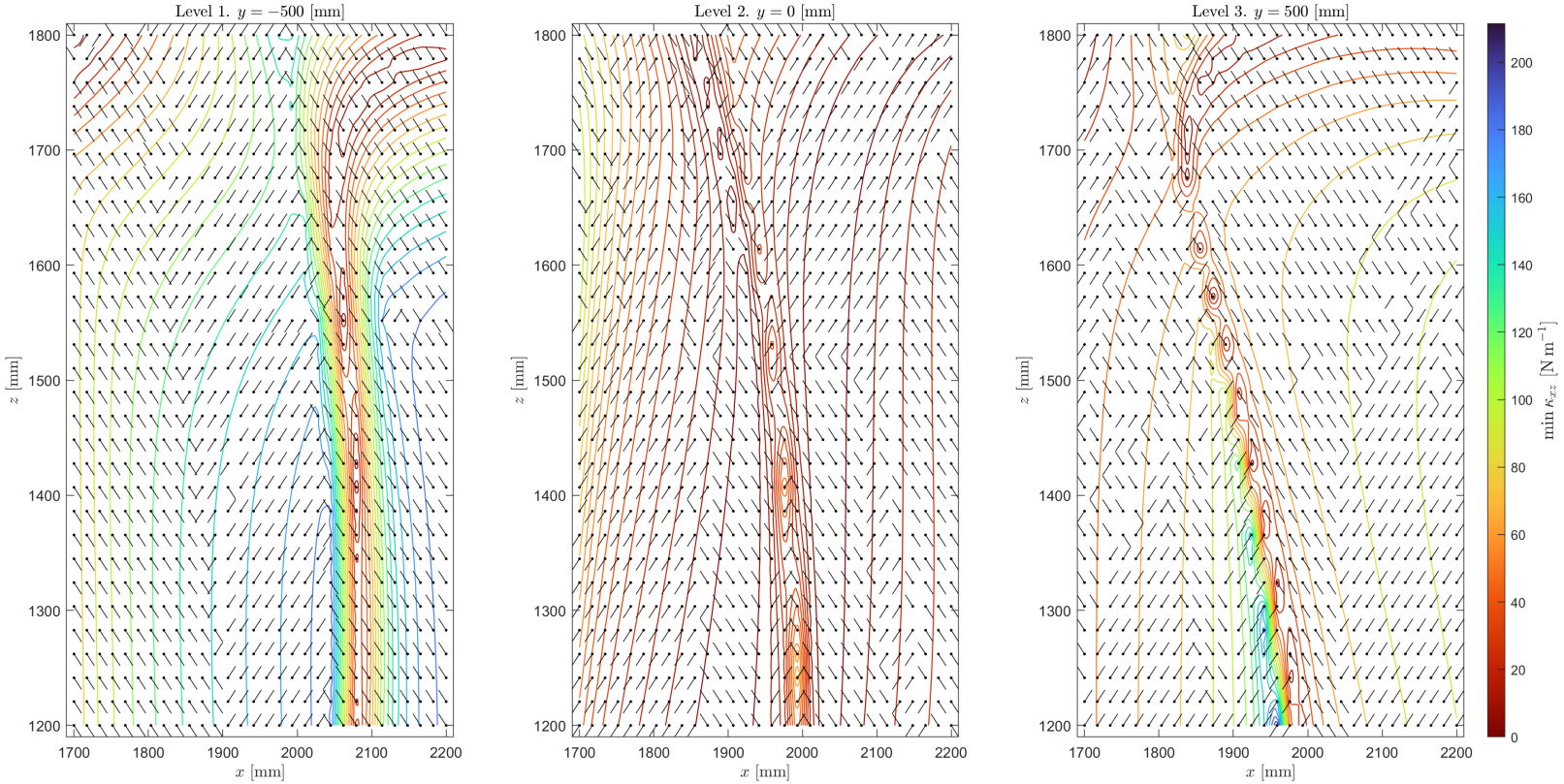


Figure 6.35 Minima quiver-contour plot for Case c-C6. Coupled robots, xz-planes, $\boldsymbol{\phi} = [10 \quad 5 \quad 20]^T \in \mathbb{R}^3$.

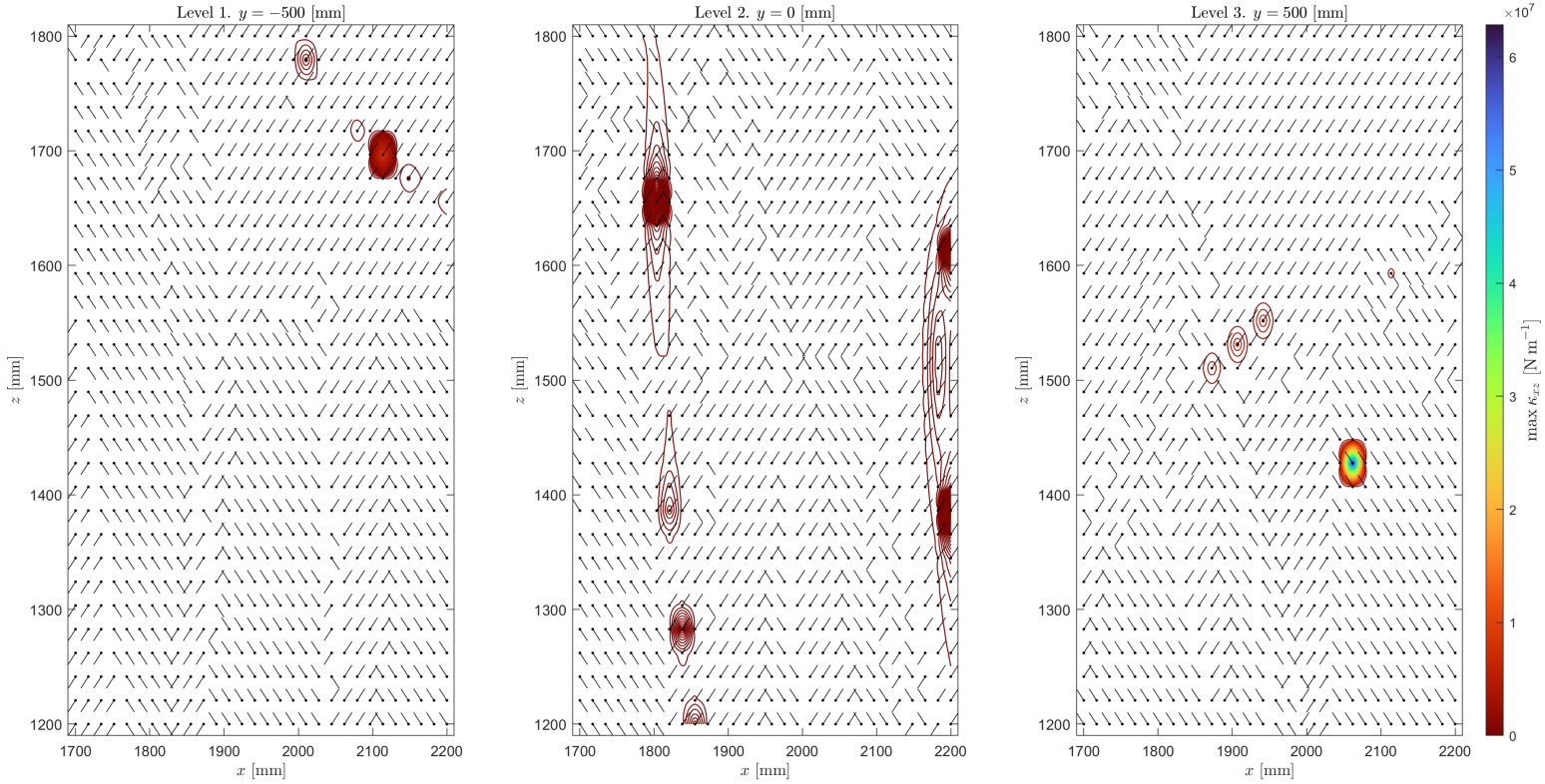


Figure 6.36 Maxima quiver-contour plot for Case c-C6. Coupled robots, xz-planes, $\boldsymbol{\phi} = [10 \quad 5 \quad 20]^T \in \mathbb{R}^3$.

Level 1. $y = -500$ [mm]					
Minimum principal stiffness xz [$\mathrm{N\,m^{-1}}$]		Maximum principal stiffness xz [$\mathrm{N\,m^{-1}}$]			
min	0.702	min	997.550		
max	190.440	max	8 429 178.556		
med	120.777	med	2 182.650	homogeneity	18.070
Level 2. $y = 0$ [mm]					
Minimum principal stiffness xz [$\mathrm{N\,m^{-1}}$]		Maximum principal stiffness xz [$\mathrm{N\,m^{-1}}$]			
min	0.020	min	688.030		
max	101.850	max	699 061.930		
med	17.903	med	3 778.895	homogeneity	211.080
Level 3. $y = 500$ [mm]					
Minimum principal stiffness xz [$\mathrm{N\,m^{-1}}$]		Maximum principal stiffness xz [$\mathrm{N\,m^{-1}}$]			
min	0.262	min	1 027.590		
max	211.524	max	62 860 479.640		
med	68.222	med	12 074.970	homogeneity	176.995

Table 6.15 Results table for Case c-C6.

6.14 Comparison of Cases s-C4-6 and c-C4-6

		Description
Contents	Comparison of Cases c-C4 and s-C4	Ratios between values for Case c-C4 and Case s-C4
	Comparison of Cases c-C5 and s-C5	Ratios between values for Case c-C5 and Case s-C5
	Comparison of Cases c-C6 and s-C6	Ratios between values for Case c-C6 and Case s-C6
Legend	<number>	Improvement after coupling.
	<number>	Deterioration after coupling.
	<number>	No change after coupling.

- **Improvement After Coupling.** As for <number> static stiffness values, larger ratio means greater improvement, i.e., the coupled configuration exhibits <number>-times larger the respective value. As for <number> stiffness homogeneities, smaller ratio means greater improvement, i.e., the coupled configuration's stiffness homogeneity is <number>-times the stiffness homogeneity of a solitary robot, resulting in the coupled configuration showing an overall increase in static stiffness homogeneity by a factor of 1/<number> over the solitary configuration.
- **Deterioration After Coupling.** As for <number> static stiffness values, smaller ratio means greater deterioration, i.e., the coupled configuration exhibits <number>-times smaller the respective value. As for <number> stiffness homogeneities, larger ratio means greater deterioration, i.e., the coupled configuration's stiffness homogeneity is <number>-times the stiffness homogeneity of a solitary robot, resulting in the coupled configuration showing an overall decrease in static stiffness homogeneity by a factor of 1/<number> over the solitary configuration.

Level 1. $z = 1\,200$ [mm]					
Minimum principal stiffness xy ratio		Maximum principal stiffness xy ratio			
min	0.060	min	3.160		
max	34.831	max	6.733		
med	11.470	med	0.874	homogeneity ratio	0.076
Level 2. $z = 1\,500$ [mm]					
Minimum principal stiffness xy ratio		Maximum principal stiffness xy ratio			
min	0.073	min	5.245		
max	37.457	max	3.160		
med	12.937	med	1.700	homogeneity ratio	0.132
Level 3. $z = 1\,800$ [mm]					
Minimum principal stiffness xy ratio		Maximum principal stiffness xy ratio			
min	29.832	min	37.000		
max	29.370	max	2.800		
med	9.522	med	0.810	homogeneity ratio	0.085

Table 6.16 Comparison of Cases c-C4 and s-C4. Ratios taken as c-C4/s-C4.

Level 1. $x = 1\,700$ [mm]					
Minimum principal stiffness yz ratio		Maximum principal stiffness yz ratio			
min	0.088	min	42.900		
max	6.320	max	0.015		
med	8.050	med	1.033	homogeneity ratio	0.128
Level 2. $x = 1\,950$ [mm]					
Minimum principal stiffness yz ratio		Maximum principal stiffness yz ratio			
min	0.910	min	17.250		
max	13.777	max	5.240		
med	10.445	med	0.302	homogeneity ratio	0.100
Level 3. $x = 2\,200$ [mm]					
Minimum principal stiffness yz ratio		Maximum principal stiffness yz ratio			
min	0.420	min	25.158		
max	17.890	max	1.700		
med	12.870	med	1.213	homogeneity ratio	0.095

Table 6.17 Comparison of Cases c-C5 and s-C5. Ratios taken as c-C5/s-C5.

Level 1. $y = -500$ [mm]					
Minimum principal stiffness xz ratio		Maximum principal stiffness xz ratio			
min	2 363.865	min	3.030		
max	33.000	max	5.350		
med	32.845	med	1.367	homogeneity ratio	0.041
Level 2. $y = 0$ [mm]					
Minimum principal stiffness xz ratio		Maximum principal stiffness xz ratio			
min	0.248	min	0.780		
max	16.773	max	0.170		
med	4.595	med	0.542	homogeneity ratio	0.118
Level 3. $y = 500$ [mm]					
Minimum principal stiffness xz ratio		Maximum principal stiffness xz ratio			
min	1.510	min	60.146		
max	35.166	max	4.350		
med	18.595	med	2.990	homogeneity ratio	0.160

Table 6.18 Comparison of Cases c-C6 and s-C6. Ratios taken as c-C6/s-C6.

6.15 Comparison of Cases s-C1-3 and s-C4-6

		Description
Contents	Comparison of Cases s-C4 and s-C1	Ratios between values for Case s-C4 and Case s-C1
	Comparison of Cases s-C5 and s-C2	Ratios between values for Case s-C5 and Case s-C2
	Comparison of Cases s-C6 and s-C3	Ratios between values for Case s-C6 and Case s-C3
Legend	<number>	Improvement after reorienting.
	<number>	Deterioration after reorienting.
	<number>	No change after reorienting.

- **Improvement After Reorienting.** As for <number> static stiffness values, larger ratio means greater improvement, i.e., the angled configuration exhibits <number>-times larger the respective value. As for <number> stiffness homogeneities, smaller ratio means greater improvement, i.e., the angled configuration's stiffness homogeneity is <number>-times the stiffness homogeneity of the vertical configuration, resulting in the angled configuration showing an overall increase in static stiffness homogeneity by a factor of 1/<number> over the vertical configuration.
- **Deterioration After Reorienting.** As for <number> static stiffness values, smaller ratio means greater deterioration, i.e., the angled configuration exhibits <number>-times smaller the respective value. As for <number> stiffness homogeneities, larger ratio means greater deterioration, i.e., the angled configuration's stiffness homogeneity is <number>-times the stiffness homogeneity of the vertical configuration, resulting in the angled configuration showing an overall decrease in static stiffness homogeneity by a factor of 1/<number> over the vertical configuration.

Level 1. $z = 1\,200$ [mm]					
Minimum principal stiffness xy ratio		Maximum principal stiffness xy ratio			
min	1.000	min	0.113		
max	1.000	max	1.477		
med	1.000	med	1.099	homogeneity ratio	1.100
Level 2. $z = 1\,500$ [mm]					
Minimum principal stiffness xy ratio		Maximum principal stiffness xy ratio			
min	1.000	min	0.195		
max	1.112	max	0.242		
med	1.000	med	0.414	homogeneity ratio	0.414
Level 3. $z = 1\,800$ [mm]					
Minimum principal stiffness xy ratio		Maximum principal stiffness xy ratio			
min	0.341	min	1.582		
max	0.992	max	1.186		
med	0.799	med	1.000	homogeneity ratio	1.312

Table 6.19 Comparison of Cases s-C4 and s-C1. Ratios taken as s-C4/s-C1.

Level 1. $x = 1\,700$ [mm]					
Minimum principal stiffness yz ratio		Maximum principal stiffness yz ratio			
min	2.663	min	0.501		
max	1.781	max	7.901		
med	1.000	med	0.380	homogeneity ratio	0.383
Level 2. $x = 1\,950$ [mm]					
Minimum principal stiffness yz ratio		Maximum principal stiffness yz ratio			
min	1.000	min	0.820		
max	1.000	max	1.466		
med	0.968	med	0.568	homogeneity ratio	0.587
Level 3. $x = 2\,200$ [mm]					
Minimum principal stiffness yz ratio		Maximum principal stiffness yz ratio			
min	0.122	min	2.512		
max	1.000	max	1.177		
med	1.000	med	0.775	homogeneity ratio	0.776

Table 6.20 Comparison of Cases s-C5 and s-C2. Ratios taken as s-C5/s-C2.

Level 1. $y = -500$ [mm]					
Minimum principal stiffness xz ratio		Maximum principal stiffness xz ratio			
min	0.008	min	0.249		
max	1.000	max	0.186		
med	1.000	med	0.207	homogeneity ratio	0.207
Level 2. $y = 0$ [mm]					
Minimum principal stiffness xz ratio		Maximum principal stiffness xz ratio			
min	1.161	min	10.063		
max	0.863	max	3.626		
med	1.000	med	1.560	homogeneity ratio	1.562
Level 3. $y = 500$ [mm]					
Minimum principal stiffness xz ratio		Maximum principal stiffness xz ratio			
min	0.548	min	0.013		
max	1.000	max	3.357		
med	1.000	med	0.537	homogeneity ratio	0.535

Table 6.21 Comparison of Cases s-C6 and s-C3. Ratios taken as s-C6/s-C3.

6.16 Comparison of Cases c-C1-3 and c-C4-6

		Description
Contents	Comparison of Cases c-C4 and c-C1	Ratios between values for Case c-C4 and Case c-C1
	Comparison of Cases c-C5 and c-C2	Ratios between values for Case c-C5 and Case c-C2
	Comparison of Cases c-C6 and c-C3	Ratios between values for Case c-C6 and Case c-C3
Legend	<number>	Improvement after reorienting.
	<number>	Deterioration after reorienting.
	<number>	No change after reorienting.

- **Improvement After Reorienting.** As for <number> static stiffness values, larger ratio means greater improvement, i.e., the angled configuration exhibits <number>-times larger the respective value. As for <number> stiffness homogeneities, smaller ratio means greater improvement, i.e., the angled configuration's stiffness homogeneity is <number>-times the stiffness homogeneity of the vertical configuration, resulting in the angled configuration showing an overall increase in static stiffness homogeneity by a factor of 1/<number> over the vertical configuration.
- **Deterioration After Reorienting.** As for <number> static stiffness values, smaller ratio means greater deterioration, i.e., the angled configuration exhibits <number>-times smaller the respective value. As for <number> stiffness homogeneities, larger ratio means greater deterioration, i.e., the angled configuration's stiffness homogeneity is <number>-times the stiffness homogeneity of the vertical configuration, resulting in the angled configuration showing an overall decrease in static stiffness homogeneity by a factor of 1/<number> over the vertical configuration.

Level 1. $z = 1\,200$ [mm]					
Minimum principal stiffness xy ratio		Maximum principal stiffness xy ratio			
min	0.123	min	0.505		
max	1.494	max	4.000		
med	0.944	med	0.854	homogeneity ratio	0.905
Level 2. $z = 1\,500$ [mm]					
Minimum principal stiffness xy ratio		Maximum principal stiffness xy ratio			
min	0.213	min	0.500		
max	0.481	max	0.177		
med	0.856	med	0.400	homogeneity ratio	0.467
Level 3. $z = 1\,800$ [mm]					
Minimum principal stiffness xy ratio		Maximum principal stiffness xy ratio			
min	0.015	min	5.959		
max	0.530	max	1.820		
med	0.366	med	0.590	homogeneity ratio	1.613

Table 6.22 Comparison of Cases c-C4 and c-C1. Ratios taken as c-C4/c-C1.

Level 1. $x = 1\,700$ [mm]					
Minimum principal stiffness yz ratio		Maximum principal stiffness yz ratio			
min	0.242	min	12.770		
max	0.650	max	0.333		
med	0.764	med	0.334	homogeneity ratio	0.438
Level 2. $x = 1\,950$ [mm]					
Minimum principal stiffness yz ratio		Maximum principal stiffness yz ratio			
min	1.268	min	12.565		
max	0.244	max	0.084		
med	0.781	med	0.420	homogeneity ratio	0.536
Level 3. $x = 2\,200$ [mm]					
Minimum principal stiffness yz ratio		Maximum principal stiffness yz ratio			
min	0.185	min	16.130		
max	0.391	max	0.020		
med	0.758	med	0.455	homogeneity ratio	0.600

Table 6.23 Comparison of Cases c-C5 and c-C2. Ratios taken as c-C5/c-C2.

Level 1. $y = -500$ [mm]					
Minimum principal stiffness xz ratio		Maximum principal stiffness xz ratio			
min	0.164	min	0.345		
max	1.140	max	5.945		
med	1.140	med	0.206	homogeneity ratio	0.181
Level 2. $y = 0$ [mm]					
Minimum principal stiffness xz ratio		Maximum principal stiffness xz ratio			
min	0.032	min	3.320		
max	0.155	max	1.735		
med	0.087	med	0.183	homogeneity ratio	2.100
Level 3. $y = 500$ [mm]					
Minimum principal stiffness xz ratio		Maximum principal stiffness xz ratio			
min	0.060	min	0.360		
max	0.784	max	3.600		
med	0.646	med	1.230	homogeneity ratio	1.900

Table 6.24 Comparison of Cases c-C6 and c-C3. Ratios taken as c-C6/c-C3.

6.17 Discussion of Results

Ultimately, it may be inferred by the reader that, while stiffness maps (quiver-contour plots) provide insight into how stiffness varies across the examined work envelope, the most pivotal aspect of the results lies in the comparison tables. Stiffness maps fall short compared to numerical data for two primary reasons. Firstly, the contour lines, or isolines, would have been more precise and densely distributed had the model been simulated across a larger set of points. Secondly, due to the necessity of flattening all data to a set of planes for reasonable visualization, the directional field becomes distorted. This distortion arises because the third component of all vectors is consistently constrained to lie within the respective plane, rendering the vector a mere projection. Consequently, while these vectors accurately represent principal directions within the respective plane, they lack information pertaining to three-dimensional space. Nevertheless, stiffness maps provide a useful connection to the presented values.

As for the values themselves, the most interesting are the median and homogeneity values, which result as the ratio between maximum and minimum principal stiffness medians. The specific values are heavily influenced by what torsion spring stiffness we have chosen, accompanied with its damping coefficient. In hindsight, choosing $\kappa_\tau = 2.5 \cdot 10^4$ $[\mathrm{N\,m\,rad^{-1}}]$ was ultimately not enough, considering the mass of the individual links. Henceforth, the values are simply relative and what matters most are, again said, the ratios between configurations. Furthermore, in many instances, the maxima and minima values deviate far from the median. From stiffness maps, it can be seen these deviations often occur at one singular point, wherein the simulation might have encountered anomalies or a kinematic singularity was reached. Moreover, the gravity compensation algorithm could influence simulation, along with other effects. It is because of the aforementioned reasons the peaks and valleys of the minimum and maximum principal stiffnesses, yet also individual values, are of less significance.

In aggregate, across all investigated cases and in light of the comparison tables, the coupling of two robots opposite one another emerges as the preeminent configuration with respect to both static stiffness values and, notably, homogeneity. This assertion holds irrespective of whether the pairing occurs in the vertical or in the angular configuration, with the increase in homogeneity consistently evident, often by a factor surpassing 10. Concerning the stiffness values per se, the coupled configuration evinces a proclivity towards heightened minimum principal stiffness, particularly in its maximum and median values, frequently by a factor exceeding 10 or 20. Conversely, save for a few exceptions, the progression of maximum values does not exhibit such escalation, with the prevailing trend indicating an increase by approximately 2-, 3-, or occasionally 5-fold. Yet, we again mention the ratios between these values are highly relative as they depend on various parameters, discussed before. Further, if one configuration reaches an absurdly enormous peak

and the other does not, the ratio is less valuable and does not necessarily extend our insight into static stiffness of six-axis robots.

Additionally, when juxtaposing all cases in the angled configuration with the cases in the vertical configuration, we come to the resolution that the angled configuration is more susceptible to lower minimum and maximum principal stiffness values. While this is not always the case, it is still a relevant statement in light of the comparison tables for both the coupled robots and a solitary one. As for stiffness homogeneity, its improvement or deterioration in the angled configuration compared to the vertical configuration is not uniform, hence why the author refrains from further general discussion of such, and refers the reader to the respective comparison table to see whether the static stiffness homogeneity has improved or worsened in the specific case they inquire.

7
Conclusion and Outlook

The focal point of this thesis was spatial static stiffness in three dimensions, particularly for six-axis serial robots of the anthropomorphic structure. During the inquiry into such topic, the reader was presented with the monumental significance of spatial stiffness, accompanied with an outline of the past and current developments on the subject. Next, the necessary theoretical groundwork was established, ranging from kinetostatics of rigid systems (Chapter 1) to nonlinearity and singular value decomposition (Chapter 3), which ultimately tied all the previously developed theory into a self-contained methodology for static stiffness evaluation, necessarily preceded by static compliance.

Within the simulation portion of the thesis, the model of a real, six-axis robot by KUKA was assembled in DS SolidWorks, and subsequently in Simscape MBS (Chapter 4). This allowed the author to perform numerous simulations of the behaviour of the robotic system under force load on its tool center point. For such investigation, multiple algorithms were created (Chapter 5), in order to streamline the data gathering process. Moreover, said algorithms were adapted for the case of robot coupling. As for results (Chapter 6), the coupled configuration came out on top in having the best stiffness homogeneity ratio and values, further supporting our case of robot coupling, in this instance opposite each other.

7.1 Missed Opportunities

Although the thesis provides a comprehensive overview of static stiffness of six axis serial robots, more work can always be done. For one, stiffness can be examined equally in terms of dynamics as it can be in terms of statics. Creating a dynamical model of such complicated kinematic structure was beyond the scope of this thesis, yet the inclusion of robot dynamics within the model is pivotal, as it accounts

for all phenomena statics could never account for. Further, such dynamic model, developed using either the Lagrange equations of the second kind or the Lagrange equations of the mixed kind, would accelerate result generation since the model needn't to be simulated at each instance. Instead, the desired pose of the tool center point could be substituted into the eigenequations of motion of the system, yielding a transfer function instantaneously. This directly allows more points to be evaluated, leading to an increase in the fidelity of stiffness maps.

Additionally, only one position of the accompanying robot was investigated, that being opposite the formerly solitary arm. It would be interesting to not have the robots be distributed 180 degrees but some different angle, e.g., 45 or 90 degrees, or even have three robots working together. The intuition is, the more robots, the better the stiffness. Yet, until verified, it cannot be determined with absolute certainty. Moreover, attaching a parallel delta robot to the top of the end-effector could increase stability, and thereby stiffness, even more.

7.2 Future Work

The ideas for future work naturally stem from all of the missed opportunities. For one, a dynamic model of the robot ought to be developed to deepen our understanding of the distribution of stiffness within three-dimensions, at least for what concerns six-axis robots. To not limit ourselves to six-axis serial robots, the stiffness of either one attached parallel robot to the serial robots configuration, or solely one, two, or more parallel robots, can be examined. Lastly, approaching the problem using different layouts of robots, e.g., distributed evenly by 45 degrees, can further enhance our inquiry into stiffness in general.

Bibliography

1. Asimov, I. *I, Robot*. New York, NY: Bantam Books, 2008. Robot Series. ISBN 978-0-553-38256-3.
2. Sun, L.; Fang, L. An Approximation Method for Stiffness Calculation of Robotic Arms with Hybrid Open- and Closed-Loop Kinematic Chains. *Advances in Mechanical Engineering*. 2018, vol. 10, no. 2. DOI 10.1177/1687814018761297.
3. Hu, J.; Liu, T., et al. Static Modeling of a Class of Stiffness-Adjustable Snake-like Robots with Gravity Compensation. *Robotics*. 2023, vol. 12, no. 2. DOI 10.3390/robotics12010002.
4. Wu, K.; Li, J., et al. Review of Industrial Robot Stiffness Identification and Modelling. *Applied Sciences*. 2022, vol. 12, no. 17. DOI 10.3390/app12178719.
5. Liang, M.; Wang, B., et al. Dynamic Optimization of Robot Arm Based on Flexible Multi-Body Model. *Journal of Mechanical Science and Technology*. 2017, vol. 31, no. DOI 10.3390/app12178719.
6. Živanović, M.; Vukobratović, M. General Mathematical Model of Multi-Arm Cooperating Robots with Elastic Interconnection at the Contact. *Journal of Dynamic Systems, Measurement, and Control*. 1997, vol. 119, no. 4. DOI 10.1115/1.2802381.
7. Živanović, M.; Vukobratović, M. Control Laws Synthesis of Multi-Arm Cooperating Robots with Elastic Interconnection at the Contacts. *Robotica*. 2000, vol. 18, no. 2. DOI 10.1017/S0263574799001940.

8. Živanović, M.; Vukobratović, M. Synthesis of Nominal Motion of the Multi-Arm Cooperating Robots With Elastic Interconnections at the Contacts. *Journal of Dynamic Systems, Measurement, and Control*. 2004, vol. 126, no. 2. DOI 10.1115/1.1767853.

9. Du, X.; Tete, H. Modelling and Continuous Stiffness Control of Robot with Compliant Wrist for Misalignment Shaft-Hole Assembly. *Measurement and Control*. 2024, vol. 57, no. 1. DOI 10.1016/j.rcim.2023.102676.

10. Luo, Q.; Zhang, X, et al. Research on a Biomimetic Flexible Ball Joint With Variable Stiffness for Robots. *ASME Journal of Mechanisms and Robotics*. 2024, vol. 16, no. 10. DOI 10.1016/j.rcim.2023.102676.

11. Kun, Ch.; Peng, X, et al. Interactive Coupling of Structural Dynamics and Milling Forces for High-Frequency Stability Prediction in Robotic Milling. *Robotics and Computer-Integrated Manufacturing*. 2024, vol. 86, no. DOI 10.1016/j.rcim.2023.102676.

12. Lynch, K.; Park, F. *Modern Robotics: Mechanics, Planning, and Control*. UK: Cambridge University Press, 2017. ISBN 978-1-107-15630-2.

13. Valášek, M. et al. *Mechanika A: Lecture Notes*. Prague: CTU, 2002.

14. Stejskal, V. et al. *Mechanika I: Lecture Notes*. Prague: CTU, 1998.

15. Siciliano, B. et al. *Robotics: Modelling, Planning, and Control*. 1st ed. London, England: Springer, 2018. Advanced Textbooks in Control and Signal Processing. ISBN 978-1-84996-634-4.

16. Belta, C.; Kumar, V. *On the Computation of Rigid Body Motion*. 2009. Available at: https://api.semanticscholar.org/CorpusID:9281725.

17. Bump, D. *Lie Groups*. 2nd ed. New York, NY: Springer, 2013. Graduate Texts in Mathematics. ISBN 978-1-4614-8023-5.

18. Humphreys, J. *Introduction to Lie Algebras and Representation Theory*. New York, NY: Springer, 1973. Graduate Texts in Mathematics. ISBN 978-0-387-90052-0.

19. Jazar, R. *Theory of Applied Robotics: Kinematics, Dynamics, and Control*. 2nd ed. New York, NY: Springer, 2010. ISBN 978-1-4899-7760-1.

20. Kosinski, A. *Differential Manifolds*. Mineola, NY: Dover Publications, 2007. Dover Books on Mathematics. ISBN 978-0-4864-6244-8.

21. Lee, J. *Introduction to Smooth Manifolds*. 2nd ed. New York, NY: Springer, 2012. Graduate Texts in Mathematics. ISBN 978-1-4419-9981-8.

22. do Carmo, M. *Riemannian Geometry*. 4th ed. Basel, Switzerland: Birkhauser, 1991. Mathematics: Theory and Applications. ISBN 978-3-7643-3490-1.

23. Munkres, J. *Topology*. 2nd ed. London, England: Pearson Education, 2013. ISBN 978-1-292-02362-5.

24. Pressley, A. *Elementary Differential Geometry*. 2nd ed. London, England: Springer, 2009. Springer Undergraduate Mathematics Series. ISBN 978-1-84882-890-2.

25. Lay, D. et al. *Linear Algebra and its Applications*. 5th ed. Upper Saddle River, NJ: Pearson, 2014. ISBN 978-0-321-98238-4.

26. Axler, S. *Linear Algebra Done Right*. 4th ed. Cham, Switzerland: Springer, 2024. Undergraduate Texts in Mathematics. ISBN 978-3-031-41025-3.

27. Angeles, J. *Fundamentals of Robotic Mechanical Systems: Theory, Methods, and Algorithms*. 3rd ed. New York, NY: Springer, 2006. Mechanical Engineering Series. ISBN 0-387-29412-0.

28. Kuipers, J. *Quaternions and Rotation Sequences*. Princeton, NJ: Princeton University Press, 2002. ISBN 0-691-10298-8.

29. Craig, J. *Introduction to Robotics*. 3rd ed. Upper Saddle River, NJ: Pearson, 2003. ISBN 0-13-123629-6.

30. Goldstein, H. *Classical Mechanics*. 3rd ed. Philadelphia, PA: Pearson Education, 2011. ISBN 978-81-317-5891-5.

31. Spong, M.; Hutchinson, S., et al. *Robot Modelling and Control*. 2nd ed. Nashville, TN: John Wiley & Sons, 2020. ISBN 978-1-119-52399-4.

32. Strang, G. *Introduction to Linear Algebra*. 4th ed. Wellesley, MA: Wellesley-Cambridge Press, 2009. ISBN 978-0-9802327-1-4.

33. Golub, H.; Van Loan, C. *Matrix Computations*. 4th ed. Baltimore, MD: Johns Hopkins University Press, 2013. Johns Hopkins Studies in the Mathematical Sciences. ISBN 978-1-421-40794-4.

34. Valášek, M. et al. *Mechanika B: Lecture Notes*. Prague: CTU, 2004.

35. The MathWorks, Inc. *MATLAB Documentation*. Available at: `https://www.mathworks.com/help/matlab/`.

36. The MathWorks, Inc. *Simulink Documentation*. Available at: `https://www.mathworks.com/help/simulink/`.

37. The MathWorks, Inc. *Simscape Documentation*. Available at: `https://www.mathworks.com/help/simscape/`.

38. The MathWorks, Inc. *Simscape Multibody Documentation*. Available at: `https://www.mathworks.com/help/sm/`.

39. Khalil, H. *Nonlinear Systems*. 2nd ed. Upper Saddle River, NJ: Pearson, 1995. ISBN 0-13-228024-8.

40. Hangos, K.; Lakner, R., et al. *Intelligent Control Systems*. New York, NY: Springer, 2001. Applied Optimization. ISBN 978-1-4020-0134-5.

41. Hangos, K.; Bokor, J., et al. *Analysis and Control of Nonlinear Process Systems*. London, England: Springer, 2004. Advanced Textbooks in Control and Signal Processing. ISBN 978-1-85233-600-4.

42. Slotine, J-J.; Li, W. *Applied Nonlinear Control*. Upper Saddle River, NJ: Pearson, 1990. ISBN 0-13-040890-5.

43. Ogata, K. *Modern Control Engineering*. 5th ed. Upper Saddle River, NJ: Pearson, 2009. ISBN 978-0-13-615673-4.

44. Bhatia, P.; Szegö, Giorigio. *Stability Theory of Dynamical Systems*. Berlin, Germany: Springer, 2002. Classics in Mathematics. ISBN 3-540-42748-1.

45. Rannacher, R. *Numerical Linear Algebra*. 3rd ed. Heidelberg, Germany: Heidelberg University Publishing, 2018. Lecture Notes Mathematics. ISBN 978-3-947732-00-5.

46. Demmel, J. *Applied Numerical Linear Algebra*. Philadelphia, MS: Society for Industrial and Applied Mathematics, 2018. ISBN 978-0-89-871389-3.

47. Anton, H.; Rorres, C. *Elementary Linear Algebra*. Nashville, TN: John Wiley & Sons, 2013. ISBN 978-1-118-43441-3.

48. Lang, S. *Linear Algebra*. 3rd ed. New York, NY: Springer, 1987. Undergraduate Texts in Mathematics. ISBN 978-0-38-796412-6.

49. KR 120 R2700-2. KUKA, 2022. Document ID: 0000-325-899. V4.1.

50. Riley, K.; Hobson, M., et al. *Mathematical Methods for Physics and Engineering*. 3rd ed. Cambridge, England: Cambridge University Press, 2006. ISBN 978-0-521-86153-3.

51. Corke, P. *Robotics, Vision, and Control: Fundamental Algorithms in MATLAB*. 1st ed. Berlin, Germany: Springer, 2011. ISBN 978-3-642-20143-1.

52. E., Hitzer; Sangwine, S. *Quaternion and Clifford Fourier Transforms and Wavelets*. Basel, Switzerland: Springer, 2013. ISBN 978-3-0348-0602-2.

53. Goode, S.; Annin, S. *Differential Equations and Linear Algebra*. 4th ed. Upper Saddle River, NJ: Pearson, 2015. ISBN 978-0-321-96467-0.

54. Sohrab, H. *Basic Real Analysis*. 2nd ed. New York, NY: Springer, 2014. ISBN 978-1-4939-1840-9.

55. Bartle, R.; Sherbert, D. *Introduction to Real Analysis*. 4th ed. Nashville, TN: John Wiley & Sons, 2011. ISBN 978-0-471-43331-6.

56. Hildebrand, F. *Introduction to Numerical Analysis*. 2nd ed. Mineola, NY: Dover Publications, 1987. Dover Books on Mathematics. ISBN 0-486-65363-3.

57. Sauer, T. *Numerical Analysis*. 2nd ed. Upper Saddle River, NJ: Pearson, 2011. ISBN 978-0-321-78367-7.

58. Quarteroni, A.; Sacco, R., et al. *Numerical Mathematics*. 2nd ed. Berlin, Germany: Springer, 2007. Texts in Applied Mathematics. ISBN 978-3-540-34658-6.

59. Wiggins, S. *Introduction to Applied Nonlinear Dynamical Systems and Chaos*. 2nd ed. New York, NY: Springer, 2003. Texts in Applied Mathematics. ISBN 0-387-00177-8.

60. Isidori, A. *Nonlinear Control Systems*. 3rd ed. Berlin, Germany: Springer, 1995. Communications and Control Engineering. ISBN 3-540-19916-0.

61. Nocedal, J.; Wright, S. *Numerical Optimization*. 2nd ed. New York, NY: Springer, 2006. Springer Series in Operations Research and Financial Engineering. ISBN 978-0387-30303-1.

62. Dennis, J.; Schnabel, R. *Numerical Methods for Unconstrained Optimization and Nonlinear Equations*. Society for Industrial and Applied Mathematics, 1996. Classics in Applied Mathematics. ISBN 0-89871-364-1.

63. The MathWorks, Inc. *Optimization Toolbox Documentation*. Available at: `https://www.mathworks.com/help/optim/`.

Appendices

A
Other Orientation Representations

As was highlighted in Chapter 1, there exist various other orientation representations, two of which are the subject of this appendix, namely the Euler-Rodrigues parameters, also referred to as the *unit quaternion* representation (Section A.1), and the Cayley-Rodrigues parameters (Section A.2). Both of these representations require "higher level" mathematics and mechanics in general, e.g., quaternion algebra, the exponential coordinate representation of rotation, etc., which are not discussed within the main portion of the thesis. Even though these former mentioned mathematical and mechanical instruments go beyond what is reasonable to present in the main body of this thesis, they find many applications in robotics, computer graphics, and other disciplines, where in some scenarios, they prove more useful and convenient than the standard, "lower-level" representations, hence why we choose to mention them as part of this appendix.

A.1 Euler-Rodrigues Parameters

Previously, we touched upon rotation about an arbitrary axis in space. Let us expand on this idea and discover an inconvenient singularity in the inverse solution which leads us to the need of a different orientation representation.

A.1.1 Axis-Angle Representation Singularity

Let $\boldsymbol{a} = [a_x \quad a_y \quad a_z]^T \in \mathbb{R}^3$ be the unit vector of axis $\mathfrak{a}$ with respect to $O_i x_i y_i z_i$ and φ the angle of revolution about $\mathfrak{a}$. Note that this representation is nonminimal in $SO(3)$ as it requires $n(n-1)/2 + 1$ parameters.

If we adapt a solution for this rotation from [15], pp. 52 — 54, it is clear to see that, in order to rotate $O_i x_i y_i z_i$ about $\mathfrak{a}$ by φ to obtain $O_\ell x_\ell y_\ell z_\ell$, we need to

perform the following steps (as per Figure A.1):

rotate $O_i x_i y_i z_i$ by $-\varrho$ about the z_i axis and by $-\chi$ about the y_i axis to obtain the system $O_j x_j y_j z_j$, ⟵ Start with an arbitrary coordinate system $O_i x_i y_i z_i$,

↓

rotate $O_j x_j y_j z_j$ by the desired angle of revolution φ about z_j to obtain the system $O_k x_k y_k z_k$, ⟶ realign $O_k x_k y_k z_k$ with $\mathfrak{a}$ by rotating by χ about y_k and by ϱ about z_k to obtain the system $O_\ell x_\ell y_\ell z_\ell$.

The resulting orientation of $O_\ell x_\ell y_\ell z_\ell$ within $O_i x_i y_i z_i$ is then

$$\begin{aligned}\boldsymbol{R}_{i\ell}(\mathfrak{a},\varphi) &= \overbrace{\boldsymbol{R}_{k\ell}(z_k,\varrho)\boldsymbol{R}_{k\ell}(y_k,\chi)}^{\text{realignment}}\overbrace{\boldsymbol{R}_{jk}(z_j,\varphi)}^{\text{desired rotation}}\overbrace{\boldsymbol{R}_{ij}(y_i,-\chi)\boldsymbol{R}_{ij}(z_i,-\varrho)}^{\text{misalignment}} \\ &= \begin{bmatrix} a_x^2\mathscr{C}+\mathrm{c}_\varphi & a_x a_y\mathscr{C}-a_z\mathrm{s}_\varphi & a_x a_z\mathscr{C}+a_y\mathrm{s}_\varphi \\ a_x a_y\mathscr{C}+a_z\mathrm{s}_\varphi & a_y^2\mathscr{C}+\mathrm{c}_\varphi & a_y a_z\mathscr{C}-a_x\mathrm{s}_\varphi \\ a_x a_z\mathscr{C}-a_y\mathrm{s}_\varphi & a_y a_z\mathscr{C}+a_x\mathrm{s}_\varphi & a_z^2\mathscr{C}+\mathrm{c}_\varphi \end{bmatrix},\end{aligned}$$

where $\mathscr{C} = 1-\cos\varphi$ and s_φ, c_φ is shorthand notation for $\sin\varphi$, $\cos\varphi$ respectively. In this case, $\boldsymbol{R}_{i\ell}(\mathfrak{a},\varphi) = \boldsymbol{R}_{i\ell}(-\mathfrak{a},-\varphi)$, hence such representation is *not* unique, as rotation by φ about $\mathfrak{a}$ ($\boldsymbol{a}$ is *positive*) is indistinguishable from rotation by $-\varphi$ about $-\mathfrak{a}$ ($\boldsymbol{a}$ is *negative* — points in the opposite direction). [15]

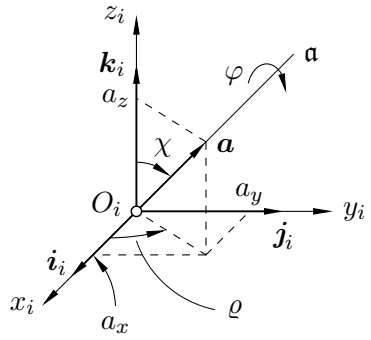


Figure A.1 Axis-angle rotation representation. Figure adapted from [15].

Additionally, in order to remove the dependence of $\boldsymbol{R}_{i\ell}(\mathfrak{a},\varphi)$ on ϱ and χ, we have utilized the following transcendental functions derived from Figure A.1:

$$\sin\varrho = \frac{a_y}{\sqrt{a_x^2+a_y^2}}, \qquad \sin\chi = \sqrt{a_x^2+a_y^2},$$

$$\cos\varrho = \frac{a_x}{\sqrt{a_x^2+a_y^2}}, \qquad \cos\chi = a_z.$$

When solving the inverse problem, the axis and angle corresponding to a given rotation matrix of the form (1.3) can be obtained as [15]

$$\begin{aligned} \varphi &= \arccos \frac{\operatorname{tr}(\boldsymbol{R}_{i\ell}) - 1}{2}, \\ \boldsymbol{a} &= \frac{1}{2\sin\varphi} \begin{bmatrix} r_{32} - r_{23} \\ r_{13} - r_{31} \\ r_{21} - r_{12} \end{bmatrix}, \end{aligned} \tag{A.1}$$

where $\operatorname{tr}(\boldsymbol{R}_{i\ell})$ is the *trace* of the rotation matrix, formally defined as

$$\operatorname{tr}(\boldsymbol{A}) \coloneqq \sum_{i=1}^{n} a_{ii},$$

for any $n \times n$ (square) matrix $\boldsymbol{A}$.

As we can see, solution (A.1) diverges if $\sin\varphi = 0$, i.e., when

$$\varphi = \begin{cases} 0 & \text{(null rotation)}, \\ \pi, \end{cases}$$

which are still valid solutions. In such instances, it becomes necessary to directly refer to the expressions obtained from the provided rotation matrix and seek a solution through this method. [15]

Theorem A.1. In the case of the null rotation, i.e., $\varphi = 0$, the unit vector $\boldsymbol{a}$ of the axis of revolution $\mathfrak{a}$ becomes arbitrary. [15] ◇

Proof. The (i, j)-th entry of the rotation matrix $\boldsymbol{R}_{i\ell}(\mathfrak{a}, \varphi)$ can be expressed as

$$r_{ij} = \delta_{ij} + (1 - \cos\varphi) a_i a_j - \sin\varphi \epsilon_{ijk} a_k, \tag{A.2}$$

where $a_i \equiv a_x$, $a_j \equiv a_y$, $a_k \equiv a_z$,

$$\delta_{ij} \coloneqq \begin{cases} 1 & \text{if } i = j, \\ 0 & \text{otherwise}, \end{cases}$$

is the Kronecker delta, and

$$\epsilon_{ijk} \coloneqq \begin{cases} 1 & \text{if } (i,j,k) \text{ is an } \textit{even} \text{ permutation of } (1,2,3), \\ -1 & \text{if } (i,j,k) \text{ is an } \textit{odd} \text{ permutation of } (1,2,3), \\ 0 & \text{otherwise}, \end{cases}$$

is the Levi-Civita symbol. When $\varphi = 0$, the (i, j)-th entry of the rotation matrix becomes $r_{ij} = \delta_{ij}$ and hence $\boldsymbol{R}_{i\ell}(\mathfrak{a}, \varphi = 0) = \boldsymbol{I}_3$, eliminating *all* components of $\boldsymbol{a}$ from (A.2). This results in an *arbitrary* unit vector $\boldsymbol{a}$ since the null rotation *always* yields the identity matrix, which concludes the proof. [19, 50] □

Remark A.1. Eq. (A.2) from Theorem A.1 can be written in matrix form as

$$\boldsymbol{R}_{i\ell}(\mathfrak{a}, \varphi) = \boldsymbol{I}_3 + \sin\varphi\,[\boldsymbol{A}] + (1 - \cos\varphi)\,[\boldsymbol{A}]^2\,, \tag{A.3}$$

in which $[\boldsymbol{A}] \in \mathbb{R}^{3\times3}$ is a skew-symmetric matrix derived from the unit vector $\boldsymbol{a}$,

$$[\boldsymbol{A}] = \begin{bmatrix} 0 & -a_z & a_y \\ a_z & 0 & -a_x \\ -a_y & a_x & 0 \end{bmatrix} \in \mathbb{R}^{3\times3}, \quad \underbrace{[\boldsymbol{A}]^T = -\,[\boldsymbol{A}]}_{\text{skew-symmetry}}.$$

Again, if $\varphi = 0$, $\boldsymbol{R}_{i\ell}(\mathfrak{a}, \varphi = 0)$ becomes the identity matrix. [12] ◇

A.1.2 Unit Quaternion Representation

As was highlighted previously, the inverse solution of the axis-angle representation becomes singular at $\varphi = 0$ since the unit vector $\boldsymbol{a}$ becomes arbitrary, giving us no information about the axis of revolution $\mathfrak{a}$. Moreover the axis-angle representation is *not* unique. To address these limitations, it is necessary to adopt an alternative four-parameter representation. One such example are the Euler-Rodrigues parameters, defined as [12, 15, 19, 27, 29]

$$\mathcal{Q} \coloneqq \begin{bmatrix} q_0 & q_1 & q_2 & q_3 \end{bmatrix}^T \in \mathbb{R}^4,$$

where

$$q_0 \coloneqq \cos(\varphi/2) \in \mathbb{R}, \tag{A.4}$$

$$\boldsymbol{q} \coloneqq \sin(\varphi/2)\boldsymbol{a} = \begin{bmatrix} q_1 & q_2 & q_3 \end{bmatrix}^T \in \mathbb{R}^3, \tag{A.5}$$

and naturally

$$\|\mathcal{Q}\| = q_0^2 + q_1^2 + q_2^2 + q_3^2 = 1. \tag{A.6}$$

If the Euler-Rodrigues parameters are looked at as a 4×1 vector, they are also referred to as the *unit quaternion* representation, since $\|\mathcal{Q}\| = 1$. Geometrically, an orientation described using the unit quaternion representation is visualized as a point on a hypersphere in $\mathbb{R}^4$. [12, 29] This can seem hard to grasp, yet their benefits make them a very ubiquitous and versatile option in robotics, computer graphics, and other subjects, with some of their biggest strengths being:

▷ **Uniqueness.** Every rotation within three-dimensional space is associated with precisely one unit quaternion. This distinctive property guarantees that there are no redundant or multiple representations for identical rotations. [15, 27, 51]

▷ **Continuity.** Unit quaternions provide seamless interpolation between rotations. In contrast to Euler angles, which may encounter discontinuities, quaternions offer smooth transitions between orientations. [12, 15, 28]

▷ **Numerical Stability.** Quaternion operations manifest superior numerical stability relative to alternative representations, thereby evincing diminished susceptibility to numerical errors. [28, 29, 52]

▷ **Global Coverage.** Unit quaternions provide comprehensive coverage of rotations throughout space, devoid of any singularities or discontinuities, unlike certain other representations like Euler angles. [12, 15, 28, 52]

Given a unit quaternion $\mathcal{Q} = [q_0 \quad \boldsymbol{q}]^T \in \mathbb{R}^4$, the corresponding rotation matrix is obtained by rotating about the unit axis in the direction of $\boldsymbol{q} \in \mathbb{R}^3$ by $\varphi = 2 \arccos q_0$ and reads

$$\begin{aligned} \boldsymbol{R}_{ij}(\mathcal{Q}) &= \Big(q^2 - \langle \boldsymbol{q}, \boldsymbol{q} \rangle\Big)\boldsymbol{I}_3 + 2\boldsymbol{q}\boldsymbol{q}^T + 2q_0 \frac{\partial(\boldsymbol{q} \times \boldsymbol{x})}{\partial \boldsymbol{x}} \\ &= \begin{bmatrix} 2(q_0^2 + q_1^2) - 1 & 2(q_1 q_2 - q_0 q_3) & 2(q_1 q_3 + q_0 q_2) \\ 2(q_1 q_2 + q_0 q_3) & 2(q_0^2 + q_2^2) - 1 & 2(q_2 q_3 - q_0 q_1) \\ 2(q_1 q_3 - q_0 q_2) & 2(q_2 q_3 + q_0 q_1) & 2(q_0^2 + q_3^2) - 1 \end{bmatrix}, \end{aligned} \tag{A.7}$$

for an arbitrary vector $\boldsymbol{x} \in \mathbb{R}^3$. [12, 15, 27]

If a generic rotation matrix $\boldsymbol{R}_{ij} \in SO(3)$ of the form (1.3) is provided, the inverse solution is then given by

$$q_0 = \frac{1}{2}\sqrt{\operatorname{tr}(\boldsymbol{R}_{ij}) + 1}, \tag{A.8}$$

$$\boldsymbol{q} = \frac{1}{2}\begin{bmatrix} \operatorname{sgn}(r_{32} - r_{32})\sqrt{r_{11} - r_{22} - r_{33} + 1} \\ \operatorname{sgn}(r_{13} - r_{31})\sqrt{r_{22} - r_{33} - r_{11} + 1} \\ \operatorname{sgn}(r_{21} - r_{12})\sqrt{r_{33} - r_{11} - r_{22} + 1} \end{bmatrix} \equiv \frac{1}{4q_0}\begin{bmatrix} r_{32} - r_{32} \\ r_{13} - r_{31} \\ r_{21} - r_{12} \end{bmatrix}, \tag{A.9}$$

where

$$\operatorname{sgn}(w) \coloneqq \begin{cases} 1 & \text{if } w \geq 0, \\ -1 & \text{if } w < 0, \end{cases}$$

for an arbitrary $w \in \mathbb{R}$. Note that by (A.8), we assume that $q_0 \geq 0$, corresponding to $\varphi \in [-\pi, \pi]$. Consequently, any rotation can be described as elucidated in the global coverage property. Further, (A.8, A.9) are *nonsingular*. [12, 15, 29]

Finally, let us discuss the inverse and product properties of the unit quaternion representation. The quaternion derived from the inverse of the rotation matrix $\boldsymbol{R}_{ij}^{-1} = \boldsymbol{R}_{ij}^T \in SO(3)$ can be computed as

$$\mathcal{Q}^{-1} = \begin{bmatrix} q_0 & -\boldsymbol{q} \end{bmatrix}^T \in \mathbb{R}^4.$$

On the other hand, if we let $\mathcal{Q} = [q_0 \quad \boldsymbol{q}]^T \in \mathbb{R}^4$ and $\mathcal{Q}^\star = [q_0^\star \quad \boldsymbol{q}^\star]^T \in \mathbb{R}^4$ be two unit quaternions corresponding to rotation matrices $\boldsymbol{R}_{ij}$, $\boldsymbol{R}_{jk} \in SO(3)$,

respectively, the unit quaternion representation of $\boldsymbol{R}_{ij}\boldsymbol{R}_{jk} \in SO(3)$ can be obtained by first arranging the entries of $\mathcal{Q}$ and $\mathcal{Q}^\star$ to the form

$$\boldsymbol{Q} = \begin{bmatrix} q_0 + iq_1 & q_2 + iq_3 \\ -q_2 + iq_3 & q_0 - iq_1 \end{bmatrix}, \quad \boldsymbol{Q}^\star = \begin{bmatrix} q_0^\star + iq_1^\star & q_2^\star + iq_3^\star \\ -q_2^\star + iq_3^\star & q_0^\star - iq_1^\star \end{bmatrix},$$

where $i \coloneqq \sqrt{-1}$ is the imaginary unit. The product $\widetilde{\boldsymbol{Q}} = \boldsymbol{Q}\boldsymbol{Q}^\star$ reads

$$\widetilde{\boldsymbol{Q}} = \begin{bmatrix} \tilde{q}_0 + i\tilde{q}_1 & \tilde{q}_2 + i\tilde{q}_3 \\ -\tilde{q}_2 + i\tilde{q}_3 & \tilde{q}_0 - i\tilde{q}_1 \end{bmatrix}.$$

The product of $\mathcal{Q}$ and $\mathcal{Q}^\star$ can be directly obtained from the entries of $\widetilde{\boldsymbol{Q}}$ and yields a new unit quaternion $\mathcal{Q}' \in \mathbb{R}^4$ of the form

$$\begin{aligned} \mathcal{Q}' = \mathcal{Q} \circ \mathcal{Q}^\star &= \begin{bmatrix} \tilde{q}_0 & \tilde{q}_1 & \tilde{q}_2 & \tilde{q}_3 \end{bmatrix}^T \\ &= \begin{bmatrix} q_0q_0^\star - q_1q_1^\star - q_2q_2^\star - q_3q_3^\star \\ q_0q_1^\star + q_1q_0^\star + q_2q_3^\star - q_3q_2^\star \\ q_0q_2^\star + q_2q_0^\star - q_1q_3^\star + q_3q_1^\star \\ q_0q_3^\star + q_3q_0^\star + q_1q_2^\star - q_2q_1^\star \end{bmatrix} \equiv \begin{bmatrix} q_0' \\ q_1' \\ q_2' \\ q_3' \end{bmatrix} \in \mathbb{R}^4, \end{aligned}$$

or more generally,

$$\mathcal{Q}' = \mathcal{Q} \circ \mathcal{Q}^\star = \begin{bmatrix} q_0q_0^\star - \langle \boldsymbol{q}, \boldsymbol{q}^\star \rangle & q_0\boldsymbol{q}^\star + q_0^\star\boldsymbol{q} + \boldsymbol{q} \times \boldsymbol{q}^\star \end{bmatrix}^T \in \mathbb{R}^4,$$

where $\circ$ denotes the quaternion product.[32] [12, 15]

A.2 Cayley-Rodrigues Parameters

Another set of parameters representing orientation are the Cayley-Rodrigues parameters. Since they are derived from the exponential coordinate representation on $SO(3)$, let us first get started by defining such representation. [12]

A.2.1 Fundamentals of Linear Differential Equations

Consider a linear differential equation of the form

$$\dot{\boldsymbol{x}}(t) = \boldsymbol{A}\boldsymbol{x}(t), \text{ where } \boldsymbol{x}(t) \in \mathbb{R}^n, \ \boldsymbol{A} \in \mathbb{R}^{n \times n}, \ \boldsymbol{x}(0) = \boldsymbol{x}_0. \tag{A.10}$$

By basic LDR theory,[33] the solution to such equation is

$$\boldsymbol{x}(t) = \exp(\boldsymbol{A}t)\boldsymbol{x}_0,$$

[32] The quaternion product is generally *not* commutative, i.e., $\mathcal{Q} \circ \mathcal{Q}^\star \neq \mathcal{Q}^\star \circ \mathcal{Q}$.

[33] For further information regarding linear differential equations, refer to sources such as [53].

where

$$\exp(\boldsymbol{A}t) = \sum_{s=0}^{+\infty} \frac{(\boldsymbol{A}t)^s}{s!} = \boldsymbol{I}_n + \boldsymbol{A}t + \frac{(\boldsymbol{A}t)^2}{2!} + \frac{(\boldsymbol{A}t)^3}{3!} + \frac{(\boldsymbol{A}t)^4}{4!} + \cdots \tag{A.11}$$

as per the Taylor expansion (see Definition A.2).

Definition A.1. Let $f(x) : \mathtt{I} \mapsto \mathbb{R}$ be an s-times differentiable function at $x_0 \in \mathtt{I}$. The s-th degree polynomial of the form

$$\begin{aligned} T_s(x) \coloneqq f(x_0) + f'(x_0)(x - x_0) + \frac{f''(x_0)}{2!}(x - x_0)^2 \\ + \cdots + \frac{f^{(s)}(x_0)}{s!}(x - x_0)^s, \end{aligned}$$

where

$$f^{(s)}(x_0) \equiv \left.\frac{\mathrm{d}^s f(x)}{\mathrm{d}x^s}\right|_{x=x_0}$$

is referred to as the s-th degree *Taylor polynomial* of $f(x)$ at x_0. [54, 55] ◇

Theorem A.2. Let $s \in \mathbb{N}$, let $f(x) : \mathtt{I} \mapsto \mathbb{R}$. Suppose $f(x)$ is infinitely differentiable at $x_0 \in \mathtt{I}$ and $\mathrm{d}^{s+1} f(x)/\mathrm{d}x^{s+1}$ exists on $\mathtt{I}$. For any $x \in \mathtt{I}$, there exists a point $\mathfrak{X}$ lying between x and x_0 such that

$$\begin{aligned} f(x) = f(x_0) + f'(x_0)(x - x_0) + \frac{f''(x_0)}{2!}(x - x_0)^2 \\ + \cdots + \frac{f^{(s)}(x_0)}{s!}(x - x_0)^s + \frac{f^{(s+1)}(\mathfrak{X})}{(s+1)!}(x - x_0)^{s+1}, \end{aligned} \tag{A.12}$$

where

$$f^{(s)}(x_0) \equiv \left.\frac{\mathrm{d}^s f(x)}{\mathrm{d}x^s}\right|_{x=x_0} \qquad f^{(s+1)}(\mathfrak{X}) \equiv \left.\frac{\mathrm{d}^{s+1} f(x)}{\mathrm{d}x^{s+1}}\right|_{x=\mathfrak{X}}$$

and also

$$R_s(x) = \frac{f^{(s+1)}(\mathfrak{X})}{(s+1)!}(x - x_0)^{s+1} \tag{A.13}$$

is the *Lagrange remainder*, henceforth Eq. (A.12) is referred to as the *Taylor polynomial* of degree s with the Lagrange form of the remainder. [50, 54, 55] ◇

Definition A.2. Let $s \in \mathbb{N}$, let $f(x) : \mathtt{I} \mapsto \mathbb{R}$ be infinitely differentiable at $x_0 \in \mathtt{I}$. If and only if the power series given by

$$f(x) = \sum_{s=0}^{+\infty} \frac{f^{(s)}(x_0)}{s!}(x - x_0)^s, \quad f^{(s)}(x_0) \equiv \left.\frac{\mathrm{d}^s f(x)}{\mathrm{d}x^s}\right|_{x=x_0} \tag{A.14}$$

has its sequence of remainders $R_s(x)$ [Eq. (A.13)] converging to zero for all $x \in \mathtt{I}' \subseteq \mathtt{I}$ as s approaches infinity, i.e.,

$$\lim_{s\to+\infty} R_s(x) = 0 \quad \forall x \in \mathtt{I}',$$

we refer to it as the *Taylor expansion* of $f(x)$ at x_0. [54, 55] ◇

Now, if we let $f(x) = \exp x$ and $x_0 = 0$, Eq. (A.14) yields the famous Maclaurin series which reads

$$\exp x = \sum_{s=0}^{+\infty} \frac{x^s}{s!} = 1 + x + \frac{x^2}{2!} + \frac{x^3}{3!} + \frac{x^4}{4!} + \cdots \tag{A.15}$$

since the derivative of $\exp x$ is always the function itself. Further, if the argument of the function one wishes to expand is of vector, matrix, or higher-order tensor form, the Taylor expansion holds the same as defined in (A.14). So, for (A.15), we have precisely (A.11) when we input $\boldsymbol{A}t$ instead of x as an argument.

Remark A.2. Other important Maclaurin series of common functions are

$$\sin x = \sum_{s=0}^{+\infty} (-1)^s \frac{x^{2s+1}}{(2s+1)!} = x - \frac{x^3}{3!} + \frac{x^5}{5!} - \frac{x^7}{7!} + \cdots, \tag{A.16}$$

$$\cos x = \sum_{s=0}^{+\infty} (-1)^s \frac{x^{2s}}{(2s)!} = 1 - \frac{x^2}{2!} + \frac{x^4}{4!} - \frac{x^6}{6!} + \cdots, \tag{A.17}$$

from where it is clear to see that whilst $\sin x$ is *odd*, $\cos x$ is *even*. ◇

A.2.2 Exponential Coordinate Representation on $SO(3)$

The *exponential coordinate* representation on $SO(3)$ is very similar to the axis-angle representation, i.e., it takes the unit vector of the axis of rotation $\boldsymbol{a} \in \mathbb{R}^3$ and the angle of revolution about said axis φ as parameters. Whilst the axis-angle representation is a four-parameter one, expressing rotation using exponential coordinates takes only three parameters, namely the vector $\boldsymbol{e} = \boldsymbol{a}\varphi \in \mathbb{R}^3$. The rotation can be conceptualized as a vector $\boldsymbol{p}_{iP}(t) \in \mathbb{R}^3$, representing the position vector of point $P \in \mathbb{E}^3$ in the frame $O_i x_i y_i z_i$, undergoing constant rotation at a rate of 1 $[\mathrm{rad\,s^{-1}}]$ around $\boldsymbol{a}$ from $t = 0$ to $t = \varphi$. The velocity of the endpoint of $\boldsymbol{p}_{iP}(t)$ can be articulated as

$$\dot{\boldsymbol{p}}_{iP}(t) = \boldsymbol{a} \times \boldsymbol{p}_{iP}(t) = [\boldsymbol{A}]\, \boldsymbol{p}_{iP}(t), \quad \boldsymbol{p}_{iP}(0) = \boldsymbol{p}_{iP_0},$$

which is a linear differential equation of the form (A.10), hence its solution is[34]

$$\boldsymbol{p}_{iP}(t) = \exp([\boldsymbol{A}]t)\boldsymbol{p}_{iP_0} \xrightarrow{t\equiv\varphi} \boldsymbol{p}_{iP}(\varphi) = \exp([\boldsymbol{A}]\varphi)\boldsymbol{p}_{iP_0}$$

[34] Note that, in this context, time t is equivalent to the angle of revolution φ.

and can be expanded to the form

$$\exp([\boldsymbol{A}]\varphi) = \boldsymbol{I}_3 + [\boldsymbol{A}]\varphi + \frac{([\boldsymbol{A}]\varphi)^2}{2!} + \frac{([\boldsymbol{A}]\varphi)^3}{3!} + \frac{([\boldsymbol{A}]\varphi)^4}{4!} + \cdots,$$

where if we utilize $[\boldsymbol{A}]^3 = -[\boldsymbol{A}]$, $[\boldsymbol{A}]^4 = -[\boldsymbol{A}]^2$, etc., we obtain

$$\begin{aligned}\exp([\boldsymbol{U}]\varphi) &= \boldsymbol{I}_3 + \underbrace{\left(\varphi - \frac{\varphi^3}{3!} + \frac{\varphi^5}{5!} - \cdots\right)}_{\text{Maclaruin series for } \sin\varphi}[\boldsymbol{A}] + \underbrace{\left(\frac{\varphi^2}{2!} - \frac{\varphi^4}{4!} + \frac{\varphi^6}{6!} - \cdots\right)}_{\text{“Maclaruin series” for } 1-\cos\varphi}[\boldsymbol{A}]^2 \\ &= \boldsymbol{I}_3 + \sin\varphi\,[\boldsymbol{A}] + (1-\cos\varphi)\,[\boldsymbol{A}]^2 = \boldsymbol{R}_{ij}(\mathfrak{a},\varphi) \in SO(3),\end{aligned}$$

known as the *Rodrigues formula for rotations*.[35] When expressed in component form, one gets the expression

$$r_{ij} = \delta_{ij} + (1-\cos\varphi)a_i a_j - \sin\varphi\epsilon_{ijk}a_k,$$

i.e., Eq. (A.2), where δ_{ij} is the Kronecker delta and ϵ_{ijk} is the Levi-Civita symbol as discussed previously.[36] [12, 19]

A.2.3 Cayley-Rodrigues Representation

The Cayley-Rodrigues parameters can be directly obtained from the exponential coordinate representation on $SO(3)$, i.e., given said representation in the form $\boldsymbol{R}_{ij}(\mathfrak{a},\varphi) = \exp([\boldsymbol{A}]\varphi)$, the parameters read [12]

$$\mathcal{C} = \boldsymbol{a}\tan(\varphi/2) = \begin{bmatrix} c_1 & c_2 & c_3 \end{bmatrix}^T \in \mathbb{R}^3.$$

Given a Cayley-Rodrigues representation $\mathcal{C}$, the rotation matrix corresponding to this rotation can be obtained using

$$\boldsymbol{R}_{ij}(\mathcal{C}) = \frac{(1-\langle\mathcal{C},\mathcal{C}\rangle)\boldsymbol{I}_3 + 2\langle\mathcal{C},\mathcal{C}\rangle + 2\,[\boldsymbol{C}]}{1+\langle\mathcal{C},\mathcal{C}\rangle},$$

where $[\boldsymbol{C}] \in \mathbb{R}^{3\times3}$ is a skew-symmetric matrix derived from the Cayley-Rodrigues representation (more precisely from its components c_1, c_2, c_3). [12]

On the other hand, the inverse solution reads

$$[\boldsymbol{C}] = \frac{\boldsymbol{R}_{ij} - \boldsymbol{R}_{ij}^T}{1+\operatorname{tr}(\boldsymbol{R}_{ij})}, \quad \operatorname{tr}(\boldsymbol{R}_{ij}) \neq -1,$$

for a generic rotation matrix $\boldsymbol{R}_{ij} \in SO(3)$ of the form (1.3). [12]

[35] We have already used this formula prior to its derivation [see Eq. (A.3)].

[36] For formal definitions of the Kronecker delta and the Levi-Civita symbol, see p. 157.

At last, let us discuss the computation of the product of two Cayley-Rodrigues representations $\mathcal{C}$, $\mathcal{C}^\star$, which correspond to rotation matrices $\boldsymbol{R}_{ij}$, $\boldsymbol{R}_{jk}$, respectively. The appeal of the Cayley-Rodrigues representation primarily lies in how straightforwardly the product can be obtained. It reads

$$\mathcal{C}' = \mathcal{C}\mathcal{C}^\star = \frac{\mathcal{C} + \mathcal{C}^\star + (\mathcal{C} \times \mathcal{C}^\star)}{1 - \langle \mathcal{C}, \mathcal{C}^\star \rangle} = \begin{bmatrix} c'_1 & c'_2 & c'_3 \end{bmatrix}^T \in \mathbb{R}^3,$$

where, in cases when $\langle \mathcal{C}, \mathcal{C}^\star \rangle = 1$, an alternative formula is necessary. Define

$$\boldsymbol{\gamma} = \frac{\mathcal{C}}{\sqrt{1 + \langle \mathcal{C}, \mathcal{C} \rangle}} \in \mathbb{R}^3, \quad \|\boldsymbol{\gamma}\| = \sin(\varphi/2),$$

in order to express the rotation matrix in the form

$$\boldsymbol{R}_{ij} = \boldsymbol{I}_3 + 2\sqrt{1 - \langle \boldsymbol{\gamma}, \boldsymbol{\gamma} \rangle}\, [\boldsymbol{\Gamma}] + 2\,[\boldsymbol{\Gamma}]^2\,,$$

where $[\boldsymbol{\Gamma}] \in \mathbb{R}^{3\times 3}$ is a skew-symmetric matrix derived from the components of $\boldsymbol{\gamma}$. Utilizing $\boldsymbol{\gamma}$, the product of $\mathcal{C}$ and $\mathcal{C}^\star$ is expressed as

$$\boldsymbol{\gamma}' = \boldsymbol{\gamma}\sqrt{1 - \langle \boldsymbol{\gamma}^\star, \boldsymbol{\gamma}^\star \rangle} + \boldsymbol{\gamma}^\star\sqrt{1 - \langle \boldsymbol{\gamma}, \boldsymbol{\gamma} \rangle} + (\boldsymbol{\gamma} \times \boldsymbol{\gamma}^\star) \in \mathbb{R}^3,$$

with the direction of $\boldsymbol{\gamma}'$ being coincident with that of $\mathcal{C}'$. [12]

B

Iterative Inverse Kinematics

In in many cases, the inverse kinematics problem does not have a closed-form solution, especially for complex robotic systems with multiple degrees of freedom. To achieve the desired pose of the end-effector, engineers often result to iterative methods, which provide an efficient approach to approximate the solution.

Within the framework of iterative inverse kinematics, the algorithm continually adjusts the joint parameters to minimize the disparity between the desired and current robot poses. This iterative refinement process typically entails gradually refining the joint angles until the error falls below a specified threshold or until a predetermined number of iterations is reached. Despite their computational overhead, iterative methods offer adaptability and resilience, making them suitable for real-time applications and dynamic environments. This Appendix is intended to present common iterative inverse kinematics (IIK) methods as a complement to the content discussed in Section 1.3.2 (Chapter 1).

B.1 Jacobian-Based Inverse Kinematics

Jacobian-based methods leverage the concept of the Jacobian matrix, which was introduced earlier to close out open-chain kinematics. It encapsulates the relationship between changes in joint variables and corresponding changes in the position and orientation of the tool center point, serving as a mathematical bridge and enabling the translation of desired tool center point poses into adjustments in joint angles. Despite their effectiveness, Jacobian-based methods may encounter challenges such as singularities or local minima, which require careful consideration and mitigation strategies. Nevertheless, their versatility and efficiency make them indispensable tools in robotics for achieving accurate motion.

B.1.1 Analytical Jacobian

Subsection 1.3.3 introduced one of two possible forms of the Jacobian matrix, i.e., the geometric Jacobian $\boldsymbol{J}_g(\boldsymbol{\varphi}) \in \mathbb{R}^{6\times\mathscr{D}}$, mapping the tool center point linear and angular velocities to the joint coordinates of a $\mathscr{D}$-degree-of-freedom arm. A different form to obtain such mapping is to compute the *analytical Jacobian* $\boldsymbol{J}_a(\boldsymbol{\varphi}) \in \mathbb{R}^{6\times\mathscr{D}}$ by simply differentiating the forward kinematics solution.

The contribution from linear velocity is computed in similar fashion to (1.24),

$$\dot{\boldsymbol{p}}_{1\mathrm{TCP}}(\boldsymbol{\varphi},\dot{\boldsymbol{\varphi}}) = \frac{\partial \boldsymbol{p}_{1\mathrm{TCP}}(\boldsymbol{\varphi})}{\partial \boldsymbol{\varphi}}\dot{\boldsymbol{\varphi}} \mapsto \boldsymbol{J}'_p(\boldsymbol{\varphi})\dot{\boldsymbol{\varphi}},$$

whilst the contribution from angular velocity is simply

$$\dot{\boldsymbol{\phi}}_{\mathrm{TCP}}(\boldsymbol{\varphi},\dot{\boldsymbol{\varphi}}) = \frac{\partial \boldsymbol{\phi}_{\mathrm{TCP}}(\boldsymbol{\varphi})}{\partial \boldsymbol{\varphi}}\dot{\boldsymbol{\varphi}} \mapsto \boldsymbol{J}'_\phi(\boldsymbol{\varphi})\dot{\boldsymbol{\varphi}},$$

however the computation of $\partial\boldsymbol{\phi}_{\mathrm{TCP}}(\boldsymbol{\varphi})/\partial\boldsymbol{\varphi}$ is not always effortless since the function $\boldsymbol{\phi}_{\mathrm{TCP}}(\boldsymbol{\varphi})$ is not generally available directly but requires computation from the corresponding rotation matrix. [12, 15, 29]

The analytical Jacobian is, similarly to (1.21), given by the mapping

$$\dot{\boldsymbol{\varepsilon}}(\boldsymbol{\varphi},\dot{\boldsymbol{\varphi}}) \coloneqq \begin{bmatrix}\dot{\boldsymbol{p}}_{1\mathrm{TCP}}(\boldsymbol{\varphi},\dot{\boldsymbol{\varphi}}) & \dot{\boldsymbol{\phi}}_{\mathrm{TCP}}(\boldsymbol{\varphi},\dot{\boldsymbol{\varphi}})\end{bmatrix}^T \mapsto \boldsymbol{J}_a(\boldsymbol{\varphi})\dot{\boldsymbol{\varphi}},$$

i.e.,

$$\boldsymbol{J}_a(\boldsymbol{\varphi}) \coloneqq \begin{bmatrix}\boldsymbol{J}'_p(\boldsymbol{\varphi}) \in \mathbb{R}^{3\times\mathscr{D}} \\ \boldsymbol{J}'_\phi(\boldsymbol{\varphi}) \in \mathbb{R}^{3\times\mathscr{D}}\end{bmatrix} \in \mathbb{R}^{6\times\mathscr{D}},$$

but is *not* the same as the geometric Jacobian as generally $\boldsymbol{\omega}_{1\mathrm{TCP}} \neq \dot{\boldsymbol{\phi}}_{\mathrm{TCP}}$. [15]

To find a correlation between $\dot{\boldsymbol{\phi}}_{\mathrm{TCP}}$ and $\boldsymbol{\omega}_{1\mathrm{TCP}}$, let us consider a set of Euler XYZ angles. In order to obtain this transformation, we need to account for contributions of each of the rotational velocities to the angular velocity with respect to the global coordinate system, i.e.:

(i) **Yaw.** The first rotation (Yaw) contributes to $\boldsymbol{\omega}_{1\mathrm{TCP}}$ by $\dot{\gamma}[1 \quad 0 \quad 0]^T$,

(ii) **Pitch.** The second rotation (Pitch) contributes to $\boldsymbol{\omega}_{1\mathrm{TCP}}$ by $\dot{\beta}[0 \quad -\mathrm{s}_\gamma \quad \mathrm{c}_\gamma]^T$,

(iii) **Roll.** The third rotation (Roll) contributes to $\boldsymbol{\omega}_{1\mathrm{TCP}}$ by $\dot{\alpha}[\mathrm{c}_\beta \quad -\mathrm{s}_\beta\mathrm{s}_\gamma \quad \mathrm{c}_\gamma]^T$.

Upon computing all contributions from rotational velocities to the angular velocity, $\boldsymbol{\omega}_{1\mathrm{TCP}}$ can be related to $\dot{\boldsymbol{\phi}}_{\mathrm{TCP}}$ by the expression

$$\boldsymbol{\omega}_{1\mathrm{TCP}}(\boldsymbol{\varphi},\dot{\boldsymbol{\varphi}}) = \boldsymbol{\Omega}_{1\mathrm{TCP}}(\boldsymbol{\phi}_{\mathrm{TCP}})\dot{\boldsymbol{\phi}}_{\mathrm{TCP}},$$

where, in the case of the Euler XYZ angles,

$$\boldsymbol{\Omega}_{1\mathrm{TCP}}(\boldsymbol{\phi}_{\mathrm{TCP}}) = \begin{bmatrix} 1 & 0 & \cos\beta \\ 0 & -\sin\gamma & -\sin\beta\sin\gamma \\ 0 & \cos\gamma & \cos\gamma \end{bmatrix} \in \mathbb{R}^{3\times 3},$$

is nonsingular for all $\sin\gamma \neq 0$ and $\tan\beta \neq \tan\gamma$. [15, 19, 29]

Provided the transformation $\boldsymbol{\Omega}_{1\mathrm{TCP}}(\boldsymbol{\phi}_{\mathrm{TCP}})$ to be known, we are able to compute the geometric Jacobian from the analytical Jacobian as

$$\boldsymbol{\nu}_{\mathrm{TCP}}(\boldsymbol{\varphi}, \dot{\boldsymbol{\varphi}}) = \begin{bmatrix} \boldsymbol{I}_3 & \boldsymbol{O}_3 \\ \mathrm{sym.} & \boldsymbol{\Omega}_{1\mathrm{TCP}}(\boldsymbol{\phi}_{\mathrm{TCP}}) \end{bmatrix} \dot{\boldsymbol{\varepsilon}}(\boldsymbol{\varphi}, \dot{\boldsymbol{\varphi}}) = \boldsymbol{\Upsilon}_a(\boldsymbol{\phi}_{\mathrm{TCP}})\dot{\boldsymbol{\varepsilon}}(\boldsymbol{\varphi}, \dot{\boldsymbol{\varphi}}),$$

yielding

$$\boldsymbol{J}_g(\boldsymbol{\varphi}) = \boldsymbol{\Upsilon}_a(\boldsymbol{\phi}_{\mathrm{TCP}})\boldsymbol{J}_a(\boldsymbol{\varphi}),$$

where $\boldsymbol{O}_3 \in \mathbb{R}^{3\times 3}$ denotes the *null matrix*. [15, 27, 29]

Remark B.1. The geometric and analytical Jacobians are distinct representations of the motion of the tool center point within robotic systems. While the geometric Jacobian relates the velocity of the TCP coordinate system in the space of joint coordinates (configuration space), the analytical Jacobian pertains to the robot's *operational space*. This operational space denotes the domain within which the TCP operates, prioritizing a task-centric portrayal of the robot's motion. It focuses on delineating the precise spatial configurations and procedures employed by the tool center point to accomplish its designated tasks. [12, 15] ◇

B.1.2 Jacobian (Pseudo-)Inverse IIK

Suppose we have the mapping (1.21). Then, the joint velocities can by obtained by inversion of the geometric Jacobian, i.e.,

$$\dot{\boldsymbol{\varphi}} = \boldsymbol{J}_g^{-1}(\boldsymbol{\varphi})\boldsymbol{\nu}_{\mathrm{TCP}}(\boldsymbol{\varphi}, \dot{\boldsymbol{\varphi}}), \tag{B.1}$$

provided $\boldsymbol{J}_g(\boldsymbol{\varphi})$ is *square* and has *full rank*. Moreover, suppose we are provided with an initial condition $\boldsymbol{\varphi}(0) \equiv \boldsymbol{\varphi}_0$. By these measures, we can obtain the joint positions by integrating (B.1) over time, i.e.,

$$\boldsymbol{\varphi}(t) = \int_0^t \dot{\boldsymbol{\varphi}}(\varpi)\,\mathrm{d}\varpi + \boldsymbol{\varphi}_0,$$

and by resorting to time discretization with step size Δt. The simplest numerical technique we can employ is the Euler integration method, which reads

$$\boldsymbol{\varphi}(t_{k+1}) = \boldsymbol{\varphi}(t_k) + \dot{\boldsymbol{\varphi}}(t_k)\Delta t,$$

ultimately resulting in

$$\boldsymbol{\varphi}(t_{k+1}) = \boldsymbol{\varphi}(t_k) + \boldsymbol{J}_g^{-1}[\boldsymbol{\varphi}(t_k)]\boldsymbol{\nu}_{\mathrm{TCP}}(t_k)\Delta t,$$

where $t_{k+1} = t_k + \Delta t$ for all $k \in \mathbb{N}_0$. [15, 51, 56, 57, 58]

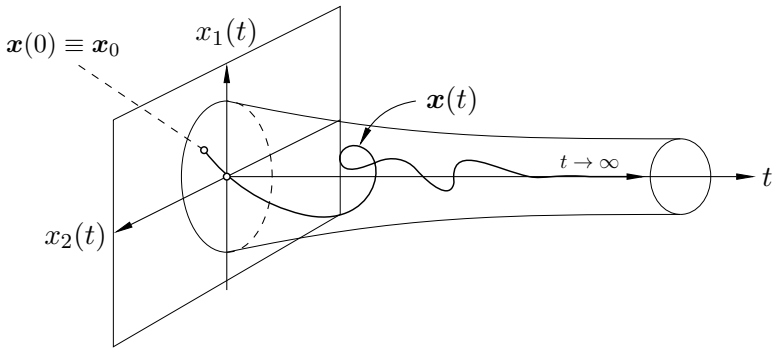


Figure B.1 Asymptotically stable two-variable system.

Let $\boldsymbol{e}_{\mathrm{TCP}}(\boldsymbol{\varphi}) \in \mathbb{R}^6$ denote the *operational space* error vector between the desired and current tool center point poses, denoted $\boldsymbol{\varepsilon} \in \mathbb{R}^6$ and $\boldsymbol{\varepsilon}_c(\boldsymbol{\varphi}) \in \mathbb{R}^6$, respectively. Mathematically, we can express such vector as

$$\boldsymbol{e}_{\mathrm{TCP}}(\boldsymbol{\varphi}) = \boldsymbol{\varepsilon} - \boldsymbol{\varepsilon}_c(\boldsymbol{\varphi}). \tag{B.2}$$

Differentiating both sides of (B.2) with respect to time yields

$$\dot{\boldsymbol{e}}_{\mathrm{TCP}}(\boldsymbol{\varphi}, \dot{\boldsymbol{\varphi}}) = \dot{\boldsymbol{\varepsilon}} - \dot{\boldsymbol{\varepsilon}}_c(\boldsymbol{\varphi}, \dot{\boldsymbol{\varphi}}), \tag{B.3}$$

which can be rewritten to the form

$$\dot{\boldsymbol{e}}_{\mathrm{TCP}}(\boldsymbol{\varphi}, \dot{\boldsymbol{\varphi}}) = \dot{\boldsymbol{\varepsilon}} - \boldsymbol{J}_a(\boldsymbol{\varphi})\dot{\boldsymbol{\varphi}}, \tag{B.4}$$

giving us the rate at which this error changes in time. Assuming $\boldsymbol{J}_a(\boldsymbol{\varphi})$ is square and has full rank, i.e., is nonsingular, choosing

$$\dot{\boldsymbol{\varphi}} = \boldsymbol{J}_a^{-1}(\boldsymbol{\varphi})[\dot{\boldsymbol{\varepsilon}} + \boldsymbol{E}\boldsymbol{e}_{\mathrm{TCP}}(\boldsymbol{\varphi})] \tag{B.5}$$

leads to an equivalent linear system

$$\dot{\boldsymbol{e}}_{\mathrm{TCP}}(\boldsymbol{\varphi}, \dot{\boldsymbol{\varphi}}) + \boldsymbol{E}\boldsymbol{e}_{\mathrm{TCP}}(\boldsymbol{\varphi}) = \boldsymbol{0}_6, \tag{B.6}$$

wherein, if $\boldsymbol{E}$ is positive definite, i.e., is symmetric and all its eigenvalues λ are *strictly* positive, (B.6) is asymptotically stable (Fig. B.1).[37] The error tends toward

[37] For deeper understanding of (nonlinear) dynamical systems and chaos theory, the author encourages readers to refer to works such as [39], [44], or [59].

zero with a convergence rate determined by the eigenvalues of $\boldsymbol{E}$. Specifically, with larger eigenvalues corresponding to faster convergence. [15, 19, 32, 51]

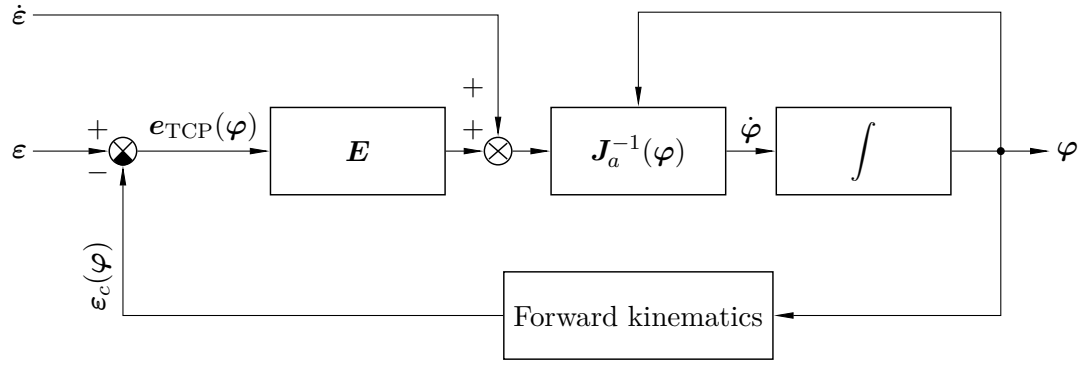


Figure B.2 Jacobian inverse IIK algorithm. Figure adapted from [15].

If $\boldsymbol{J}_a^{-1}(\boldsymbol{\varphi})$ does not exist, either because the analytic Jacobian is rectangular or rank deficient, (B.5) is unusable. The rectangularity of the Jacobian may stem from the manipulator being *kinematically redundant*, i.e., $\mathscr{D} > 6$ and thereby there exists $(\mathscr{D} - 6)$ redundant degrees of freedom, equivalently expressed as the manipulator being $(\mathscr{D} - 6)$-times redundant. In such instances, it becomes necessary to resort to the expression

$$\dot{\boldsymbol{\varphi}} = \boldsymbol{J}_a^{\dagger}(\boldsymbol{\varphi})[\dot{\boldsymbol{\varepsilon}} + \boldsymbol{E}\boldsymbol{e}_{\mathrm{TCP}}(\boldsymbol{\varphi})] + [\boldsymbol{I}_{\mathscr{D}} - \boldsymbol{J}_a^{\dagger}(\boldsymbol{\varphi})\boldsymbol{J}_a(\boldsymbol{\varphi})]\dot{\boldsymbol{\varphi}}^*,$$

instead of (B.5), where

$$\boldsymbol{J}_a^{\dagger}(\boldsymbol{\varphi}) \coloneqq \boldsymbol{J}_a^T(\boldsymbol{\varphi})[\boldsymbol{J}_a(\boldsymbol{\varphi})\boldsymbol{J}_a^T(\boldsymbol{\varphi})]^{-1} \in \mathbb{R}^{\mathscr{D}\times 6}$$

is termed the Moore-Penrose *right pseudo-inverse* of $\boldsymbol{J}_a(\boldsymbol{\varphi}) \in \mathbb{R}^{6\times\mathscr{D}}$ and $\dot{\boldsymbol{\varphi}}^* \in \mathbb{R}^J$ is a vector of arbitrary joint velocities. [15]

B.1.3 Jacobian Transpose IIK

Developing a computationally simpler algorithm than the Jacobian inverse approach entails deriving the relationship between $\dot{\boldsymbol{\varphi}}$ and $\boldsymbol{e}_{\mathrm{TCP}}(\boldsymbol{\varphi})$ while maintaining error convergence to zero and avoiding the need for linearization of (B.4). Finding such relationship necessitates a brief exploration in stability theory as it is established using the *Lyapunov direct method*. [15]

▷ **Lyapunov Direct Method.** An intuitive approach to define stability involves associating an energy-based description with an autonomous system, if possible. Under this framework, if the rate of change of this energy is *negative* for each system state except the equilibrium state, then the energy decreases along any system trajectory until it reaches its minimum at the equilibrium state. [15]

In our case, this equilibrium state is at $\boldsymbol{e}_{\mathrm{TCP}}(\boldsymbol{\varphi}) = \mathbf{0}_6$, i.e., when the error between the desired and current TCP poses is zero. A scalar function $\mathscr{L}(\boldsymbol{e}_{\mathrm{TCP}})$ is a *Lyapunov function* if it satisfies the following properties:

(i) $\mathscr{L}(\boldsymbol{e}_{\mathrm{TCP}})$ and $\dot{\mathscr{L}}(\boldsymbol{e}_{\mathrm{TCP}})$ are continuous.

(ii) $\mathscr{L}(\boldsymbol{e}_{\mathrm{TCP}})$ is positive definite, i.e., $\mathscr{L}(\boldsymbol{e}_{\mathrm{TCP}}) > 0$, for all $\boldsymbol{e}_{\mathrm{TCP}}(\boldsymbol{\varphi}) \neq \mathbf{0}_6$, whereas $\mathscr{L}(\boldsymbol{e}_{\mathrm{TCP}}) = 0$ at the equilibrium state $\boldsymbol{e}_{\mathrm{TCP}}(\boldsymbol{\varphi}) = \mathbf{0}_6$.

(iii) $\dot{\mathscr{L}}(\boldsymbol{e}_{\mathrm{TCP}})$ is negative definite, i.e., $\dot{\mathscr{L}}(\boldsymbol{e}_{\mathrm{TCP}}) < 0$, for all $\boldsymbol{e}_{\mathrm{TCP}}(\boldsymbol{\varphi}) \neq \mathbf{0}_6$.

(iv) As $\|\boldsymbol{e}_{\mathrm{TCP}}(\boldsymbol{\varphi})\| \to +\infty$, $\mathscr{L}(\boldsymbol{e}_{\mathrm{TCP}}) \to +\infty$.

The existence of such function guarantees global asymptotic stability. In the case where $\dot{\mathscr{L}}(\boldsymbol{e}_{\mathrm{TCP}})$ is only negative semi-definite, i.e., $\dot{\mathscr{L}}(\boldsymbol{e}_{\mathrm{TCP}}) \leq 0$, global asymptotic stability is ensured if and only if $\dot{\mathscr{L}}(\boldsymbol{e}_{\mathrm{TCP}}) \equiv 0$ exclusively along the equilibrium trajectory $\boldsymbol{e}_{\mathrm{TCP}}(\boldsymbol{\varphi}) \equiv \mathbf{0}_6$. [15, 39, 59, 60]

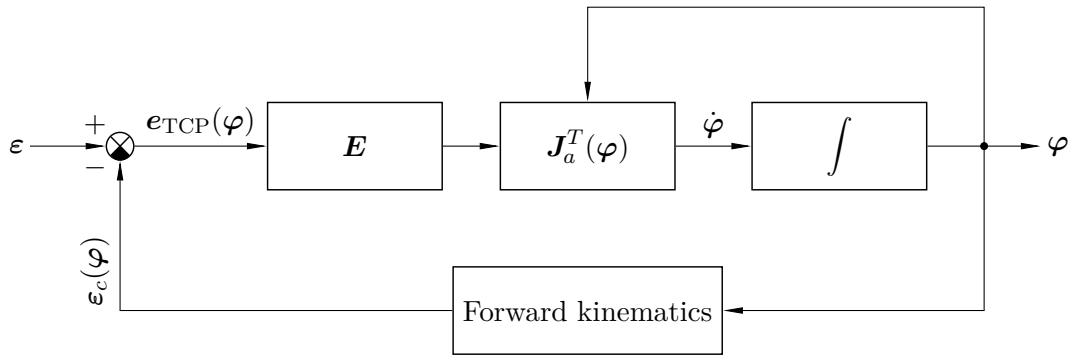


Figure B.3 Jacobian transpose IIK algorithm. Figure adapted from [15].

The Lyapunov function candidate for determining $\dot{\boldsymbol{\varphi}}(\boldsymbol{e}_{\mathrm{TCP}})$ is

$$\mathscr{L}(\boldsymbol{e}_{\mathrm{TCP}}) = \frac{1}{2}\boldsymbol{e}_{\mathrm{TCP}}^T(\boldsymbol{\varphi})\boldsymbol{E}\boldsymbol{e}_{\mathrm{TCP}}(\boldsymbol{\varphi}),$$

where $\boldsymbol{E}$ is positive definite. Differentiating with respect to time yields

$$\dot{\mathscr{L}}(\boldsymbol{e}_{\mathrm{TCP}}) = \frac{1}{2}\left[\dot{\boldsymbol{e}}_{\mathrm{TCP}}^T(\boldsymbol{\varphi},\dot{\boldsymbol{\varphi}})\boldsymbol{E}\boldsymbol{e}_{\mathrm{TCP}}(\boldsymbol{\varphi}) + \boldsymbol{e}_{\mathrm{TCP}}^T(\boldsymbol{\varphi})\boldsymbol{E}\dot{\boldsymbol{e}}_{\mathrm{TCP}}(\boldsymbol{\varphi},\dot{\boldsymbol{\varphi}})\right].$$

Moreover, since $\boldsymbol{E}$ is a constant matrix, we can write

$$\dot{\mathscr{L}}(\boldsymbol{e}_{\mathrm{TCP}}) = \frac{1}{2}\left[\boldsymbol{e}_{\mathrm{TCP}}^T(\boldsymbol{\varphi})\boldsymbol{E}^T\dot{\boldsymbol{e}}_{\mathrm{TCP}}(\boldsymbol{\varphi},\dot{\boldsymbol{\varphi}}) + \boldsymbol{e}_{\mathrm{TCP}}^T(\boldsymbol{\varphi})\boldsymbol{E}\dot{\boldsymbol{e}}_{\mathrm{TCP}}(\boldsymbol{\varphi},\dot{\boldsymbol{\varphi}})\right],$$

which can be further simplified to the form

$$\dot{\mathscr{L}}(\boldsymbol{e}_{\mathrm{TCP}}) = \boldsymbol{e}_{\mathrm{TCP}}^T(\boldsymbol{\varphi})\boldsymbol{E}\dot{\boldsymbol{e}}_{\mathrm{TCP}}(\boldsymbol{\varphi},\dot{\boldsymbol{\varphi}})$$

as $\boldsymbol{E}$ is symmetric. Accounting for (B.3) and (B.4), we get

$$\dot{\mathscr{L}}(\boldsymbol{e}_{\mathrm{TCP}}) = \boldsymbol{e}_{\mathrm{TCP}}^T(\boldsymbol{\varphi})\boldsymbol{E}\dot{\boldsymbol{\varepsilon}} - \boldsymbol{e}_{\mathrm{TCP}}^T(\boldsymbol{\varphi})\boldsymbol{E}\boldsymbol{J}_a(\boldsymbol{\varphi})\dot{\boldsymbol{\varphi}},$$

and finally, choosing

$$\dot{\boldsymbol{\varphi}} = \boldsymbol{J}_a^T(\boldsymbol{\varphi})\boldsymbol{E}\boldsymbol{e}_{\mathrm{TCP}}(\boldsymbol{\varphi}),$$

results in the expression

$$\dot{\mathscr{L}}(\boldsymbol{e}_{\mathrm{TCP}}) = \boldsymbol{e}_{\mathrm{TCP}}^T(\boldsymbol{\varphi})\boldsymbol{E}\dot{\boldsymbol{\varepsilon}} - \boldsymbol{e}_{\mathrm{TCP}}^T(\boldsymbol{\varphi})\boldsymbol{E}\boldsymbol{J}_a(\boldsymbol{\varphi})\boldsymbol{J}_a^T(\boldsymbol{\varphi})\boldsymbol{E}\boldsymbol{e}_{\mathrm{TCP}}(\boldsymbol{\varphi}), \tag{B.7}$$

which is negative definite for a constant reference, i.e., $\dot{\boldsymbol{\varepsilon}} = \boldsymbol{0}_6$, and under the assumption of $\boldsymbol{J}_a(\boldsymbol{\varphi})$ being full rank. Since $\mathscr{L}(\boldsymbol{e}_{\mathrm{TCP}})$ is originally positive definite, the system is asymptotically stable and the error between the desired and current poses of the tool center point converges to zero. On the other hand, (B.7) is only negative semi-definite in cases when

$$\ker[\boldsymbol{J}_a^T(\boldsymbol{\varphi})] \neq \emptyset,$$

where $\ker[\boldsymbol{J}_a^T(\boldsymbol{\varphi})]$ is the kernel of the analytical Jacobian, i.e., the subspace of joint velocities that do not generate any tool center point velocity in the current manipulator configuration, and $\emptyset$ denotes the empty set. In such scenarios, the algorithm may encounter a standstill at $\dot{\boldsymbol{\varphi}} = \boldsymbol{0}_J$ with $\boldsymbol{e}_{\mathrm{TCP}}(\boldsymbol{\varphi}) \neq \boldsymbol{0}_6$. However, it can be shown this circumstance occurs exclusively when the designated TCP pose cannot be reached from the current configuration. [15, 26, 29, 32]

B.2 Gradient-Based Inverse Kinematics

Gradient-based methods are optimization techniques that hinge on the gradient, or the first derivative, of an objective function relative to the variables targeted for optimization. These techniques progressively adjust these variables in a manner that diminishes or enhances the objective function, guided by the gradient's direction. By computing the gradient of the discrepancy between the desired and current tool center point poses concerning the joint angles, these methods iteratively refine the joint angles until they converge to a solution.

B.2.1 First-Order Newton Method IIK

Let $\boldsymbol{f}(\boldsymbol{\varphi}) : \mathbb{R}^J \mapsto \mathbb{R}^6$ be a differentiable forward kinematics solution of the form (1.13'), mapping the current joint coordinates vector $\boldsymbol{\varphi} \in \mathbb{R}^J$ to the tool center point configuration vector $\boldsymbol{\varepsilon}(\boldsymbol{\varphi}_c) \in \mathbb{R}^6$. Define $\boldsymbol{h}(\boldsymbol{\varphi}) : \mathbb{R}^J \mapsto \mathbb{R}^6$ as

$$\boldsymbol{h}(\boldsymbol{\varphi}) \coloneqq \boldsymbol{\varepsilon} - \boldsymbol{f}(\boldsymbol{\varphi}) : \mathbb{R}^J \mapsto \mathbb{R}^6,$$

in terms of the desired TCP configuration $\boldsymbol{\varepsilon} \in \mathbb{R}^6$, and solve for

$$\boldsymbol{h}(\boldsymbol{\varphi}) \coloneqq \boldsymbol{\varepsilon} - \boldsymbol{f}(\boldsymbol{\varphi}) = \mathbf{0}_6,$$

such that $\boldsymbol{\varphi} \in \mathbb{R}^J$ becomes the solution $\boldsymbol{\varphi} \in \mathbb{R}^J$. This is analogous to

$$\boldsymbol{e}_{\mathrm{TCP}}(\boldsymbol{\varphi}) = \boldsymbol{\varepsilon} - \boldsymbol{\varepsilon}_c(\boldsymbol{\varphi}) = \mathbf{0}_6,$$

i.e., minimizing (or nulling) the error between the desired and current poses of the tool center point, $\boldsymbol{\varepsilon} \in \mathbb{R}^6$ and $\boldsymbol{\varepsilon}_c(\boldsymbol{\varphi}) \in \mathbb{R}^6$, respectively. [12, 29]

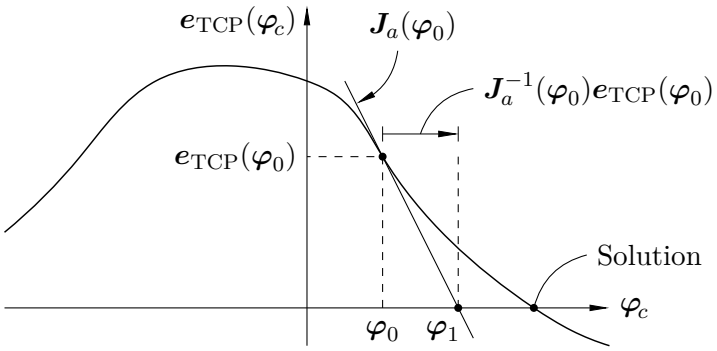


Figure B.4 First-order Newton method. Figure adapted from [12].

Starting with an initial guess $\boldsymbol{\varphi}_0 \in \mathbb{R}^J$, the desired configuration of the tool center point can be expressed using the Taylor expansion, i.e.,

$$\boldsymbol{\varepsilon} = \boldsymbol{\varepsilon}_c(\boldsymbol{\varphi}) = \boldsymbol{\varepsilon}_c(\boldsymbol{\varphi}_0) + \boldsymbol{J}_a(\boldsymbol{\varphi}_0)(\boldsymbol{\varphi} - \boldsymbol{\varphi}_0) + \cdots,$$

which can be rewritten to the form

$$\boldsymbol{\varepsilon} = \boldsymbol{\varepsilon}_c(\boldsymbol{\varphi}_0) + \boldsymbol{J}_a(\boldsymbol{\varphi}_0)\Delta\boldsymbol{\varphi}_0 + \cdots,$$

and by electing to terminate the Taylor expansion at its initial order, we can approximate the new solution as

$$\boldsymbol{\varphi}_1 = \boldsymbol{\varphi}_0 + \Delta\boldsymbol{\varphi}_0 = \boldsymbol{\varphi}_0 + \boldsymbol{J}_a^{-1}(\boldsymbol{\varphi}_0)\boldsymbol{e}_{\mathrm{TCP}}(\boldsymbol{\varphi}_0),$$

or more generally, any following solution can be computed using

$$\boldsymbol{\varphi}_{k+1} = \boldsymbol{\varphi}_k + \boldsymbol{J}_a^{-1}(\boldsymbol{\varphi}_k)\boldsymbol{e}_{\mathrm{TCP}}(\boldsymbol{\varphi}_k), \quad \forall\ k \in \mathbb{N}_0,$$

an expression termed the *first-order Newton method*, or the *Newton-Raphson method* for solving a system of nonlinear equations. This formula works only under the assumption that $\boldsymbol{J}_a^{-1}(\boldsymbol{\varphi}_k)$ exists. If not, one has to again resort to the Moore-Penrose pseudo-inverse of the analytical Jacobian. [12, 15, 27, 29]

Remark B.2. The convergence of the Newton method is heavily dependent on the initial guess $\boldsymbol{\varphi}_0 \in \mathbb{R}^J$. As the initial guess moves farther from the solution, the slope of the error function ultimately diverges from the solution. As an example, if we were to choose an initial guess in the "second quadrant" in Fig. B.4, the slope at this initial guess would steer far left, never reaching the solution. ◇

Remark B.3. If there exist multiple solutions to the inverse kinematics problem, the Newton method tends to find the one closer to the initial guess $\boldsymbol{\varphi}_0 \in \mathbb{R}^J$. ◇

Remark B.4. The initial idea of the Newton method's first order can be broadened to include higher-order Taylor expansions. This extension has the potential to hasten convergence by leveraging additional insights into the error function. However, calculating higher-order derivatives and inverting associated matrices can frequently incur significant computational costs or become infeasible in extreme scenarios. In such instances, engineers turn to alternatives, such as the *quasi-Newton* methods, which offer approximations of these higher-order derivatives matrices without the need for their explicit computation. [61, 62] ◇

B.3 MATLAB's Optimization Toolbox

In addition to already mentioned methods, MathWorks provides their Optimization Toolbox for MATLAB, which can be utilised to solve for $\boldsymbol{\varphi} \in \mathbb{R}^J$. The toolbox includes a large number of functions, each to be used in a specific case, some of which shall be briefly covered in the following text. However, for comprehensive insight into these functions and their use cases, consider referencing [35, 63].

- ▷ **`lsqnonlin`.** The `lsqnonlin` function can be used to solve for $\boldsymbol{\varphi} \in \mathbb{R}^6$ in cases when the inverse kinematics problem can be formulated as a least-squares problem, e.g., minimising the error between the desired and current TCP poses. [63]
- ▷ **`lsqcurvefit`.** The `lsqcurvefit` function is used for fitting curves to data. In the context of the inverse kinematics problem, if one has a forward kinematics model of the system, the `lsqcurvefit` function tries to fit this forward kinematics model to the provided data, i.e., the desired pose of the TCP. [63]

Naturally, the functionality of MATLAB's Optimization Toolbox extends beyond this short showcase and can be used in various different fields. The showcase of the `lsqnonlin` and `lsqcurvefit` serves merely as an illustrative example to underscore the large amount of options for numerical solution of the inverse kinematics problem, depending on the way we choose to describe it.