Design, analysis and kinematic control of highly redundant serial robotic arms

Doctorate Thesis

Candidate: Angelica Ginnante

Supervisors: Stéphane Caro
Enrico Simetti
François Leborne

Jury: Lucia Pallottino
David Daney
Philippe Wenger
Giorgio Cannata

Contents

List of Figures

List of Tables

List of Algorithms

Glossary

CAD	Computer-Aided Design
CNC	Computer Numerical Control
Opt-SNS	Optimal Saturation in the Null Space
PKM	Parallel Kinematic Mechanism
RP	Reverse Priority
RTR	Robot Transmission Ratio
SE(N)	N-dimensional Euclidean motion group
SNS	Saturation in the Null Space
TCP	Tool contact point
T&A	Tilt & Azimuth
T&T	Tilt & Torsion
TPIK	Task Priority Inverse Kinematics

Nomenclature

Robot design parameters

α	Slope of *Tube 1* and *Tube 2*
β	Angular offsets of the robotic arm
β_i	Optimized angular offset i
l	Links length of the robotic arm
l_i	Optimized link length i
r	Distance between *Platform 1* to the constant velocity joint and between the constant velocity joint to *Platform 2*
$\boldsymbol{\zeta}$	Design parameters

Robot variables

$\overline{\mathrm{CoM}}$	Mean center of mass distance
$\mathcal{F}$	Frame
L	Characteristic length
m	Dimension of the task space
n	Dimension of the joint space
n_r	Dimension of the real joint space
n_v	Dimension of the virtual joint space
$\boldsymbol{\omega}$	Angular velocity
q_1	Angular position of the first motor in NB-module
q_2	Angular position of the second motor in NB-module
$\mathbf{q}$	Joint position vector of robot
$\mathbf{q}_r$	Joint position vector of real joints
$\mathbf{q}_v$	Joint position vector of virtual joints
$\mathbf{p}$	Translation vector

$\dot{\mathbf{p}}$	Linear velocity
ϕ	Azimuth angle
$\mathbf{R}$	Rotation matrix
σ	Torsion angle
θ	Tilt angle
$\mathbf{t}$	Tip twist, $[\dot{\mathbf{p}}^\top, \boldsymbol{\omega}^\top]^\top$
$\mathbf{T}$	Homogeneous transformation matrix
$\mathbf{t}_d$	Target end-effector twist
$\vec{v}$	Displacement vector

Jacobian matrices

$\mathbf{J}$	General kinematic Jacobian matrix
$\mathit{\Delta}$	Determinant of the kinematic Jacobian matrix
$\mathbf{J}_1$	Kinematic Jacobian matrix of NB-module from $\mathbf{t}$ to $[\dot{\phi}, \dot{\theta}]^\top$
$\mathbf{J}_2$	Kinematic Jacobian matrix of NB-module from $[\dot{\phi}, \dot{\theta}]^\top$ to $\dot{\mathbf{q}}$
$\mathbf{J}_3$	Reduced kinematic Jacobian matrix of NB-module for kinematic analysis
$\mathbf{J}_e$	Kinematic Jacobian matrix of robot end-effector
$\mathbf{J}_{\mathrm{NB}}$	NB-module kinematic Jacobian matrix
$\mathbf{J}_{\mathrm{task}}$	Jacobian matrix related to one task
$\mathbf{J}_k$	Jacobian matrix of k^{th} task vector
$\mathbf{J}_\eta$	Dexterity Jacobian matrix
$\mathbf{J}_\mu$	Manipulability Jacobian matrix
$\mathbf{J}_\rho$	RTR Jacobian matrix
$\mathbf{J}_w$	Weighted kinematic Jacobian matrix using the characteristic length L

Task priority algorithm variables

$\mathscr{A}$	Action
$a^i(\mathrm{x})$	Task activation functions
$a^P(\mathbf{P})$	Action activation function
$\mathbf{A}_k$	Diagonal matrix of activation functions for k^{th} task vector
γ	Positive gain
E	Equality task
I	Inequality task

λ	Positive gain related to target convergence rate in TPIK algorithm
$\mathbf{P}$	Previous and current executed actions and time elapsed in current step
S_k	Solution at the k^{th} task
S_{k-1}	Manifold of solutions of the previous tasks before the k^{th} one
$\mathbf{x}$	Task vector
$\mathrm{x}(\mathbf{q})$	Robot configuration dependent scalar variable
$(\mathrm{x}_m,\mathrm{x}_M)$	Inequality control objective thresholds
x^*	Desired goal for $\mathrm{x}(\mathbf{q})$
$\dot{\mathrm{x}}^*$	Changing rate associated to x^*
$\dot{\bar{\mathrm{x}}}$	Feedback reference rate
$\dot{\bar{\mathbf{x}}}_k$	Vector of all reference rates of scalar control tasks

Kinetostatic indices

κ	Conditioning number
η	Dexterity
μ	Manipulability
ν	Bounded manipulability
ρ	Robot transmission ratio
$\epsilon(\eta,\nu,\rho)$	Linear combination of η, ν and ρ
H	Global conditioning index

Axis names

x	Axis x
$\vec{x}$	Unit vector axis x
y	Axis y
$\vec{y}$	Unit vector axis y
z	Axis z
$\vec{z}$	Unit vector axis z

Trajectory variables

$\vec{\mathbf{f}}$	Vector of forces
$\vec{\mathbf{v}}$	Vector of velocities

Workspace surface plot

A, B, C	Node grids
d	Distance
s	Positive or negative sign
R_1, R_2	Circumference rays
(x_J, y_J, z_J)	Node coordinates of grid J

General variables

$\mathbf{0}_{1\times3}$	Vector of zeros
δ	Threshold kinetostatic index velocity
$\delta_{\text{kinematic}}$	Threshold kinematic index velocity
δ_{velocity}	Threshold end-effector velocity
k	Counter variable
m_d	Number of optimized designs
m_r	Number of repetitions
(λ_1,λ_2)	Scaling factors
p	Number of trajectories
t	Time instant

Symbols

$\gg$	Much greater operator
$\triangleq$	Define operator

Considering a general variable a.

$\overline{a}$	Mean value of a
$\hat{a}$	$0.5\ a_{\min} + 0.25\ (\overline{a} + a_{\max})$
$a_{\max}$	Max value reached by a
$a_{\min}$	Min value reached by a
$\dot{a}$	Dexterity rate

CHAPTER 1

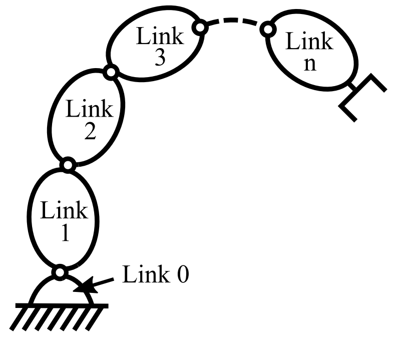
Introduction

This manuscript presents the work done during the Ph.D. thesis of Angelica Ginnante in collaboration between *Ecole Centrale de Nantes*, *Università degli Studi di Genova* and the company *Nimbl'Bot*, started in November 2020. The work was supervised by Professors Stéphane Caro and Enrico Simetti, and Doctor François Leborne.

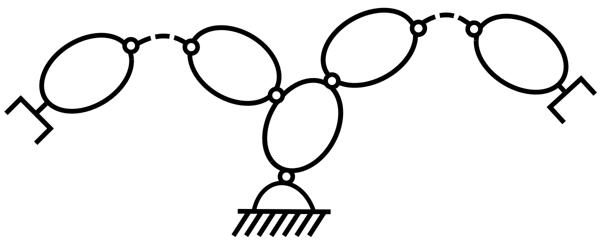
1.1 Robotic manipulators

The use of robotic manipulators in industry has grown in the last decades to improve and speed up industrial processes. Based on the desired application, different robotic solutions were developed. In [ISA15], the authors listed all the applications where manipulators are employed to improve the performance and resulting quality. Machining application is one of the main fields under study since it is a crucial task in the manufacturing industry to transform raw materials into functional parts [Che08]. Several researches were performed on robotic manipulators for machining tasks, pointing out the advantages and drawbacks [PDSC11, ISA15, KNH$^+$19]. Machining applications are mainly performed by computer numerical control (CNC) machine tools because of their high accuracy. Nevertheless, they are generally expensive and do not provide a high versatility [JW19]. Therefore, industrial manipulators started to be investigated. They can cover larger workspaces, increasing the range of achievable operations. Moreover, they reduce the scrap rates and production costs compared to CNC machines [CDGF13]. Industrial manipulators have already shown satisfactory performance in some tasks, like grinding [LUE90], polishing [TGA93] and deburring [NPRRA02]. Their main drawback is their overall lack of stiffness compared to CNC machines, leading to increased manufacturing errors [DCGF12, CDGF13].

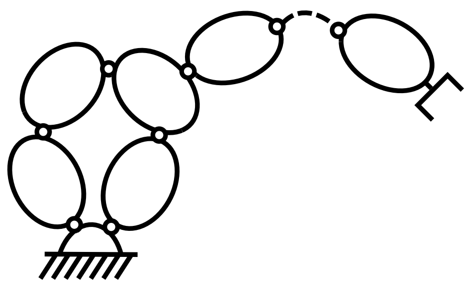
In [DK04], a robot manipulator is defined as composed of two main parts: (i) the end-effector to manipulate or transform objects and (ii) the articulated mechanical structure that moves the end-effector. One robot can have multiple end-effectors simultaneously in the same design. The articulated mechanical architecture aims to bring the end-effector to a desired pose or follow a specific trajectory. The robotic structure is composed of a rigid link series and is articulated by inserting joints between the links, generating a chain. The joints inserted in the chain can be of two types: revolute and prismatic. The first provides a rotational movement and the second a translational motion. The active joints actuate the robot and the passive joints that can not be directly actuated are moved by the active joints. The manipulators can be divided into four different sub-groups. The first is a simple open chain, called serial robots and shown in Fig.1.1a. There are three main classes for serial manipulators: (i) articulated robot [DP14], (ii) SCARA robot [DP14, KSP17] and (iii) Cartesian robot [DP14]. Each manipulator has its own scope and needs a different way of modeling. The authors of [DK04] present a complete guide to geometrically, kinematically and dynamically design a manipulator. The other two manipulator categories are the tree structure, shown in Fig.1.1b, and the closed



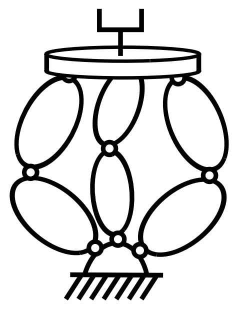
(a) Serial robot



(b) Tree-structure robot



(c) Closed-chain robot



(d) Parallel robot

Figure 1.1: Types of robotic manipulators [DK04]

chain, shown in Fig.1.1c. The presence of a closed chain always provides higher stability and strength to the robot, improving its stiffness. A specific type of closed-chain robot is the parallel one, shown in Fig.1.1d. The authors of [DK04] define a parallel robot as a mobile platform connected to a fixed base by a set of identical legs and an end-effector directly fixed to the mobile platform. However, a parallel robot can be formed of non-identical legs and include passive joints. A more detailed explanation for parallel robots is presented in [Mer06]. As described in [DK04], the geometric and kinematic modeling techniques developed for serial manipulators are inaccurate when used on parallel robots. So, specific modeling methods need to be employed. Moreover, the inverse geometric and kinematic problems are easy to solve in the case of parallel designs, but the direct geometric and kinematic problems are much more complex. The most common manipulators employed in industrial applications are serial and parallel.

In [DP14], the authors present a comparison of these two categories, listing advantages and drawbacks. Table 1.1 collects some of their main characteristics comparing these two robot types [DP14]. In general, serial and parallel manipulators have opposite advantages and disadvantages. In fact, serial robots are characterized by a larger workspace and higher flexibility, but their stiffness is commonly lower. On the contrary, parallel robots are usually stiffer and can hold higher payloads, but

Feature	**Serial**	**Parallel**
Workspace with respect to footprint	Large	Small
Stiffness	Low	High
Solving direct kinematic model	Easy	Complex
Solving inverse kinematic model	Complex	Easy
Modeling/solving dynamics	Simple	Complex
Payload/weight ratio	Low	High
Calibration	Simple	Complicated
Speed and acceleration	Low	High
Force error	Average	Accumulate
Position error	Accumulate	Average

Table 1.1: Serial and parallel robots comparison

their workspace is minimal compared to a serial design. So, one type is chosen over the other if the final application requires a wider workspace or a more robust design. This choice comports a trade-off favoring one category.

1.2 Hybrid manipulators

As described in Section 1.1, using a serial or parallel robot means choosing between a larger workspace or a stiffer architecture. When both are needed, hybrid manipulators can bring together the best of both worlds. The hybrid robots can be defined as a serial chain of non-serial mechanisms [TGK99, Tan00], a combination of closed-chain and open-chain architectures. Figure 1.2 shows a hybrid robot formed of a sequence of parallel robots serially connected. Hybrid robots can have very different designs. In [CBH08], the authors proposed a method for synthesizing new hybrid robots based on the desired application. The main drawback of hybrid robots is the complexity of their kinematics model [Tan00]. In fact, these robots can be actuated by complex closed loop or parallel mechanisms serially attached. So, the classical techniques used to analyze simpler designs, such as revolute and prismatic joints, cannot be used. In [PSVP13], a hybrid robot for surgical applications is proposed and its kinematic model is computed and analyzed. The work demonstrates

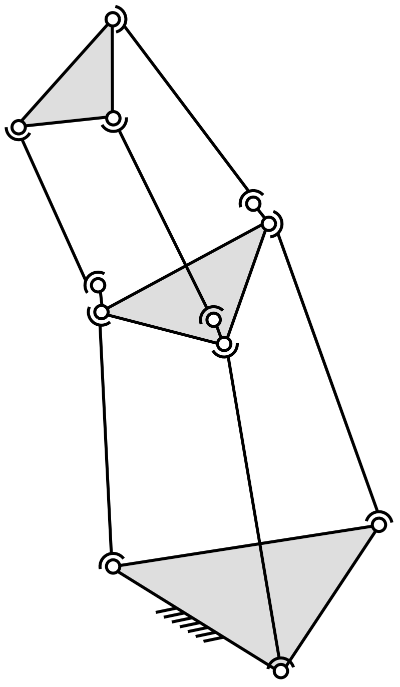


Figure 1.2: Drawing of a hybrid robot architecture [Tan00]

the complexity of defining hybrid robot kinematics.

In [KWdGF$^+$20], the authors present a survey that examines the development of series-parallel hybrid robots across different application domains: humanoids, multi-legged robotic systems and industrial manipulators. Humanoids are bipedal robots that mimic human anatomy with complex mechatronic systems. High dynamic performance in humanoids requires a stiff architecture and good mass distribution, which can be achieved using parallel mechanisms to design serial legs. Some examples of humanoid robots with parallel mechanisms include Lola [LBUP06], NASA Valkyrie [RSH$^+$15], TORO [EWO$^+$14], LARMbot [CWCC16], CARL [SNM$^+$17] and Disney Research bipedal robot [GKY18]. These robots utilize various types of parallel mechanisms in joints, such as spatial slider crank, rotational parallel mechanisms or parallel kinematic mechanism (PKM) modules. Then, there are the multi-legged robots designed for high-payload applications that incorporate closed-loop linkages and parallel mechanisms. The design of these robots incorporates different types of parallel mechanisms, like Stewart platforms, PKMs or parallelogram linkages, to enhance joint strength and stability. Several designs are proposed in the survey, for example HeritageBot [CCRC18], Menzi Muck M545 [JLKH19], MIT Cheetah [WWS$^+$17] and the quadrupedal platform Stoch[DBG$^+$19]. Finally, in industrial automation, series-parallel hybrid robots are used to improve the stiffness and enhance the workspace size. There exist different designs developed for

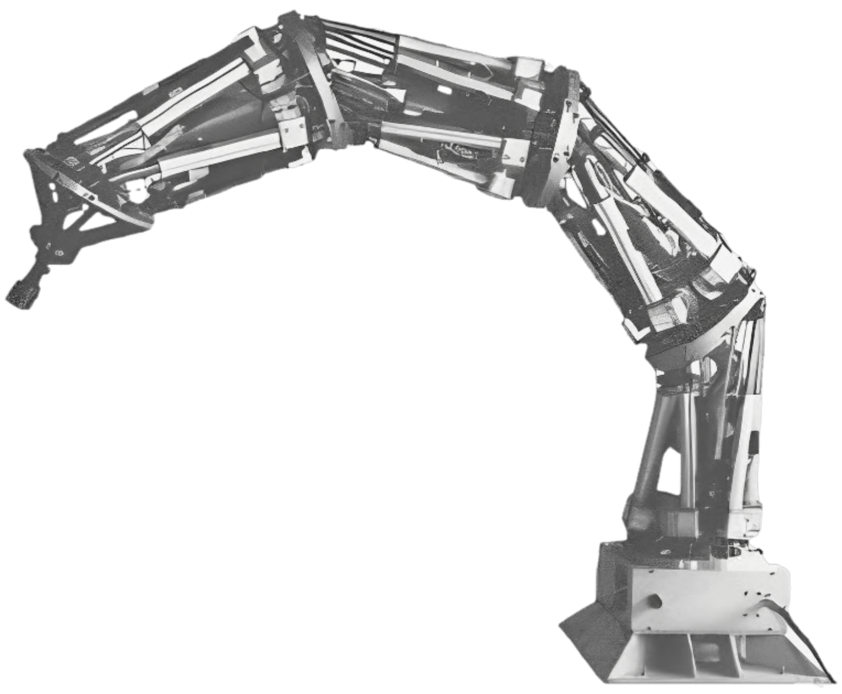


Figure 1.3: Representation of the Logabex LX4 [MD95]

real applications. One of the first examples is the Logabex LX4 robot [MD95], a serial concatenation of Stewart platforms. Figure 1.3 shows the Logabex LX4 robot. Other industrial manipulators from ABB, KUKA, Comau and the FANUC M-3iA/6A Delta robot employ parallelogram mechanisms to increase the stiffness for pick-and-place operations.

1.3 Kinematic redundant manipulators

"Redundant" means "exceeding what is necessary or normal". A robot is kinematically redundant with respect to a task when it has more degrees of freedom than the required amount necessary to perform that task [Sic90]. In fact, no manipulator is inherently redundant, but there are specific tasks with respect to which the robot becomes redundant [SKK08]. Mathematically, the kinematic redundancy appears when the dimension of its actuation vector $\mathbf{q} \in \mathbb{R}^n$ is greater than the dimension of the task vector $\mathbf{x} \in \mathbb{R}^m$, namely when $n > m$ [CB94]. So, the desired task can be achieved by multiple possible robot configurations. Both serial and parallel robots can be kinematically redundant. In the case of serial manipulators, redundancy is introduced by adding actuated joints into the serial chain. Contrarily, introducing redundancy in a parallel mechanism is less straightforward since there are several ways to do that, as described in [GS18]. This research concentrates on the kinematic redundancy of serial designs and does not address the topic of parallel mechanism redundancy.

In the manufacturing industry, the kinematic redundancy of robotic manipulators can be viewed as a possible way to improve the robotic machine abilities and performance [GST19]. One of the main motivations to introduce kinematic redundancy in a robotic manipulator is to increase the robustness to possible faults, improving the reliability [COW08]. Moreover, kinematic redundancy is also employed to increase the robot dexterity and enables new robot behaviors, like self-motion, i.e., a set of joint velocities causing no Cartesian motion to the end-effector [COW08]. The kinematic redundancy can also be used to work in cluttered environments [MRG17], such as medical robotics, and solve several tasks simultaneously while optimizing some performance criteria [SW95]. Restricting a manipulator to the minimum necessary number of degrees of freedom for accomplishing a task can lead to significant drawbacks in practical applications [COW08]. The limitations appear not only in the case of singularity issues but also in the presence of constraints like joint limits or obstacles within the workspace. A variety of seven degrees of freedom robots is

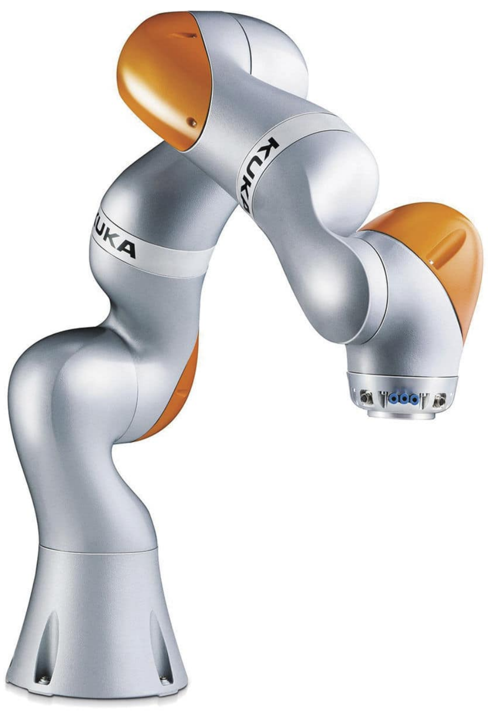


Figure 1.4: Representation of the KUKA LBR iiwa robot

largely used for different industrial applications to enhance dexterity. Some famous robotic examples are the KUKA LBR iiwa, shown in Fig. 1.4, and the ABB Yumi, shown in Fig. 1.5.

Robots are considered hyper-redundant when their number of degrees of freedom is much greater than the dimension of the task [HN91, Tan00], namely when $n \gg m$. Adding more degrees of freedom to already redundant manipulators allows solving many more simultaneous tasks [COW08]. Hyper-redundant robots can be divided into two categories: (i) rigid-link and (ii) continuum designs. Rigid-link hyper-redundant manipulators are the most straightforward evolution for redundant robots [COW08]. They are obtained by adding more links-joints to the already redundant manipulators. Generally, the link dimensions are reduced to make the robot design resemble a biological spine. This approach allows the creation of compact robotic manipulators with a high level of redundancy. The key concept behind rigid-link hyper-redundant robots is that they maintain all the geometric and kinematic conventions of classical manipulators, such as Denavit–Hartenberg based approaches and Jacobian computation methods, simplifying their use [COW08]. One example

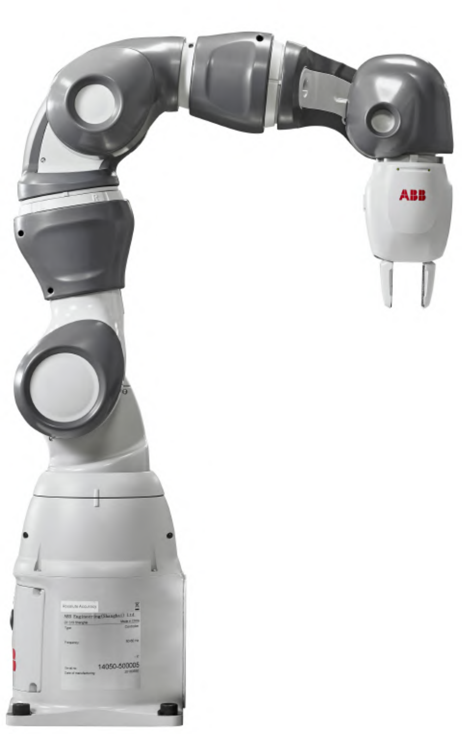


Figure 1.5: Representation of the single-arm ABB Yumi robot

of rigid-link hyper-redundant manipulators is the 30 degrees of freedom planar manipulator developed at Caltech [CB93]. Rigid-link hyper-redundant manipulators can also be employed in dual-arm designs like the one presented in [KTV+90] and a NASA special-purpose dexterous manipulator [HW03]. Recently, hyper-redundant manipulators have been increasingly used for inspection applications. This robots are sometimes called snake robots. As biological snakes, these robots can enter in small, irregular and challenging environments [Pet17] where it would be dangerous for human operators to go. Some examples of rigid-link robotic snakes can be found in the reviews [LPSG13, SAM+17, Pet17]. Figure 1.6 shows an example of snake robots. It comprises ten identical joint modules with passive wheels created for locomotion across flat surfaces.

The second category of hyper-redundant robots is called continuum. This type of manipulator carries the concept of kinematic redundancy to the extreme where the number of joints tends to infinity and the link lengths tend to zero [COW08]. Thanks to the number of joints that tend to infinity, the size of continuum robots can be highly reduced, making them optimal in the field of surgery [BKRC15]. In this case, different and more complex ways of modeling are required. The continuum

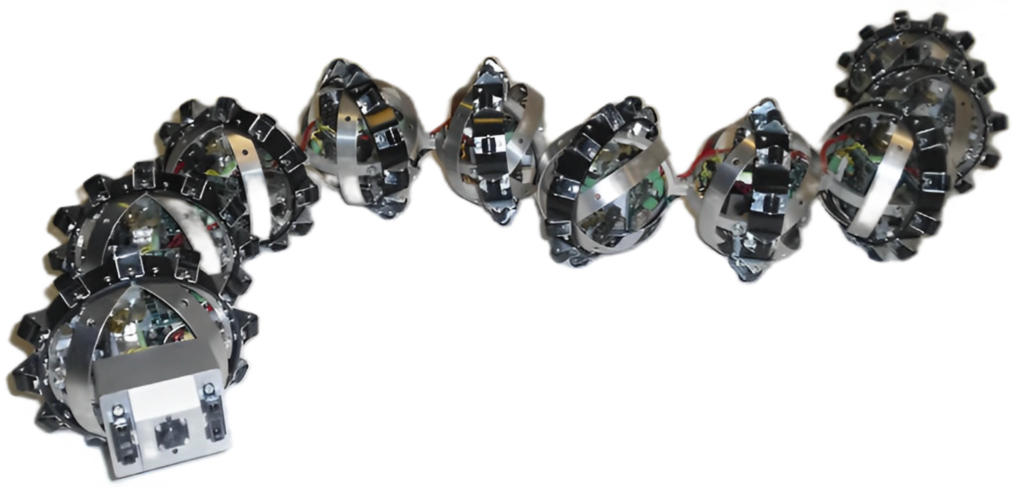


Figure 1.6: Representation of the snake robot Wheeko

robot category is not treated in the manuscript.

1.3.1 Kinematic redundancy resolution

The primary challenge posed by redundant manipulators lies in resolving their kinematic redundancy, as multiple viable solutions exist for a given task. In hyper-redundant robots, this challenge becomes even more pronounced, with the number of potential solutions approaching infinity. The simplest way to deal with this problem is using the pseudo-inverse Jacobian matrix method [DK04]. However, this approach neither avoids singularities nor takes advantage of the robot kinematic redundancy to optimize the robot configuration. Later, different types of algorithms and optimization techniques were developed to address the redundancy problem better. As described in [SKK08], the kinematic redundancy resolution methods divide into two main groups, via optimization and via task augmentation. In the first case, the degrees of freedom excess can be used to improve the value of performance criteria while executing the main task. The improved metric can depend on both the robot configuration and the velocities and forces applied to its end-effector. One example of optimization for kinematic redundancy resolution is the minimum effort solution [GW00]. This technique exploits the least infinity norm optimization to solve the inverse kinematic problem of redundant robots. In this research, the infinity norm optimization provides better results than the pseudo-inverse since the infinity norm minimizes the maximum component magnitude, meeting all the physical limit constraints. In [CLV06], the authors presented a kinematic redundancy resolution algorithm for a serial-parallel manipulator based on local kinematic optimization. The proposed algorithm effectively solved the inverse kinematic problem of a serial-parallel redundant manipulator, meeting the joint requirements for both active and passive joints. Moreover, the algorithm ensured smooth profiles for the active joints while performing the desired application. In [RMG16], the authors treated a problem of trajectory tracking for a general kinematic redundant robot as two interdependent problems, inverse kinematics and trajectory optimization, to identify the time-optimal path tracking solution. In this research, an enhancement to the differential inverse kinematics resolution method involves the addition of an optimal linear combination of null-space basis vectors from the corresponding Jacobian velocity vector. Many other researches were developed about the kinematic redundancy resolution via optimization in the literature.

The second way to solve the inverse kinematic problem exploiting the redundancy of a manipulator is via task augmentation [SKK08]. For example, considering a robot with seven degrees of freedom and a tracking trajectory task that requires only five degrees of freedom, two degrees of freedom remain available. The task vector can be augmented from five to six or seven degrees of freedom by adding other objectives. This resolution methodology is particularly effective in the case of hyper-redundant robots. In fact, these robots have many available degrees of freedom that are not employed by the main task and many additional tasks can be considered. The task augmentation technique is employed in this manuscript to kinematically control the employed manipulators, exploiting their kinematic redundancy. In fact, adding new objectives to the main task allows respecting different constraints, optimizing metrics or performing simultaneous applications. The literature review about task augmentation methods is proposed later in Section 3.1.

1.4 Nimbl'Bot robot overview

This section presents an overview of the first robot prototype developed by the company Nimbl'Bot and its purpose. A complex topic when talking about machining applications is the high precision milling, grooving and trimming of small metallic components. The elements to produce can have complex shapes and their required accuracy can be difficult to reach. CNC machines or existing serial machining robots can be too big and not have the required flexibility to properly shape these components. So, Nimbl'Bot started investigating new robotic solutions to solve this problem. The key concept behind the performed studies was identifying a new robotic design that ensures stiffness, positioning accuracy, compactness, modularity and flexibility. As pointed out in the previous sections, the typical actuation joints, revolute and prismatic, can not be used to build stiff, flexible and compact serial manipulators. So, Nimbl'Bot developed and patented a new actuation mechanism whose goal is to ensure stiffness and avoid backlash. Moreover, its compact design avoids the generation of bulky manipulators. This actuation mechanism is composed of two kinematic chains that together form a closed design. These two chains help distributing the applied forces on the entire design improving the stiffness and positioning accuracy. This actuation mechanism is called NB-module in this manuscript and its geometric and kinematic models are fully analyzed in Chapter 2. The NB-module is actuated by two motors that generate two rotational actuation. So, one mechanism provides two degrees of freedom.

The first prototype of the Nimbl'Bot robot is composed of a serial arrangement of ten NB-modules. The NB-module design requirement to ensure stiffness is to have a maximum solid reachable angle of $\pm\pi/6$ rad. So, many NB-modules need to be arranged together to ensure a sufficiently large orientation workspace. The prototype is shown in Fig. 1.7 and named NB-R1 in this manuscript. It can be divided into three regions, i.e., the shoulder, formed of three NB-modules, the elbow,

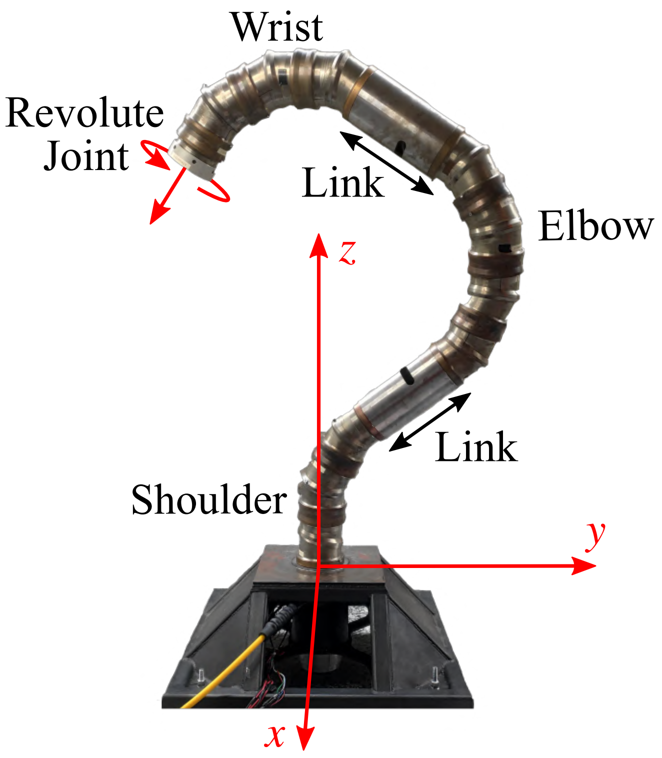


Figure 1.7: NB-R1 robot actuated by ten NB-modules mounted in series and a final revolute joint. Shoulder and wrist made of three NB-modules, covered solid angle of $\pm\pi/2$ rad each. Elbow made of four NB-modules, solid angle of $\pm 2\pi/3$ rad.

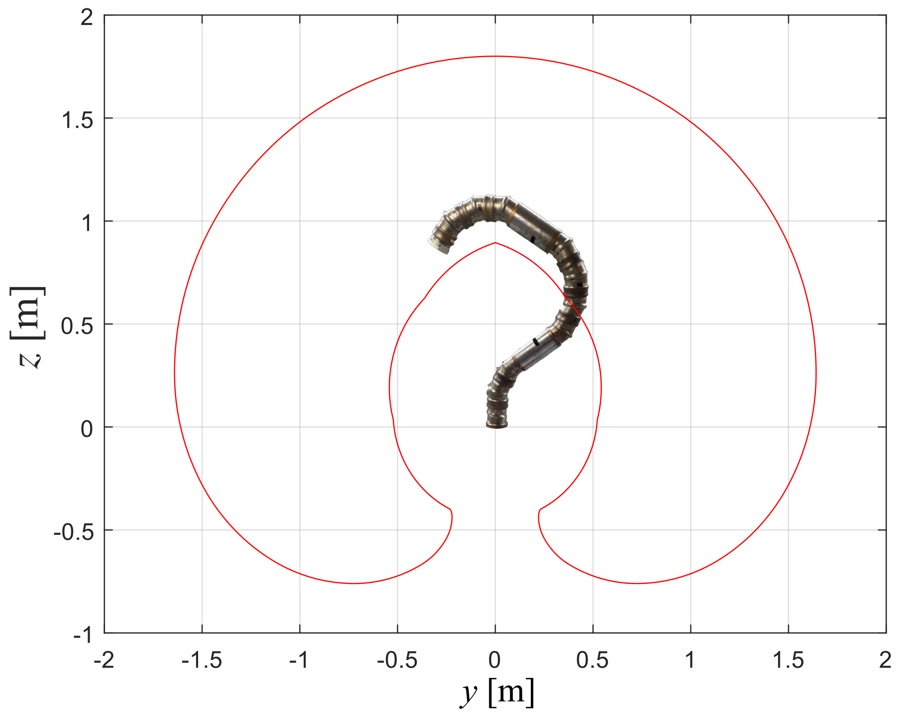
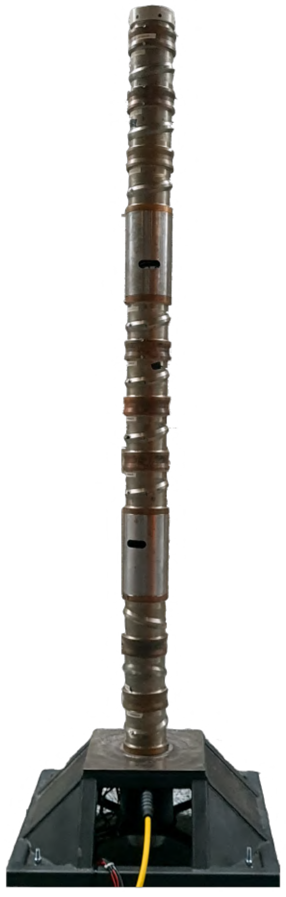
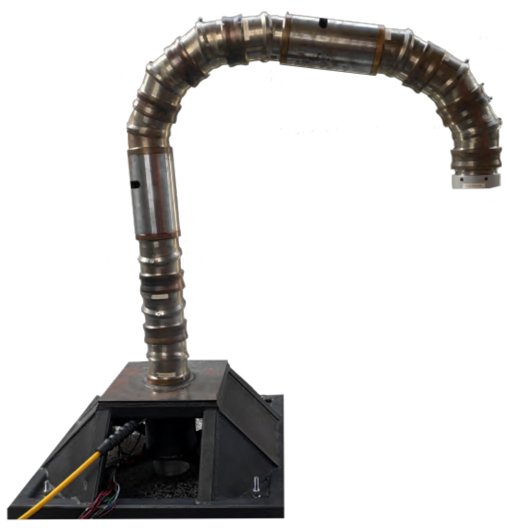


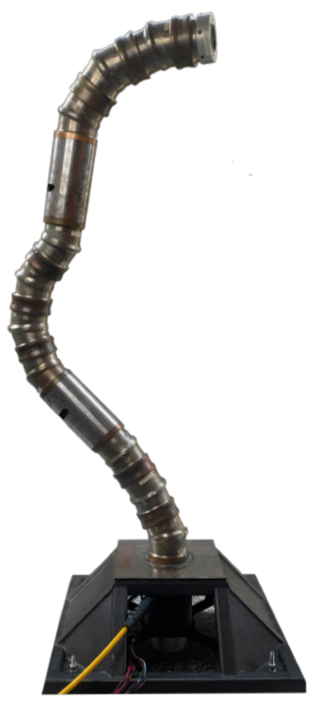
Figure 1.8: Vertical section of the NB-R1 workspace boundaries

four NB-modules, and the wrist, three NB-modules. Two links connect these three regions to increase even more the workspace size, allowing the end-effector to reach further poses. A vertical section of the NB-R1 workspace is shown in Fig. 1.8. This workspace is obtained through a new proposed method explained later in Chapter 5. The complete workspace is obtained by rotating this section around the axis z.

Since each NB-module provides two degrees of freedom plus a final revolute joint added at the end to allow adjusting the tool orientation, the NB-R1 has 21 degrees of freedom. This makes it a kinematic hyper-redundant robot considering that machining operations usually requires five degrees of freedom. So, the NB-R1 could possibly perform the same machining task with almost infinite possible configuration. This high redundancy provides versatility and the ability of working in cluttered environments. Moreover, this robot can be defined as hybrid, mixing serial and closed chains, and modular, an attachment of modules. The serial and closed chains mix gives strength to final design and the modularity improve the robot robustness and fault tolerance. A deeper discussion about modular robots is provided in Section 2.1. Finally, a cable passes inside the NB-R1 structure, constraining the design and canceling the remaining mechanical backlash. The two links are hollow to allow the routing of internal cables and reduce the final weight. Figure 1.9 shows four postures of the NB-R1 prototype.


(a)


(b)


(c)

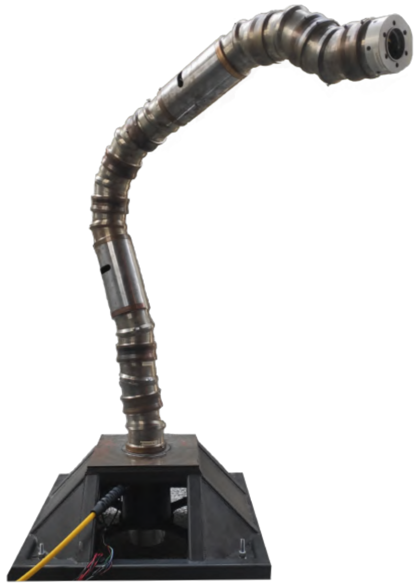

(d)

Figure 1.9: Four postures of the NB-R1 prototype

1.5 Thesis contribution and outline

The research proposed in this manuscript revolves around some open issues related to kinematically redundant spatial robots. As mention in the previous sections, these type of robots are increasingly explored because of the opportunities they can provide in the manufacturing industry. However, there are still several problems that needs to be addressed. Here, three main issues are analyzed and studied to propose some possible solutions. The first issue concerns the kinematic redundancy resolution problem. To address this topic, a task priority kinematic resolution algorithm is used for the kinematic control of redundant manipulators. This type of algorithm exploits the kinematic redundancy of the robot to solve multiple simultaneous tasks. The second issue is related to the design optimization of kinematic redundant robots as a function of their main application. A new design optimization method based on the task priority kinematic resolution algorithm is proposed. This new process gives as output some guidelines to build performant robots with respect to the desired application and working area. The third issue is related to the workspace determination of kinematic redundant robots, which is a complex and important topic. A new workspace determination process again base on the task priority kinematic resolution algorithm is proposed. This new method can be defined as ray-based and accurately detect the workspace boundaries of highly redundant designs in a small period of time. The proposed solutions to the three addressed issues are all tested on some Nimbl'Bot robot designs.

Here, the chapters organization is presented. Chapter 2 describes the mechanism developed by Nimbl'Bot to build kinematic redundant robots, called NB-module. The geometric and kinematic models of this mechanism are presented and explained. Then, the NB-module design parameters are investigated as a function of its geometric and kinematic performance. Part of the work presented in this chapter was published in [GLC^{+}21, GCSL23a]. Chapter 3 presents the task priority kinematic resolution algorithm and the tasks developed to perform a kinetostatic optimization of the robot configuration while performing other tasks. The algorithm is tested making the NB-R1 robot following some trajectories while improving its kinetostatic performance. Part of the work presented in this chapter was published in [GCSL23a]. Chapter 4 describes the new design optimization process for kinematic redundant manipulators. This method is tested to optimize the NB-R1 robot design with respect to a set of trajectories. The optimization is performed to obtain high kinetostatic performance while following th trajectories. In the end, some guidelines to build kinetostatic performant designs are obtained. Part of the work presented in this chapter was published in [GSCL23]. Chapter 5 presents a workspace determination algorithm developed for kinematic redundant robots. This process is tested on three different designs of the Nimbl'Bot robot to demonstrate its versatility. Then, it is compared with other two methods demonstrating its preeminence. Part of the work presented in this chapter was published in [GCSL23b]. Chapter 6 presents the conclusion and future works. Figure 1.10 shows a flowchart with the manuscript organization.

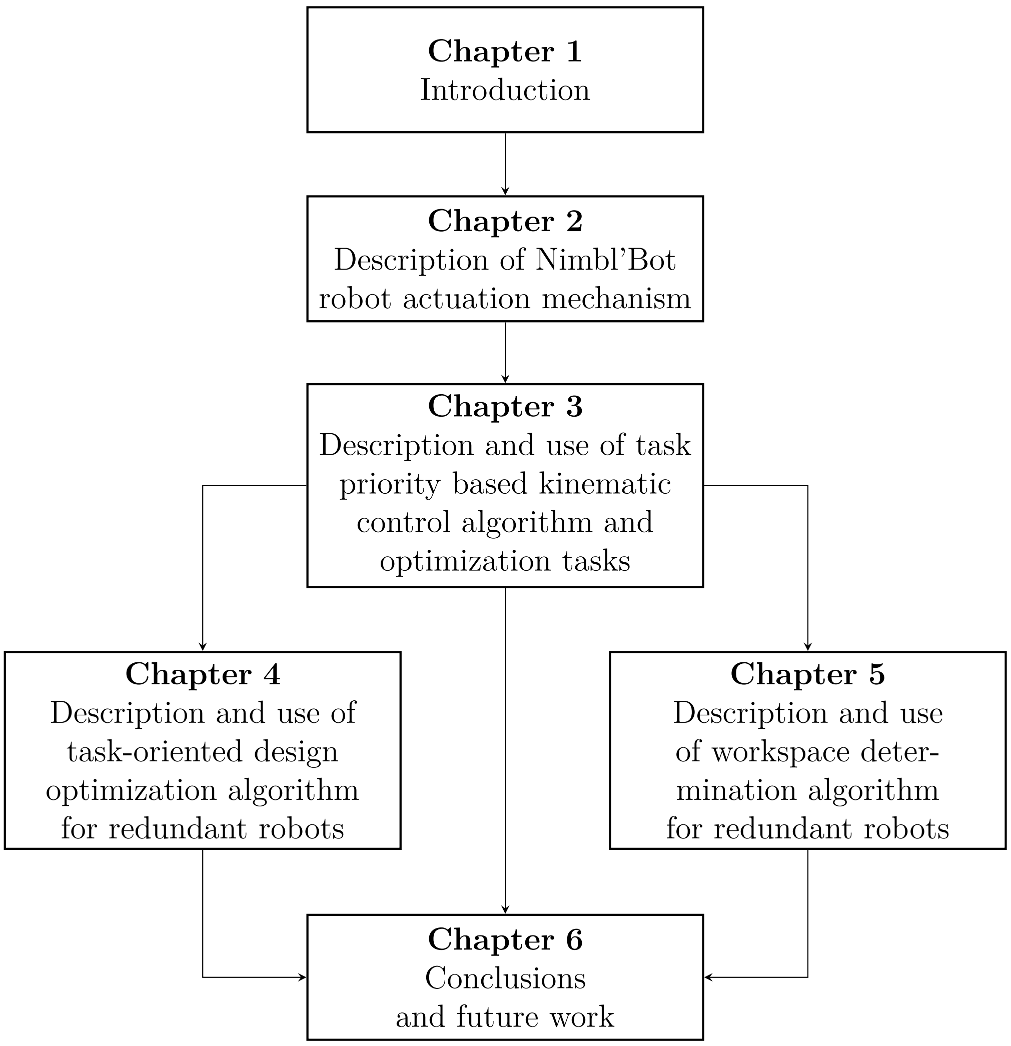

Chapter 1
Introduction
Chapter 2
Description of Nimbl'Bot robot actuation mechanism
Chapter 3
Description and use of task priority based kinematic control algorithm and optimization tasks
Chapter 4
Description and use of task-oriented design optimization algorithm for redundant robots
Chapter 5
Description and use of workspace determination algorithm for redundant robots
Chapter 6
Conclusions and future work

Figure 1.10: Flowchart of the thesis manuscript outline

Chapter 2

Geometric and Kinematic Analysis of Nimbl'Bot NB-Module

This chapter describes the two degrees of freedom mechanism patented by the company Nimbl'Bot [Duf21] and called NB-module. Here, the models are recalled and further analysis are performed. The chapter is organized as follows. Section 2.1 introduces the possible ways to build modular redundant manipulators and the existing actuation mechanisms that can be used. Section 2.2 describes the NB-module design and its actuation. Section 2.3 presents the geometric model computation for the NB-module and shows its workspace and joint space for specific design parameter values. Section 2.4 computes the NB-module kinematic model and analyzes its kinematic performance based on the design parameter values. The conclusion are presented in Section 2.5.

2.1 State of the art on mechanism and modular robots

As introduced in Section 1.3, kinematic redundant robots can be designed in different ways and provide several advantages. Kinematic redundant robots are generally classified into three categories: discrete, continuous and modular robots [CB92, CB95]. Modular robots were firstly introduced in [FN88, FK90]. They can be defined as an assembly of several actuation mechanisms, or modules [Bra16]. Each module is functionally and structurally independent [Bra16] and disposes of few degrees of freedom [AM15]. Usually, modular robots are composed of many modules and have a high number of degrees of freedom, leading to kinematic redundancy [Bra16]. Modular robots were developed as a solution to the low flexibility and adaptability of fixed-body monolithic conventional robots [AM15]. This type of robots has three main advantages: versatility, robustness and lower costs [APS19, BRS+17]. The modular system versatility refers to the ability of transforming into numerous shapes [APS19]. This feature allows completing various tasks in different and cluttered environments. The versatility can be evaluated by the number of isomorphic configurations [DDN19] that the robot can reach and the number of degrees of freedom it possesses. This characteristic increases together with the number of modules that compose the robot. Moreover, the robot design can be quickly adapted according to a given task by changing the number of modules [Bra16]. Redundancy and self-repair due to the use of many identical modules provide robustness to the system, enabling any broken module to be replaced easily [APS19]. Thanks to the large amount of degrees of freedom, if one module stops working properly it can be disabled and the others can still complete the desired task. 3D printing prototyping and later batch production are cost-effective methods for developing repeated modules while maintaining low costs [APS19]. One drawback of modular robots is the raising complexity of computing the geometric and kinematic models [Bra16]. In fact, the modules used to built the modular manipulator can have complex non-conventional designs. So, identifying their geometric and kinematic models can become more difficult.

A first approach to the kinematic modeling for modular robots was presented in [BZL89]. The research describes a methodology to derive the individual kinematic models of all the modular units and a global kinematic model for any robot configured using these modular units. In [CY96], the authors presented a newly developed modular robot aimed for assembly task. The authors employed the so called dyad kinematics along with a graph traversing algorithm to derive the forward kinematics. Other types of modular robots developed for industrial application, like assembly task, were proposed later in [ABR08, SY11, LXGC17]. Thanks to the system versatility, modular robots can be employed in cluttered environments. A mechanism for modular redundant snake robots was introduced in [WBC+03]. The authors demonstrated how the obtained robot was able to inspect unreachable areas dangerous for users. In [WJP+07, JWT+11, WBB+12], different types of modular snake robots are described demonstrating their abilities in inspection applications. In [SWBC03], the authors presented a two degrees of freedom mechanism for modular redundant robots. The investigated concept uses a complex design optimized for compactness, strength and range of motion. Then, a similar improved design was proposed in [SWC06]. In [RSBT18], a flexible universal spatial robotic tail actuated by a cable-driven segment is introduced. Another possible application for modular redundant robot is the surgical environment. In [OHAH+20], the authors proposed an actuation mechanism for surgical snake robots and tested its abilities. However, none of these mechanisms were ever used to build machining robots. The mechanism called NB-module and proposed by Nimbl'Bot is specifically developed to build compact machining robots.

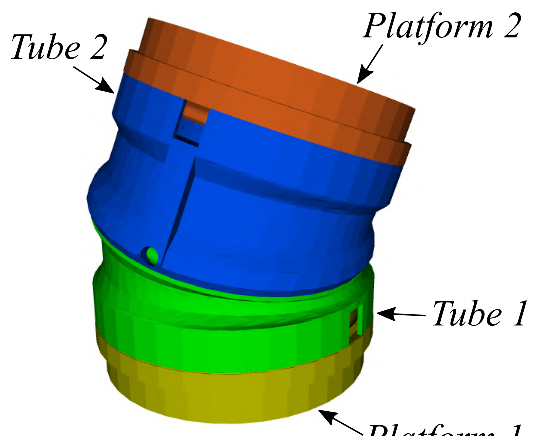
2.2 Nimbl'Bot mechanism description

The NB-module is the actuation mechanism developed by the company Nimbl'Bot to build compact stiffer modular robots. This innovative mechanism is designed to donate strength and stiffness to manipulators, improving the end-effector pose precision and reducing the backlash. This new design consists of a closed kinematic chain mechanism composed of two chains, one internal and the other external. As Chapter 1 points out, the closed chain architectures provide higher stiffness and stability to the entire manipulator. Moreover, the NB-module design is compact, avoiding the development of bulky designs. It is actuated by two motors, providing two degrees of freedom. This section describes the NB-module external and internal design. Then, its actuation is presented.

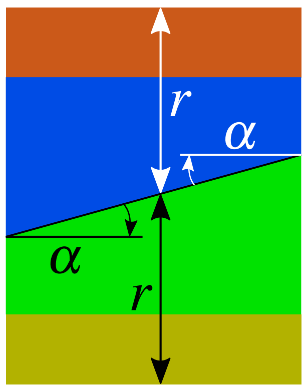
2.2.1 Description of the NB-module external kinematic chain

The external kinematic chain has seven different components. Four of them are shown in Fig. 2.1a. The fixed base, in yellow, is named *Platform 1* and is considered centered on the origin frame for which the NB-module transformation matrix is calculated. Above *Platform 1*, there is a rotating cylinder, in green, named *Tube 1*. *Tube 1* is a hollow cylinder cut by an oblique plane with height r and slope α. The design parameters r and α are shown in Fig. 2.1b. The first motor actuates the component *Tube 1*. The motor is attached directly to the inner side of *Tube 1*. In this way, *Tube 1* can rotate around the vertical axis that passes through the center of *Platform 1*. *Tube 2*, in blue, is placed over *Tube 1*. In this case, they have the same shape, but, in principle, their height r and slope α could be different. The second motor actuates *Tube 2* and is attached to its inner side. *Tube 2* can rotate around the axis perpendicular to and centered in *Platform 2*. The external kinematic chain is closed by *Platform 2*, the moving platform in orange, which is the end-effector of the NB-module.

Cutting obliquely the cylindrical tubes results in an elliptical shape. However,



(a) 3D view with component names



(b) Front view scheme with design parameters

Figure 2.1: External view of the NB-module

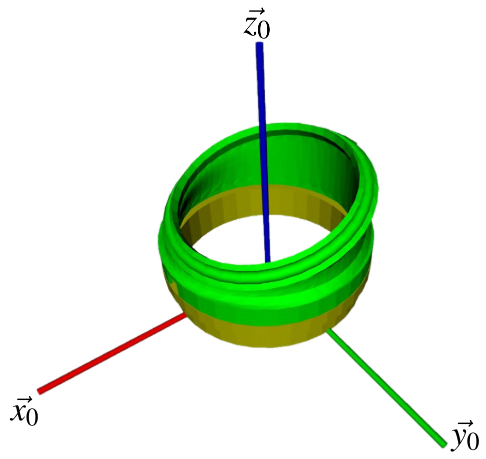


Figure 2.2: View of *Tube 1* oblique side

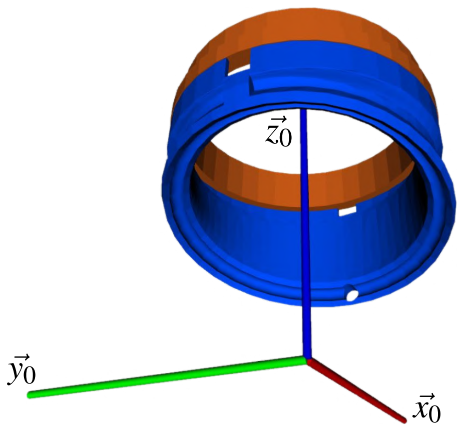
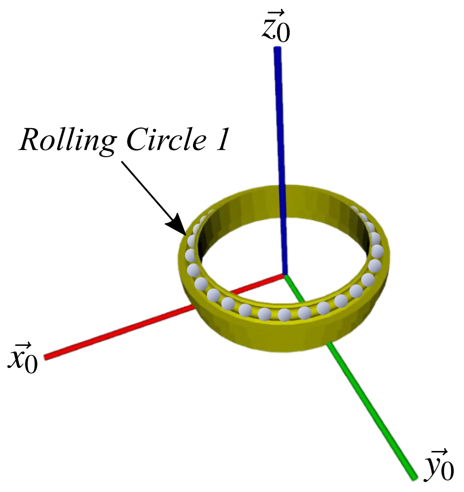
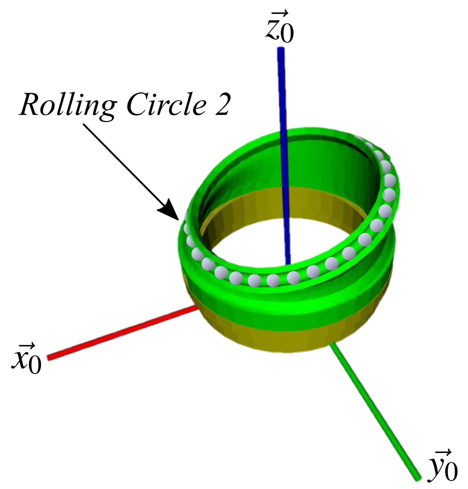


Figure 2.3: View of *Tube 2* oblique side

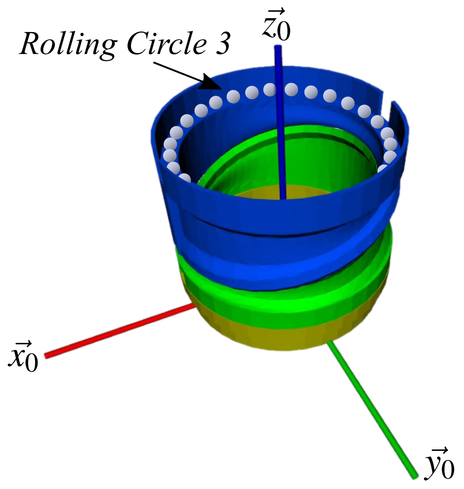
the tube oblique sides are reshaped in a circular way to allow a continuous rotation between the oblique planes. So, as it can be seen in Figs. 2.2 and 2.3, a circular groove is designed above the inclined sides of the two tubes. Three rolling circles formed of a series of small balls are inserted between the platforms and the tubes to allow a fluid movement. The balls are inserted in the grooves machined in the platforms and tubes, as shown in Fig. 2.4. These rolling circles are called *Rolling Circle 1*, *Rolling Circle 2* and *Rolling Circle 3* and represent the three last elements of the external kinematic chain. Consequently, *Tube 1* and *Tube 2* can independently rotate with a continuous movement.



(a) Location of *Rolling Circle 1* over *Platform 1*



(b) Location of *Rolling Circle 2* over *Tube 1*



(c) Location of *Rolling Circle 3* over *Tube 2*

Figure 2.4: Location of rolling circles formed of a series of balls in NB-module

2.2.2 Description of the NB-module internal kinematic chain

The internal kinematic chain has four components, as shown in Fig. 2.5. Two of those also belong to the external chain, i.e. *Platform 1* and *Platform 2*, generating the closed kinematic chain mechanism. *Platform 1* is linked to the component *Ball Nut* in purple through a prismatic joint, which prevents internal breaks while the NB-module is actuated. These could occur due to dimensional inaccuracies in the mechanical parts. Following that, there is the element *Ball Joint Axis* in cyan that forms a constant velocity joint with the element *Ball Nut*. Finally, *Ball Joint Axis* is linked to *Platform 2* through another prismatic joint, again to avoid internal breaks. The variable r equal to the tube heights represents also the distance between *Platform 1* and the constant velocity joint and between the constant velocity joint

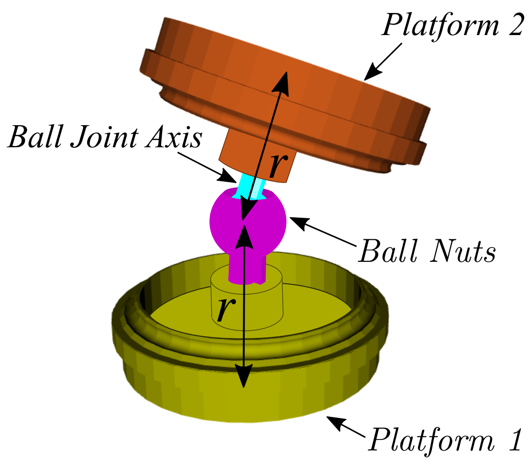


Figure 2.5: Internal view of the NB-module with component names

and *Platform 2*.

One NB-module interesting feature is the presence of a constant velocity joint, which works like a universal joint, but allows its two ends to rotate at the same velocity [Car09]. Therefore, fixing *Platform 1* means forcing no rotation to the whole internal kinematic chain. So, thanks to the constant velocity joint and the rolling circles that decouple the rotation of each component, the tube rotations lead to an inclination of *Platform 2* with no rotation about its normal axis. The NB-module amounts to a zero-torsion mechanism.

2.2.3 NB-module actuation

The lower half NB-module, yellow *Platform 1* and green *Tube 1*, is actuated in the same way as the upper half NB-module, blue *Tube 2* and orange *Platform 2*. The lower half NB-module actuation generates a rotation angle between *Platform 1* Plane and *Tube 1* Plane, called q_1. The upper half NB-module actuation works equally, generating a rotation angle called q_2 between *Platform 2* Plane and *Tube 2* Plane. Figures 2.6 and 2.7 show the actuation joint variables q_1 and q_2 with respect to the component planes. A peculiarity of the NB-module is that both the motors have endless courses and can infinitely rotate, never reaching a limit. Figure 2.8 shows the zero position of the NB-module. In this configuration, q_1 and q_2 are both equal to 0. The shortest side of *Tube 1* is along the positive side of axis $\vec{x_0}$ and the shortest side of *Tube 2* is along the negative side of axis $\vec{x_0}$.

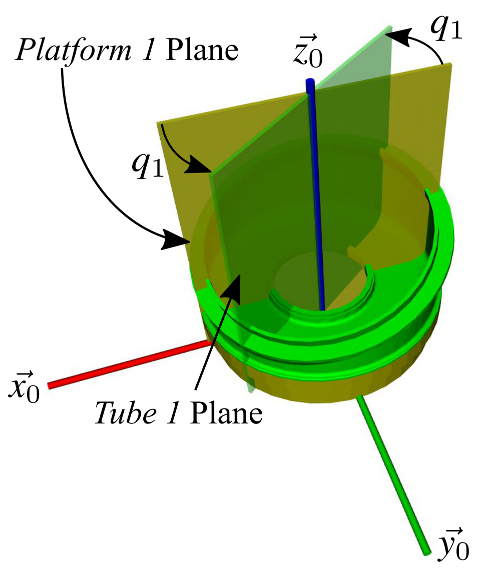


Figure 2.6: Actuation variable q_1 of the NB-module

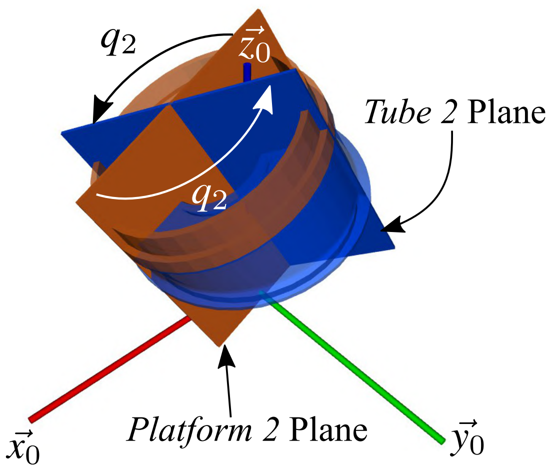


Figure 2.7: Actuation variable q_2 of the NB-module

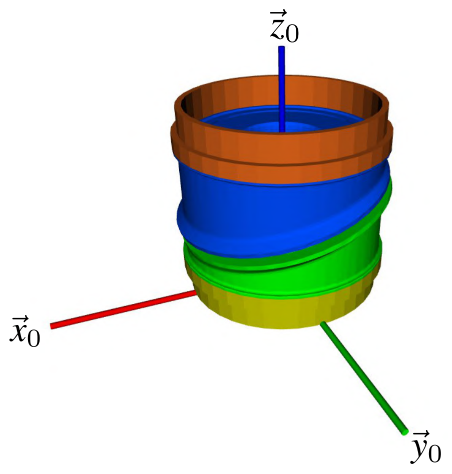


Figure 2.8: Home pose of the NB-module

2.3 Geometric model of NB-module

This section presents the geometric model of the NB-module. Firstly, the zero-torsion characteristic of the NB-module is explained. Then, the complete NB-module geometric model is presented with its transformation matrices. Finally, its workspace is shown and analyzed, setting the design parameters to specific values.

2.3.1 Zero-torsion mechanism

The ending platform of a zero-torsion mechanism never rotates about its normal axis [NCW18], but it can tilt around the axis parallel to the platform. The zero-torsion rotation matrix can be derived from the Tilt & Torsion (T&T) angles notation defined in [BZG02]. Figure 2.9 depicts the T&T angles convention. It involves only two rotation angles, the tilt angle θ rotating around the axis a and the torsion angle σ rotating around the axis $\vec{z^*}$. The azimuth angle ϕ, rotating around the axis $\vec{z}$, defines the vertical plane orientation perpendicular to the axis a. The tilt θ, torsion σ and azimuth ϕ angles are shown in Fig. 2.9. These angles take values as follows: $\theta \in [0, \pi)$ rad, $\sigma \in (-\pi, \pi]$ rad and $\phi \in (-\pi, \pi]$ rad. The T&T convention rotation matrix is:

$$\mathbf{R}(\phi,\theta,\sigma) = \begin{bmatrix} \cos\phi\cos\theta\cos(\sigma-\phi) - \sin\phi\sin(\sigma-\phi) & -\cos\phi\cos\theta\sin(\sigma-\phi) - \sin\phi\cos(\sigma-\phi) & \cos\phi\sin\theta \\ \sin\phi\cos\theta\cos(\sigma-\phi) + \cos\phi\sin(\sigma-\phi) & -\sin\phi\cos\theta\sin(\sigma-\phi) + \cos\phi\cos(\sigma-\phi) & \sin\phi\sin\theta \\ -\sin\theta\cos(\sigma-\phi) & \sin\theta\sin(\sigma-\phi) & \cos\theta \end{bmatrix}. \quad (2.1)$$

The value of σ can be set to zero to obtain the rotation matrix of a zero-torsion mechanism,

$$\mathbf{R}(\phi,\theta) = \begin{bmatrix} \cos^2\phi\cos\theta + \sin^2\phi & \cos\phi\sin\phi(\cos\theta - 1) & \cos\phi\sin\theta \\ \sin\phi\cos\phi(\cos\theta - 1) & \sin^2\phi\cos\theta + \cos^2\phi & \sin\phi\sin\theta \\ -\sin\theta\cos\phi & -\sin\theta\sin\phi & \cos\theta \end{bmatrix}. \quad (2.2)$$

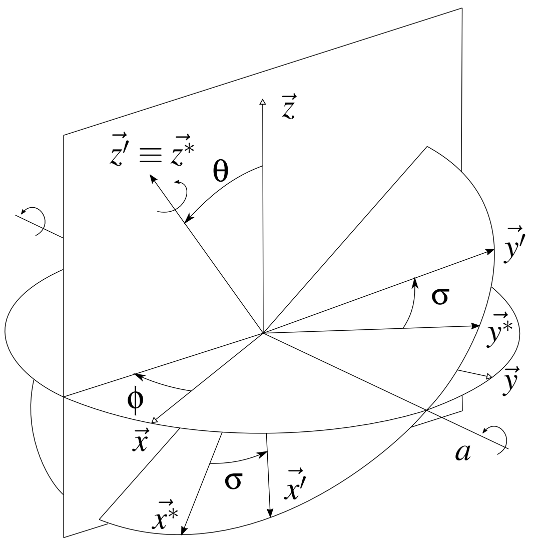
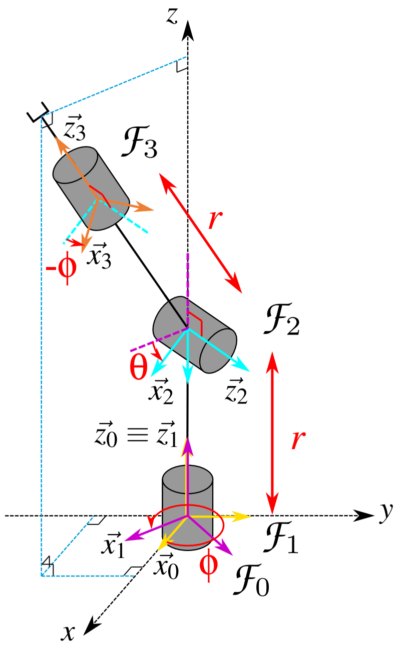


Figure 2.9: Tilt and torsion angle notation [BZG02]

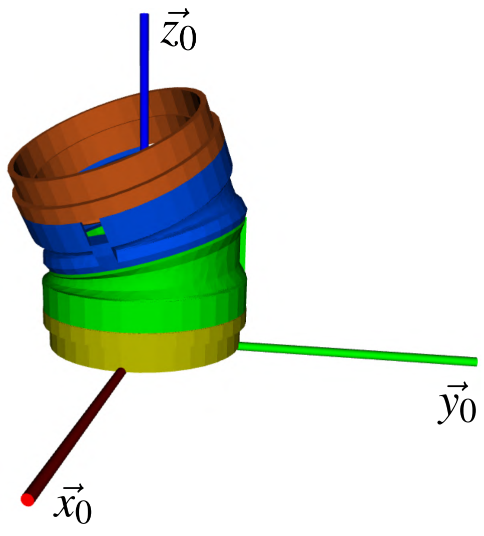
2.3.2 NB-module transformation matrix computation

Here, the geometric model of the NB-module is described. Since the NB-module is a zero-torsion mechanism, the rotation matrix of a frame rigidly attached to *Platform 2* with respect to a frame rigidly attached to *Platform 1* can be described using the notation presented in section 2.3.1. Here, the torsion angle σ is set to zero and the T&T notation becomes the Tilt & Azimuth (T&A) notation. Figure 2.10 shows the T&A based geometric model of the NB-module next to its CAD representation. A series of three revolute joints form the geometric model. The first revolute joint represents the azimuth angle ϕ of the NB-module. The second revolute joint is the tilt angle θ. The third revolute joint is constrained to have the negative value of the azimuth angle -ϕ as a consequence of the NB-module zero-torsion characteristics. Given the kinematic chain of Fig. 2.10a, the NB-module rotation matrix ${}^0\mathbf{R}_3(\phi, \theta)$ and translation vector ${}^0\mathbf{p}_3(\phi, \theta, r)$ pointing from the origin of frame $\mathcal{F}_0$ to the origin of frame $\mathcal{F}_3$ are

$$
{}^0\mathbf{R}_3(\phi, \theta) = \begin{bmatrix} \cos^2\phi\cos\theta + \sin^2\phi & \cos\phi\sin\phi(\cos\theta - 1) & \cos\phi\sin\theta \\ \sin\phi\cos\phi(\cos\theta - 1) & \sin^2\phi\cos\theta + \cos^2\phi & \sin\phi\sin\theta \\ -\sin\theta\cos\phi & -\sin\theta\sin\phi & \cos\theta \end{bmatrix}, \quad (2.3)
$$



(a) Geometric model scheme



(b) CAD representation of geometric model posture

Figure 2.10: Tilt and azimuth geometric model of the NB-module with CAD representation

and

$$ {}^{0}\mathbf{p}_3(\phi,\theta,r) = \begin{bmatrix} r\sin\theta\cos\phi \\ r\sin\theta\sin\phi \\ r + r\cos\theta \end{bmatrix}. \tag{2.4}$$

The equation defining the translation vector ${}^{0}\mathbf{p}_3$ also represents the so-called spherical coordinates. The complete homogeneous transformation matrix of the NB-module from frame $\mathcal{F}_0$ to frame $\mathcal{F}_3$ is expressed as

$$ {}^{0}\mathbf{T}_3(\phi,\theta,r) = \left[\begin{array}{c|c} {}^{0}\mathbf{R}_3(\phi,\theta) & {}^{0}\mathbf{p}_3(\phi,\theta,r) \\ \hline \mathbf{0}_{1\times 3} & 1 \end{array}\right]. \tag{2.5}$$

The azimuth ϕ and tilt θ angles express the NB-module transformation matrix in the T&A notation. However, these angles do not represent the tubes angular position, called q_1 and q_2. So, the azimuth ϕ and tilt θ angles need to be expressed as functions of the actuation variables q_1 and q_2,

$$\begin{cases} \phi = \dfrac{q_1 + q_2 - \pi}{2} \\ \theta = \arctan\left(-\dfrac{2\tan\alpha\sin\left(\frac{q_1-q_2}{2}\right)}{1-\tan^2\alpha\sin^2\left(\frac{q_1-q_2}{2}\right)}\right) \end{cases}, \tag{2.6}$$

where *arctan* is the tangent inverse function. The sign given as input to *arctan* is very important since multiple angles can return the same tangent value So, it is necessary to consider the *arctan* input sign to accurately determine the correct angle. Similarly, q_1 and q_2 can be expressed as functions of ϕ and θ,

$$\begin{cases} q_1 = \phi + \arccos\left(-\dfrac{\cos\alpha\,(\cos\theta - 1)}{\sin\alpha\sin\theta}\right) \\ q_2 = \phi - \arccos\left(-\dfrac{\cos\alpha\,(\cos\theta - 1)}{\sin\alpha\sin\theta}\right) + \pi \end{cases}, \tag{2.7}$$

where α is the slope of the oblique planes in *Tube 1* and *Tube 2*. When the tilt θ is equal to 0, the value of the actuation variables is $q_1 = q_2 = \phi + \pi/2$.

The azimuth ϕ and tilt θ angles simplify visualizing the NB-module orientation. In fact, the planes identified by these two angles help understanding the NB-module direction. Figure 2.11 shows the Azimuth Plane with the azimuth angle ϕ. So, ϕ gives the orientation along which *Platform 2* is tilted. Figure 2.12 shows the Tilt

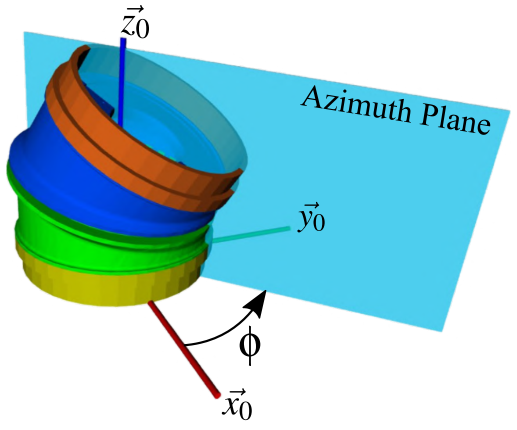


Figure 2.11: Azimuth plane on the NB-module

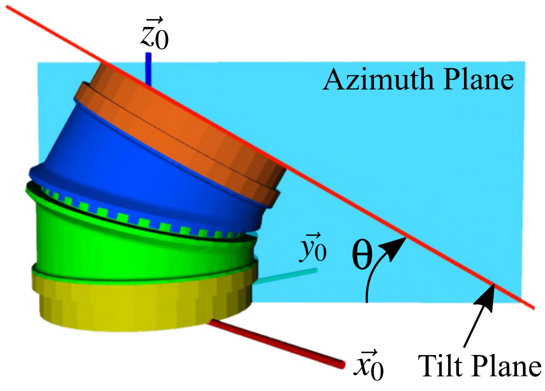


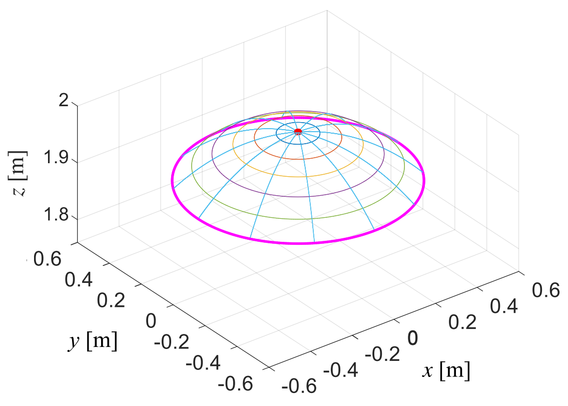
Figure 2.12: Tilt and azimuth planes on the NB-module

Plane, which is parallel to the top of the *Platform 2* and oriented along with the Azimuth Plane. The angle between the plane spanned by axes $\vec{x_0}$ and $\vec{y_0}$ and the Tilt Plane is the tilt angle θ.

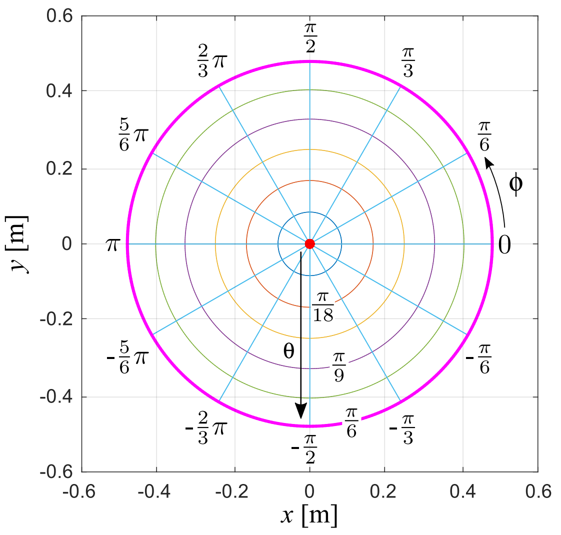
2.3.3 NB-Module workspace and joint space for specific design parameter values

The NB-module workspace can be computed using the translation vector ${}^0\mathbf{p}_3$ expressed in Eq. (2.4), representing the spherical coordinates. In fact, the workspace is a portion of a sphere whose dimension depends on the length of r and the amplitude of α. Here, the design parameters are set to $r = 1$ m and $\alpha = \pi/12$ rad $= 15°$ to give an example of the NB-module workspace. Figure 2.13 shows the 3D and 2D views of the workspace. It corresponds to all the positions reached by frame $\mathcal{F}_3$ for all the possible values of q_1 and q_2 in $(-\pi, \pi]$ rad. Each point on the sphere portion can be reached by two combinations of q_1 and q_2, both of them corresponding to the same orientation of the moving platform. Figure 2.13b plots the tilt θ and azimuth ϕ angles on the 2D view of the workspace. The value of ϕ stays in the range $(-\pi, \pi]$ rad and θ in $[-\pi/6, \pi/6]$ rad. In fact, the maximum absolute possible value of θ is twice the slope of each tube, i.e. $2\alpha = \pi/6$ rad. The NB-module workspace is symmetric with respect to the z-axis.

Figure 2.14 shows the NB-module joint space. The joint space is the space of q_1 and q_2. It can be divided into two aspects because the NB-module inverse geometric model has up to two solutions. Both the areas can cover the entire workspace with the same orientation for the ending platform. There are two limit cases where the



(a) 3D view of workspace



(b) 2D view of workspace

Figure 2.13: NB-module workspace views for $r = 1$ m and $\alpha = \pi/12$ rad $= 15°$

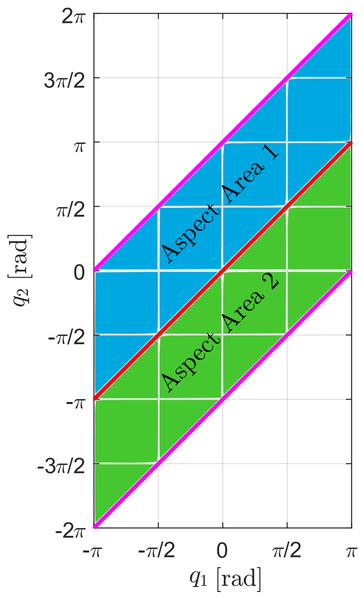


Figure 2.14: NB-module joint space with two aspect areas

module does not have strictly two solutions for a single tip pose. When the ending platform of the module is flat, i.e. $\theta = 0$, there are infinite possible couples of q_1 and q_2. In fact, the only condition that leads to $\theta = 0$ is $q_1 = q_2$. This case is underlined in red in Figs. 2.13a, 2.13b and 2.14. The second limit case is when the tilt reaches its maximum value, i.e. $|\theta| = \pi/6$ rad. Here, there exists only one possible couple of q_1 and q_2 for each ϕ with $|\theta| = \pi/6$ rad. This case happens when $|q_1 - q_2| = \pi$ rad. This case is underlined in magenta in Fig.s 2.13a, 2.13b and 2.14.

2.4 Kinematic model of NB-module

This section presents the kinematic model of the NB-module based on the parameterization defined in Fig. 2.10a. At first, the kinematic Jacobian matrix of the NB-module is computed and explained. Then, the kinematic performance are measured using a kinematic metric, setting the design parameters to specific values. The results are plotted on the workspace and joint space. Finally, a general kinematic performance analysis is performed on the NB-module as a function of its design parameters.

2.4.1 NB-module Jacobian matrix computation

Here, the NB-module kinematic Jacobian matrix $\mathbf{J}_{\text{NB}}$ computation is presented. This matrix is computed as a function of the joint values $\dot{\mathbf{q}} = [q_1, q_2]^\top$. It maps the joints velocities $\dot{\mathbf{q}} = [\dot{q}_1, \dot{q}_2]^\top$ to the NB-module tip twist $\mathbf{t} = \left[\dot{\mathbf{p}}^\top, \boldsymbol{\omega}^\top\right]^\top \in \mathbb{R}^6$ where $\dot{\mathbf{p}} \in \mathbb{R}^3$ and $\boldsymbol{\omega} \in \mathbb{R}^3$ are the linear and angular velocity vectors of frame $\mathcal{F}_3$, respectively. So, the relation between $\mathbf{t}$ and $\dot{\mathbf{q}}$ is

$$\mathbf{t} = \mathbf{J}_{\text{NB}}\,\dot{\mathbf{q}}. \tag{2.8}$$

The NB-module kinematic Jacobian matrix $\mathbf{J}_{\text{NB}}$ is computed in two steps. First of all, the kinematic Jacobian matrix $\mathbf{J}_1(\phi, \theta, r) \in \mathbb{R}^{6\times 2}$ is calculated as a function of the angles $[\phi, \theta]^\top$. This matrix maps the tilt and azimuth angles time derivatives $\left[\dot{\phi}, \dot{\theta}\right]^\top$ to the NB-module tip twist $\mathbf{t}$. The Jacobian $\mathbf{J}_1$ results to be

$$\mathbf{t} = \mathbf{J}_1(\phi, \theta, r)\left[\dot{\phi} \quad \dot{\theta}\right]^\top \tag{2.9}$$

with

$$\mathbf{J}_1(\phi, \theta, r) = \begin{bmatrix} -r\sin\phi\sin\theta & r\cos\phi\cos\theta \\ r\cos\phi\sin\theta & r\sin\phi\cos\theta \\ 0 & -r\sin\theta \\ -\cos\phi\sin\theta & -\sin\phi \\ -\sin\phi\sin\theta & \cos\phi \\ 1-\cos\theta & 0 \end{bmatrix}. \tag{2.10}$$

The matrix $\mathbf{J}_1$ is derived from the geometric model presented in Fig. 2.10a and Eq. (2.5).

Then, the kinematic Jacobian matrix $\mathbf{J}_2(q_1, q_2) \in \mathbb{R}^{2\times 2}$, which maps the joints velocities $\dot{\mathbf{q}} = [\dot{q}_1, \dot{q}_2]^\top$ to the angular velocities $[\dot{\phi}, \dot{\theta}]^\top$, is obtained upon time differentiation of Eq. (2.7), and takes the form

$$[\dot{\phi}, \dot{\theta}]^\top = \mathbf{J}_2(q_1, q_2)\, [\dot{q}_1, \dot{q}_2]^\top \tag{2.11}$$

with

$$\mathbf{J}_2(q_1, q_2) = \frac{1}{2} \begin{bmatrix} 1 & 1 \\ -b & b \end{bmatrix} \tag{2.12}$$

where

$$b = \frac{2 \tan\alpha \cos\left(\dfrac{q_1 - q_2}{2}\right)}{1 + \tan^2\alpha \sin^2\left(\dfrac{q_1 - q_2}{2}\right)}. \tag{2.13}$$

The complete kinematic Jacobian matrix $\mathbf{J}_{\mathrm{NB}}$ of the NB-module is computed as

$$\mathbf{J}_{\mathrm{NB}} = \mathbf{J}_1\, \mathbf{J}_2. \tag{2.14}$$

2.4.2 Kinematic performance analysis for specific design parameter values

Here, the kinematic performance of the module is evaluated and, then plotted on the NB-module workspace and joint space for some specific design parameter values. The kinematic index used to evaluate the NB-module performance is called dexterity η. The dexterity $\eta(\mathbf{J})$ of a generic kinematic Jacobian matrix $\mathbf{J}$ characterizes the kinematic performance of a generic manipulator in a given configuration. It is defined as the inverse of the conditioning number $\kappa(\mathbf{J})$ [ALC92],

$$\kappa(\mathbf{J}) = ||\mathbf{J}||_2\, ||\mathbf{J}^{-1}||_2 \quad \text{and} \quad \eta(\mathbf{J}) = 1/\kappa(\mathbf{J}), \tag{2.15}$$

where $||\mathbf{J}||_2$ is the 2-norm of $\mathbf{J}$. Since the 2-norm of $\mathbf{J}$ is employed, the conditioning number κ can also be defined as the ratio between the bigger $\sigma_{\max}$ and smaller $\sigma_{\min}$ singular values of $\mathbf{J}$.

$$\kappa(\mathbf{J}) = \frac{\sigma_{\max}(\mathbf{J})}{\sigma_{\min}(\mathbf{J})} \quad \text{and} \quad \eta(\mathbf{J}) = \frac{\sigma_{\min}(\mathbf{J})}{\sigma_{\max}(\mathbf{J})}. \tag{2.16}$$

The conditioning number κ is bounded by 1 and ∞. So, the dexterity η is bounded by 0 and 1. The higher η, the better the manipulator dexterity and the better the robot can move along or rotate around all directions. The manipulator reaches an isotropic posture when $\eta = 1$. The smaller η, the worse the manipulator dexterity and the closer to a singularity. When $\eta = 0$, the robot is in a singular configuration and loses one or more degrees of freedom, meaning that a movement/rotation is not possible anymore.

The NB-module is a two degrees of freedom mechanism, providing two rotational movements around the axis $\vec{x}$ and $\vec{y}$. So, the NB-module dexterity η is computed as a function of the kinematic Jacobian matrix $\mathbf{J}_3$ that is a part of $\mathbf{J}_{\mathrm{NB}}$ and maps $\dot{\mathbf{q}}$ to $[\omega_x, \omega_y]^\top$,

$$\begin{bmatrix} \omega_x & \omega_y \end{bmatrix}^\top = \mathbf{J}_3 \dot{\mathbf{q}} \tag{2.17}$$

where

$$\mathbf{J}_3 = \begin{bmatrix} -\cos\phi\sin\theta & -\sin\phi \\ -\sin\phi\sin\theta & \cos\phi \end{bmatrix} \mathbf{J}_2. \tag{2.18}$$

Figures 2.15 and 2.16 show the isocontours of the dexterity η plotted on the NB-module's workspace and joint space, respectively. In this case, the design parameters are again set to $r = 1$ m and $\alpha = \pi/12$ rad $= 15°$. It is apparent that the NB-module

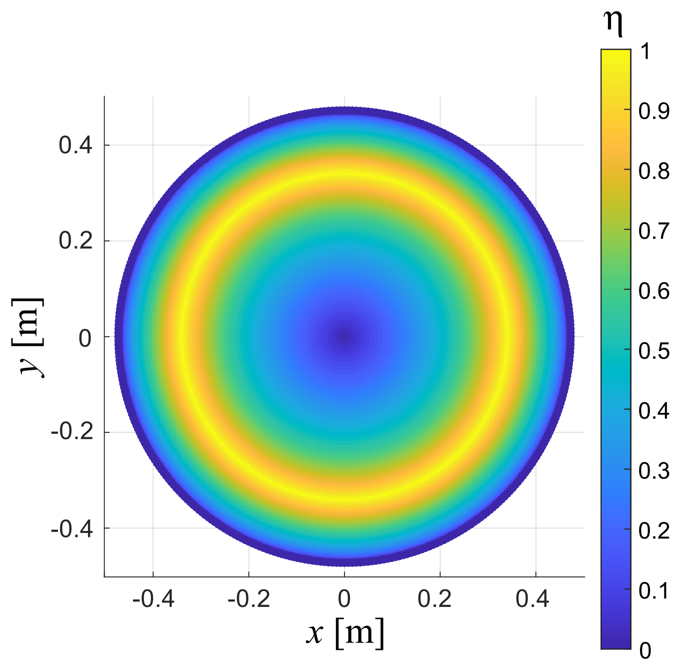


Figure 2.15: Dexterity η shown on NB-module workspace for $r = 1$ m and $\alpha = \pi/12$ rad $= 15°$

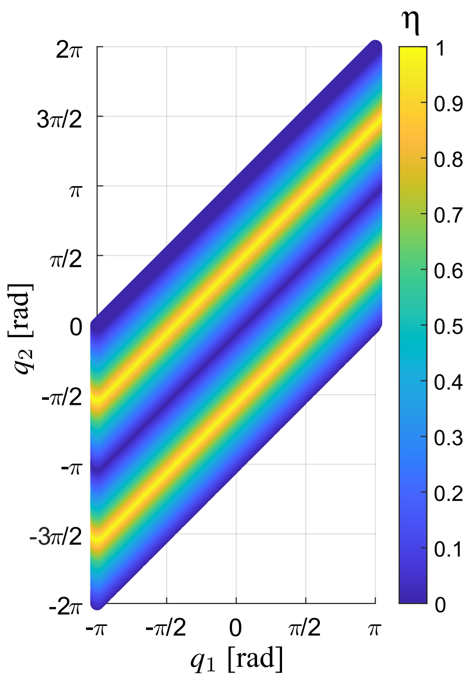


Figure 2.16: Dexterity η shown on NB-module joint space

reaches a singular configuration, i.e. $\eta = 0$, when

$$\begin{cases} \theta = 0 & \text{i.e.} \quad q_1 = q_2 \\ |\theta| = \pi/6 & \text{i.e.} \quad |q_1 - q_2| = \pi \end{cases}. \tag{2.19}$$

Furthermore, the NB-module reaches an isotropic configuration, i.e. $\eta = 1$, when

$$|\theta| = \arctan\left(\pm\frac{\sqrt{2}\ \tan\alpha}{2+\tan^2\alpha}\right) \quad \text{i.e.} \quad |q_1 - q_2| = \pi/2. \tag{2.20}$$

2.4.3 Kinematic performance analysis for generalized design parameter values

In the previous sections, the geometric and kinematic performance are analyzed setting the NB-module design parameter α to $\alpha = \pi/12$ rad $= 15°$. However, this parameter affects both the dimension of the Cartesian workspace and the kinematic performances of the NB-module. Therefore, this section focuses on the effect of α on the NB-module workspace and dexterity. Two different indices are considered in this analysis. The first one is the workspace size and the second one is the global conditioning index H. The global conditioning index H is a performance index based on the distribution of the conditioning number of the kinematic Jacobian matrix $\mathbf{J}_3$ over the entire robot workspace [GA91]. The global conditioning index H is defined as

$$H = \frac{A}{B}, \tag{2.21}$$

where

$$A = \int_W \eta \, dW \quad \text{and} \quad B = \int_W dW, \tag{2.22}$$

which represents the sum of the dexterity η on the workspace W over the area of the workspace W. Since it is easier to perform the integration in the joint space and then transform it into the Cartesian space, A and B can be rewritten as

$$A = \int_{q_1} \int_{q_2} \eta \, |\mathit{\Delta}| \, dq_2 \, dq_1 \quad \text{and} \quad B = \int_{q_1} \int_{q_2} |\mathit{\Delta}| \, dq_2 \, dq_1. \tag{2.23}$$

The absolute value of the determinant of the kinematic Jacobian matrix $|\mathit{\Delta}|$ is necessary to transform the sum from the joint space into the Cartesian workspace. The variable q_1 is integrated in the range $(-\pi, \pi]$ rad while q_2 in $[q_1 - \pi, q_1 + \pi]$ rad. So, $\mathit{\Delta}$ and η take the form of

$$\mathit{\Delta} = \frac{4 \sin(q_1 - q_2) \tan^2(\alpha)}{(\tan^2(\alpha) - \cos(q_1 - q_2) \tan^2(\alpha) + 2)}, \tag{2.24}$$

and

$$\eta = |\sin(q_1 - q_2)|. \tag{2.25}$$

Interestingly, the dexterity η is independent of the value of α, while the determinant $\mathit{\Delta}$ is dependent. So, the isocontours of the dexterity on the joint space, shown in Fig. 2.16, are not affected by a changing value of α. On the other hand, $\mathit{\Delta}$ depends from α and, as follows, the isocontours of the dexterity in the Cartesian workspace change with α.

Figure 2.17 shows the graph of the global conditioning index H as a function of α. It should be noted that H remains high as long as $\alpha \leq \pi/4$ rad. Figure 2.18 depicts the surface area S of the NB-module workspace as a function of α. The design parameter r is set to 1 m. The larger α, the larger S and the closer the NB-module workspace shape to a sphere. Figure 2.19 shows the isocontours of the dexterity for three different values of α.

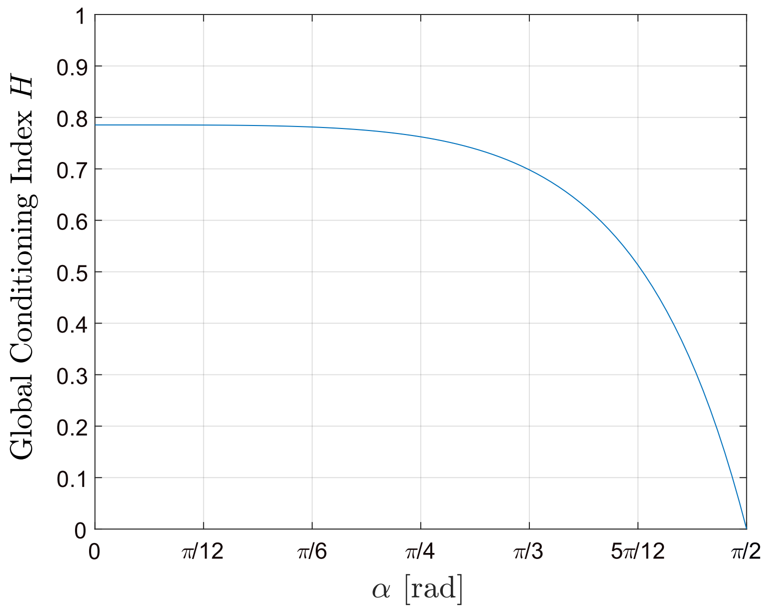


Figure 2.17: Global conditioning index H of kinematic Jacobian matrix $\mathbf{J}_3$ as a function of the NB-module tube slope α

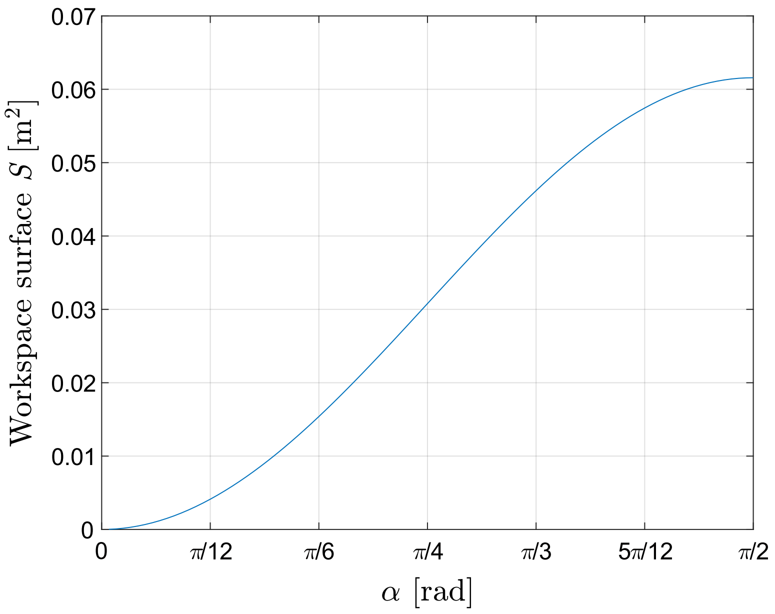
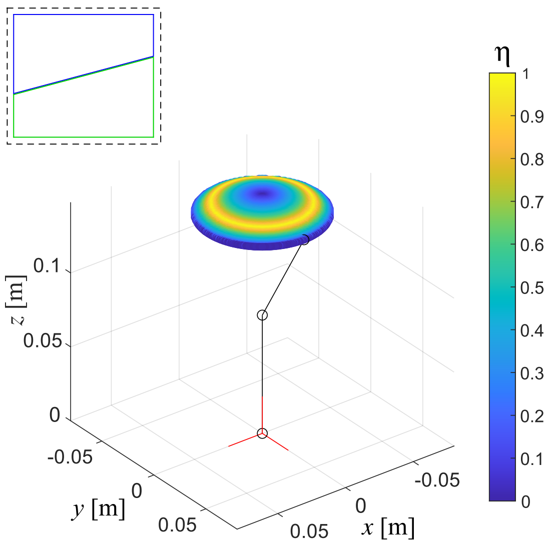
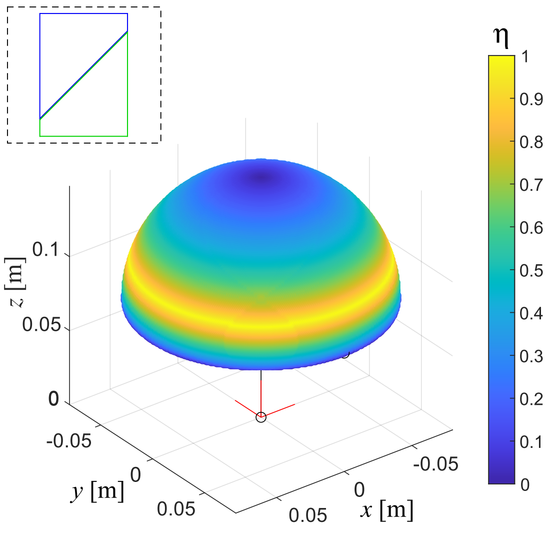


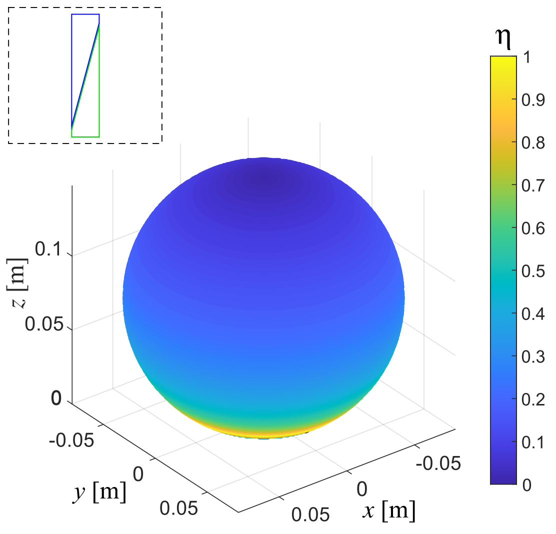
Figure 2.18: Surface area S of the NB-module workspace as a function of the NB-module tube slope α



(a) $\alpha = \pi/12$ rad $= 15°$



(b) $\alpha = \pi/4$ rad $= 45°$



(c) $\alpha = 5\pi/12$ rad $= 75°$

Figure 2.19: Dexterity η shown in the NB-module workspace and schematics of the NB-module for three values of tube slope α, origin axis in red. In the top-left corner, the configuration of *Tube 1* and *Tube 2* for the corresponding tube slope α.

2.5 Conclusions of the NB-module analyses

This chapter presented the NB-module design and features. This mechanism was built to ensure stability and flexibility to robotic manipulators. One of the most interesting feature of the NB-module is the constant velocity joint placed in the internal kinematic chain which leads to the zero-torsion characteristic. Then, its geometric and kinematic models were investigated, plotting the workspace and joint space and its kinematic performance for specific design parameter values. As proved, the NB-module reaches isotropic configurations when $|q_1 - q_2| = \pi/2$ rad and singularities when $|q_1 - q_2| = \pi$ rad or $q_1 = q_2$. Finally, the NB-module kinematic performance was analyzed as a function of its design parameters using the global conditioning index H. The tube height variable r has no effect on the NB-module kinematic performance. On the contrary, the tube slope α affects the index H. It is shown that α equal to $\pi/4$ rad is the best trade-off between the global conditioning index H and workspace surface S. Part of the work presented in this chapter was published in [GLC+21, GCSL23a].

Chapter 3

Task Priority Based Inverse Kinematics of Redundant Manipulators

This chapter describes a task priority based kinematic control algorithm for redundant manipulators and proposes some tasks for the robot kinetostatic performance improvement. These algorithm are used to kinematically control a redundant manipulator to track a set of different trajectories while optimizing the kinetostatic performance. It is presented in Chapter 1 and shown in Fig. 1.7 on page 11. Since this robot has a high amount of degrees of freedom, it is useful to test the TPIK algorithm and kinetostatic optimization task result while tracking different trajectories. The chapter is organized as follow. Section 3.1 introduces the topic of kinematic control and trajectory tracking in case of redundancy via task augmentation. Section 3.2 presents the TPIK algorithm and its features. Section 3.3 describes the kinetostatic indices employed to improve the robot configuration and how to include these indices in the TPIK algorithm. Section 3.4 presents the trajectory tracking test of the NB-R1 for a series of trajectories. Section 3.5 analyzes and compares the obtained kinetostatic results with and without performance optimization. The conclusions related to the trajectory tracking results using the NB-R1 robot are described in Section 3.6.

3.1 Task priority based kinematic redundancy resolution

In trajectory tracking applications, the kinematic redundancy of robotic manipulators can be viewed as a possible way to improve the machine abilities and performance [GST19]. As already introduced by Section 1.3.1, the kinematic redundancy can be used for solving several tasks simultaneously via task augmentation [SW95, SKK08]. The concept behind task enhancement consists of adding new objectives to the main task that the robot has to perform. This augmentation is helpful to exploit the degrees of freedom advance in redundant robots. It is performed in the task-space and the kinematic Jacobian matrix used to solve the main task is extended, including new rows related to the additional task solutions. The inverse, or pseudo-inverse, of the extended Jacobian provides a joint velocity solution to satisfy the simultaneous tasks [SKK08]. Different priorities can be added to the tasks based on their relevance to ensure the satisfaction of more important tasks. In this case, the solution for each task is searched in the null space of the higher priority tasks [ODAS15]. The main problem of this technique lies in the algorithmic singularities [SKK08, FDL14]. This type of singularity differs from the kinematic one and may arise even when all the considered Jacobian matrices are full rank. The algorithmic singularities appear when the extended kinematic problem is singular and the desired task velocity cannot be realized because of an incompatibility between all the tasks.

Some early works on task priority based kinematic resolution techniques were proposed in [NHY87, SS91, SK05]. These works have no defense against the kinematic and algorithmic singularities. Later, more complex methods were developed to deal with both these types of singularities. In [FDLK12], the authors proposed a so-called Saturation in the Null Space (SNS) algorithm that implements a predictive prioritizing technique for multiple tasks. This method was designed to handle the joint-space limits in the context of a single task and extended to handle prioritized tasks. The SNS algorithm shows satisfying performance, never violating the hard bounds, preserving the correct task priority hierarchy even in unfeasible cases and performing an automatic task scaling. Then, the SNS algorithm was modified in a constrained quadratic programming problem to minimize both the joint velocity norm and the task scaling, as presented in [FDL13]. The new algorithm was named Optimal Saturation in the Null Space (Opt-SNS). However, this method can still suffer from algorithmic singularities. In [FDL14], a new inverse kinematic solver, called Reverse Priority (RP), was developed to avoid the algorithmic singularities. The RP method computes joint motion contributions from the lowest-priority task to the primary task. It employs a special projection matrix to maintain the correct priority order. In [EMW14], the authors present a hierarchical quadratic programming control algorithm used to find a solution to multiple and antagonistic objectives for humanoid robot motion generation. This quadratic programming algorithm aimed to reduce the time consumption required to solve a prioritized list of tasks. Another multiple tasks control framework is presented in [DLCA19], called Set-Based Multi-Task Priority Inverse Kinematics Framework. This method can handle both equality and inequality tasks using a priority system. For inequality tasks, the control objective has to keep the task value inside a specific interval. Additionally, the proposed multiple task method can accommodate optimization tasks that usually have lower priorities. One of the main drawbacks of all the mentioned task priority-based algorithms is that activating or deactivating one or more tasks can generate discontinuities in the joint velocity solutions [SC16].

In [SC16, SCWA18, SCWA19], the authors proposed a new kinematic control algorithm for redundant robots, named Task Priority Inverse Kinematic (TPIK). This algorithm finds the robot joint velocities that better fits the set of prioritized tasks. Each task is solved by searching for a solution in the null space of all the higher priority tasks. This control framework has a mechanism of prevention for kinematic and algorithmic singularities. Moreover, it can activate and deactivate one or more tasks without generating algorithmic discontinuities. This feature is handy for deactivating those tasks that do not require to be fulfilled at a specific moment, avoiding an over-constrain of the robotic system. In this case, it is employed to deal with the NB-R1 high redundancy for tracking some machining trajectories. In addition to the tracking trajectory task, some optimization tasks are included in the algorithm to improve the kinetostatic performance of the robot.

3.2 Task priority based inverse kinematic algorithm

This section describes the kinematic control algorithm used to kinematically control in simulation the NB-R1 while performing some tracking trajectory test. Before introducing the kinematic control algorithm, some definitions are recalled from the work [SCWA18]. The vector $\mathbf{q} \in \mathbb{R}^n$ is the joint variable vector, describing the arm configuration, where n is the number of joints. The joint velocities are collected in the vector $\dot{\mathbf{q}} \in \mathbb{R}^n$.

The notion of control objectives defines the goals of the robot. A control objective is a scalar variable $\mathrm{x}(\mathbf{q})$ computed as a function of the robot configuration vector $\mathbf{q}$ and represents the state of one task. A control objective can be of two different types, equality and inequality. Equality control objectives aim to satisfy the relationship $\mathrm{x}(\mathbf{q}) = \mathrm{x}_0$. Inequality control objectives take the form $\mathrm{x}(\mathbf{q}) \leq \mathrm{x}_M$, or $\mathrm{x}(\mathbf{q}) \geq \mathrm{x}_m$, or both simultaneously, where x_m and x_M are the lower and upper bounds of the variable $\mathrm{x}(\mathbf{q})$ [SCWA18]. Control objectives can be divided into categories depending on their scope:

- system safety objectives, e.g. joint limits or obstacle avoidance,
- action oriented objectives, e.g. reaching a desired pose or following a desired trajectory,
- optimization objectives, e.g. minimizing the joint velocities or optimizing the kinetostatic performance metrics.

This division is purely semantic and helps identify the correct priority level for each control objective. Then, each scalar control objective is associated with a feedback reference rate $\dot{\bar{\mathrm{x}}}$. The closed-loop rate control law drives the actual variable $\mathrm{x}(\mathbf{q})$ to the desired point x^* with the associated feed-forward changing rate $\dot{\mathrm{x}}^*$ and is defined as

$$\dot{\bar{\mathrm{x}}} = \lambda(\mathrm{x}^* - \mathrm{x}(\mathbf{q})) + \dot{\mathrm{x}}^*, \tag{3.1}$$

where λ is a positive gain related to the target convergence rate. The actual derivative of x is defined as a function of the joint velocity vector $\dot{\mathbf{q}}$ as follows:

$$\dot{\mathrm{x}}(\mathbf{q}, \dot{\mathbf{q}}) = \mathbf{J}_{\text{task}}(\mathbf{q})\dot{\mathbf{q}} = \begin{bmatrix} \frac{\partial \mathrm{x}}{\partial q_1} & \dots & \frac{\partial \mathrm{x}}{\partial q_n} \end{bmatrix} \dot{\mathbf{q}}, \tag{3.2}$$

where $\mathbf{q} = [q_1 \ \dots \ q_n]$.

An activation function $a^i(\mathrm{x}) \in [0, 1]$ is associated to each control objective $\mathrm{x}(\mathbf{q})$ and represents whether the objective is relevant or not in a given time instant. The tasks associated with inequality control objectives are relevant only when the scalar variable $\mathrm{x}(\mathbf{q})$ is near or out of the validity region. So, the activation function assumes zero values within the validity region of the associated inequality objective and one when it is not, with a smooth transition between the two states. For tasks associated with equality control objectives, the activation function is set to $a^i(\mathrm{x}) = 1$ because they always need to be active.

A specific priority is assigned to each task based on the relative importance of each objective. The meaning of the priority is that the highest priority tasks are solved first using the available robot degrees of freedom and are not affected by the lower priority ones. Hence, lower priority tasks are solved if enough robot degrees of freedom remains. When two or more tasks have the same priority, they are grouped in a multidimensional control task. A specific list of prioritized tasks is called control action $\mathscr{A}$.

With the previous definitions, the following quantities associated with each priority level in a control action $\mathscr{A}$ can be defined [SCWA19]:

- $\bar{\dot{\mathbf{x}}}_k = [\bar{\dot{\mathrm{x}}}_{1,k},\ \bar{\dot{\mathrm{x}}}_{2,k},\ \dots\ ,\ \bar{\dot{\mathrm{x}}}_{m_k,k}]^\top$ is the vector collecting all the reference rates of the scalar control tasks, where m_k is the number of scalar tasks for the priority level k.
- $\mathbf{J}_k$ is the Jacobian matrix associated with the k^{th} task vector $[\dot{\mathrm{x}}_{1,k},\ \dots\ ,\ \dot{\mathrm{x}}_{m_k,k}]^\top$ with respect to the joint velocity vector $\dot{\mathbf{q}}$.
- $\mathbf{A}_k = \mathrm{diag}(a_{1,k},\ \dots\ ,\ a_{m_k,k})$ is a diagonal matrix of the activation functions.

To find the system velocity reference vector $\bar{\dot{\mathbf{q}}}$ that meets the priority requirements of a given action, the TPIK algorithm solves a sequence of nested minimization problems

$$S_k = \arg \underset{\bar{\dot{\mathbf{q}}} \in S_{k-1}}{\mathrm{R}-\min} ||\mathbf{A}_k(\bar{\dot{\mathbf{x}}}_k - \mathbf{J}_k\bar{\dot{\mathbf{q}}})||^2, \tag{3.3}$$

where S_{k-1} is the manifold of all the previous priority level solutions. The notation $\mathrm{R}-\min$ highlights that each minimization is performed through specific regularized space projections to implement priorities among the tasks defined in [SC16]. In addition to Eq. (3.3), other regularization costs are included. These regularization costs avoid discontinuities in the system velocity vector due to kinematic and algorithmic singularities. In [SC16], the authors fully describe these regularization costs that are not analyzed here.

A significant advantage of the TPIK algorithm is the use of the activation functions to handle inequality control objectives without over-constraining the system. Both equality and inequality control require a certain amount of robot degrees of freedom specified by the associated task. When an inequality task is inside its validity region, the activation function goes to zero, therefore not consuming any degrees of freedom. So, safety tasks, like joint limits, can be placed at the top of the hierarchy without over-constraining the system.

Finally, the TPIK algorithm adopts another continuous sigmoidal function $a^P(\mathbf{P})$ to perform a smooth activation/deactivation transition between two actions. This function is related to the vector $\mathbf{P}$ that includes the previous and current executed actions and the time elapsed in the actual step. This function $a^P(\mathbf{P})$ is used together with $a^i(\mathrm{x})$. More details are presented in [SC16].

3.3 Proposed tasks for robot kinetostatic performance optimization

In the proposed case, the TPIK algorithm employs some tasks based on kinetostatic performance indices to optimize the robot configuration while performing the desired application. These indices are dexterity, manipulability and robot transmission ratio. They are defined starting from the kinematic Jacobian matrix $\mathbf{J}_e$ that relates the end-effector velocity with respect to the robot base. The kinematic Jacobian matrix $\mathbf{J}_e$ can be written as

$$\mathbf{t} = \begin{bmatrix} \dot{\mathbf{p}} \\ \boldsymbol{\omega} \end{bmatrix} = \mathbf{J}_e(\mathbf{q})\dot{\mathbf{q}} = \begin{bmatrix} \mathbf{J}_l(\mathbf{q}) \\ \mathbf{J}_a(\mathbf{q}) \end{bmatrix} \dot{\mathbf{q}}, \tag{3.4}$$

where $\mathbf{t} = \begin{bmatrix} \dot{\mathbf{p}}^\top, \boldsymbol{\omega}^\top \end{bmatrix}^\top \in \mathbb{R}^6$ is the robot end-effector twist, with $\dot{\mathbf{p}} \in \mathbb{R}^3$ and $\boldsymbol{\omega} \in \mathbb{R}^3$ the linear and angular velocity vectors of the end-effector, respectively. Since the kinematic Jacobian matrix $\mathbf{J}_e$ contains non-homogeneous terms, namely linear and angular, it needs to be weighted to compute the kinetostatic performance indices correctly. The weighting of $\mathbf{J}_e$ employs the characteristic length L that was introduced in [Ang92] to solve the absence of dimensional homogeneity in the kinematic Jacobian matrix entries and is computed in [KA05]. To weight $\mathbf{J}_e$, the revolute joint columns of the linear kinematic Jacobian matrix part are divided by L. The weighted kinematic Jacobian matrix is written as $\mathbf{J}_w$. The weighting is a critical issue when analyzing the kinetostatic performance of $\mathbf{J}_e$ [KAW15, ZKA12]. In the rest of this Section, the three kinetostatic indices are presented and their Jacobian matrix is explained.

3.3.1 Manipulability

The manipulability is an index that measures the kinematic abilities of the robotic system through its weighted kinematic Jacobian matrix $\mathbf{J}_w$ [Yos85]. The manipulability of a manipulator is defined as

$$\mu = \sqrt{\det(\mathbf{J}_w \mathbf{J}_w^\top)}, \tag{3.5}$$

and amounts to the product of all the singular values of $\mathbf{J}_w$. The higher the manipulability value, the larger the manipulability hyper-ellipsoid and the better the kinematic performance of the mechanism [Ang03]. It should be noted that the manipulator reaches a kinematic singularity when μ vanishes.

The derivative of the manipulability as a function of the joint variables is explained in [Par00] and used in [MKYC02]:

$$\frac{\partial \mu}{\partial q_i} = \mu \ \mathrm{trace}\left\{ \frac{\partial \mathbf{J}_w}{\partial q_i} \mathbf{J}_w^+ \right\}, \tag{3.6}$$

where the matrix $\mathbf{J}_w^+$ is the pseudo-inverse of the weighted kinematic Jacobian matrix [SC16]. Hence, the manipulability Jacobian matrix $\mathbf{J}_\mu$ as a function of the joint variables is:

$$\mathbf{J}_\mu = \begin{bmatrix} \frac{\partial \mu}{\partial q_1} & \dots & \frac{\partial \mu}{\partial q_n} \end{bmatrix}, \tag{3.7}$$

where n, which represents the number of columns of $\mathbf{J}_w$, is the dimension of joint space.

3.3.2 Dexterity

The dexterity index was already presented in Section 2.4.2 to analyze the NB-module kinematic performance. Here, the dexterity $\eta(\mathbf{J}_w)$ characterizes the kinematic performance of a complete manipulator in a given configuration. It is defined as the inverse of the conditioning number $\kappa(\mathbf{J}_w)$ of its weighted kinematic Jacobian matrix $\mathbf{J}_w$ [ALC92]:

$$\kappa(\mathbf{J}_w) = ||\mathbf{J}_w||_2 \ ||\mathbf{J}_w^{-1}||_2 \tag{3.8}$$

and

$$\eta(\mathbf{J}_w) = 1/\kappa(\mathbf{J}_w). \tag{3.9}$$

To recall, the index η is bounded by 0 and 1. The higher η, the better the manipulator dexterity. The manipulator reaches an isotropic posture when $\eta = 1$. The smaller η, the worse the manipulator dexterity and the closer to a singularity. Moreover, η can be defined as the ratio between the smallest and highest singular values of $\mathbf{J}_w$ indicating how close the manipulability hyper-ellipsoid is to being a hyper-sphere [PC06].

The formula proposed in Eq. (3.9) can not be derived because it is not in an analytical form. So, the Frobenius norm of $\mathbf{J}_w$ can be used [RCC08] to obtain the analytical expression of η:

$$\eta(\mathbf{J}_w) = \frac{m}{\sqrt{\mathrm{trace}(\mathbf{J}_w \mathbf{J}_w^\top) \ \mathrm{trace}[(\mathbf{J}_w \mathbf{J}_w^\top)^{-1}]}}, \tag{3.10}$$

where m, which represents the number of rows of $\mathbf{J}_w$, is the dimension of the task space. The following definitions are introduced to improve the readability of the equations:

$$\gamma_1 \triangleq \sqrt{\text{trace}(\mathbf{J}_w \mathbf{J}_w^\top)} \tag{3.11}$$

and

$$\gamma_2 \triangleq \sqrt{\text{trace}[(\mathbf{J}_w \mathbf{J}_w^\top)^{-1}]}, \tag{3.12}$$

where $\triangleq$ is the define operator. With these definition, it follows that

$$\eta(\mathbf{J}_w) = \frac{m}{\gamma_1(\mathbf{J}_w)\,\gamma_2(\mathbf{J}_w)}. \tag{3.13}$$

Then, the dexterity Jacobian matrix is determined to relate the velocity rate of η with respect to the joint velocity vector $\dot{\mathbf{q}}$. The Frobenius formula used in Eq. (3.10) expresses η as a function of joint position vector $\mathbf{q}$ in an analytical way allowing its derivation. So, the derivative of Eq. (3.10) with respect to each joint position $q_i \in \mathbf{q}$ is

$$\frac{\partial \eta}{\partial q_i} = -\eta \left(\frac{\partial \gamma_1}{\partial q_i} \frac{1}{\gamma_1} + \frac{1}{\gamma_2} \frac{\partial \gamma_2}{\partial q_i} \right), \tag{3.14}$$

where

$$\frac{\partial \gamma_1}{\partial q_i} = \frac{1}{\gamma_1} \text{trace} \left\{ \mathbf{J}_w \frac{\partial \mathbf{J}_w^\top}{\partial q_i} \right\} \tag{3.15}$$

and

$$\frac{\partial \gamma_2}{\partial q_i} = \frac{1}{\gamma_2} \text{trace} \left\{ -\mathbf{J}_w \frac{\partial \mathbf{J}_w^\top}{\partial q_i} (\mathbf{J}_w \mathbf{J}_w^\top)^2 \right\}. \tag{3.16}$$

In conclusion, the dexterity Jacobian matrix $\mathbf{J}_\eta$ as a function of the joint variables is:

$$\mathbf{J}_\eta = \begin{bmatrix} \frac{\partial \eta}{\partial q_1} & \dots & \frac{\partial \eta}{\partial q_n} \end{bmatrix}, \tag{3.17}$$

where n is the number of columns of $\mathbf{J}_w$ and dimension of joint space.

3.3.3 Robot transmission ratio

The robot transmission ratio (RTR) $\rho(\mathbf{J}_w)$ quantifies the effectiveness of the actuator force in producing a prescribed robot motion [ZKA12]. It corresponds to the angle between the joint velocity $\dot{\mathbf{q}}$ and torque $\boldsymbol{\tau}$ vectors in the joint space and is defined as

$$\rho = \frac{|\boldsymbol{\tau}^\top \dot{\mathbf{q}}|}{||\boldsymbol{\tau}|| \, ||\dot{\mathbf{q}}||} = |\cos\angle(\boldsymbol{\tau}, \dot{\mathbf{q}})|. \tag{3.18}$$

This metric is bounded between 0 and 1. In case of kinetostatic redundancy, ρ can be expressed in terms of the end-effector twist $\mathbf{t}$ and the wrench $\mathbf{w}$ applied to it, leading to

$$\rho = \frac{|\mathbf{w}^\top \mathbf{t}|}{||\mathbf{J}_w^\top \mathbf{w}|| \, ||\mathbf{J}_w^+ \mathbf{t}||}, \tag{3.19}$$

where $\mathbf{t}$ is the robot end-effector twist defined in Eq. (3.4) and $\mathbf{w} = [\mathbf{f}^\top, \mathbf{m}^\top]^\top$ is the wrench that collects the forces $\mathbf{f}$ and moments $\mathbf{m}$ exerted by the environment on the end-effector. To ensure that ρ is dimensionless, the linear part $\dot{\mathbf{p}}$ in $\mathbf{t}$ and the moment $\mathbf{m}$ in $\mathbf{w}$ are divided by the characteristic length L.

The RTR Jacobian matrix is obtained upon differentiation of Eq. (3.19) with respect to each joint position $q_i \in \mathbf{q}$:

$$\frac{\partial \rho}{\partial q_i} = \rho \, \frac{\mathbf{w}^\top \mathbf{J}_w \frac{\partial \mathbf{J}_w^\top}{\partial q_i} \mathbf{w} ||\mathbf{J}_w^+ \mathbf{t}||^2 - ||\mathbf{J}_w^\top \mathbf{w}||^2 \mathbf{t}^\top \mathbf{J}_w^{+\top} \frac{\partial \mathbf{J}_w^+}{\partial q_i} \mathbf{t}}{(||\mathbf{J}_w^\top \mathbf{w}|| \, ||\mathbf{J}_w^+ \mathbf{t}||)^2}, \tag{3.20}$$

where the values of the end-effector twist $\mathbf{t}$ and wrench $\mathbf{w}$ are known from the trajectory planning. The pseudo-inverse weighted kinematic Jacobian matrix derivative $\partial \mathbf{J}_w^+ / \partial q_i$ is defined in [GP73]. The RTR Jacobian matrix $\mathbf{J}_\rho$ as a function of the joint variables is:

$$\mathbf{J}_\rho = \begin{bmatrix} \frac{\partial \rho}{\partial q_1} & \dots & \frac{\partial \rho}{\partial q_n} \end{bmatrix}, \tag{3.21}$$

where $n = \text{columns}(\mathbf{J}_w)$ is the dimension of joint space.

3.4 Trajectory tracking test in simulation description

This section describes the tests performed in a computer simulation on the NB-R1 robot, shown in Fig. 1.7 on page 11. The test consists in making the NB-R1 track different trajectories with and without the tasks related to dexterity, manipulability and RTR collecting their values. Both the dexterity and the manipulability are kinematic performance indices. Maximizing them simultaneously forces the manipulability hyper-ellipsoid to be as big as possible and close to a hyper-sphere. So, ideally, the robot will be able to move with the same higher velocity amplification factor in all directions while reducing actuator velocity limits. Moreover, the

RTR index forces the joint velocity and torque vectors to align, making the robot movement along the desired direction more effective. After collecting the metric values on each trajectory, these are compared to demonstrate the benefit of using the optimization tasks and identify the best robot configuration series to follow each trajectory.

The NB-R1 has to track four trajectories of the same shape and size. Figure 3.1 shows the NB-R1 next to the four trajectories. Two of them are oriented horizontally and the others are vertical. These trajectories describe a cubic area whose side are 0.5 m $\times$ 0.5 m centered in $(x, y, z) = (0.0, 1.05, 0.45)$. The machining tool is shown in the top right corner of Fig. 3.1. The tool ending part is rotated of 45° around the red point. This allows the robot to reach all the points on each trajectory. The trajectories are planned to cut a squared shape using a machining tool and the measures are shown in Fig. 3.2. The tool trajectory is divided in four parts (a), (b), (c) and (d). Figure 3.2 also shows the orientation of the velocity vector $\vec{\mathbf{v}}$ and the tangential and radial force vectors $\vec{\mathbf{f}}_t$ and $\vec{\mathbf{f}}_r$ applied on the machining tool. The magnitudes of $\vec{\mathbf{v}}$, $\vec{\mathbf{f}}_t$ and $\vec{\mathbf{f}}_r$ are constant along the entire trajectory. The profiles of $\vec{\mathbf{v}}$, $\vec{\mathbf{f}}_t$ and $\vec{\mathbf{f}}_r$ are depicted in Fig. 3.3. The gravity force and the cutting one along axis $\vec{z}_p$ are neglected in this work. The details about NB-R1 and trajectory features are in Tables 3.1 and 3.2, respectively. The details of the machine and the implementation are given

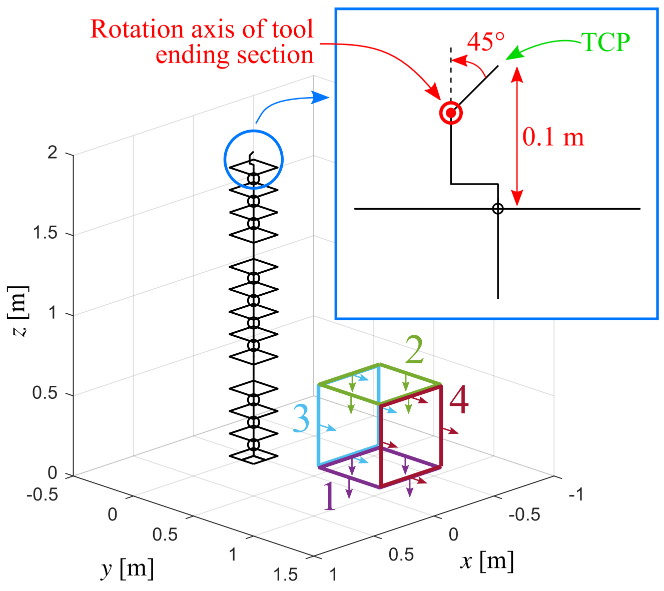


Figure 3.1: Schematics of the NB-R1 with the four trajectories to track in 3D space. In the top right corner, the machining tool attached to the NB-R1 end-effector used in the cutting phase. Tool contact point (TCP) highlighted in green. Tool ending section rotated around red point of 45°.

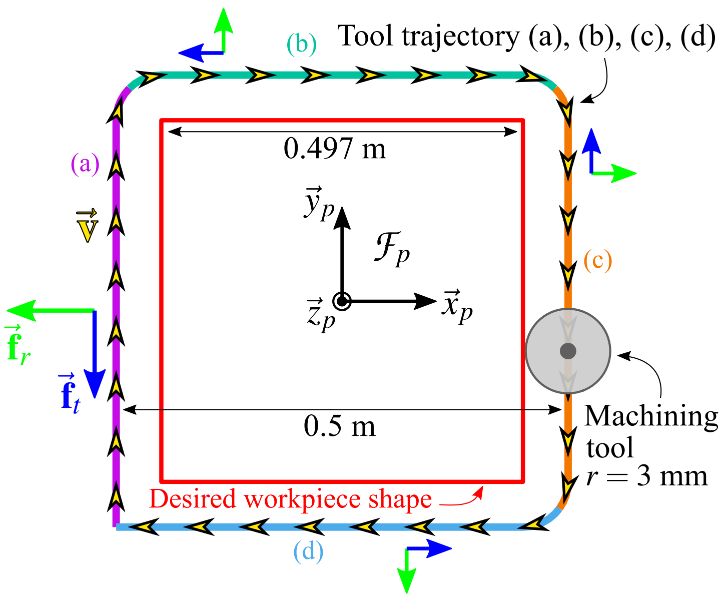
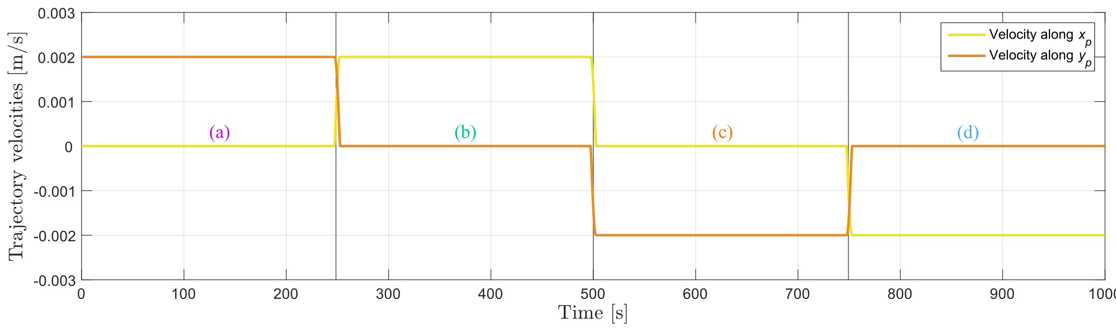
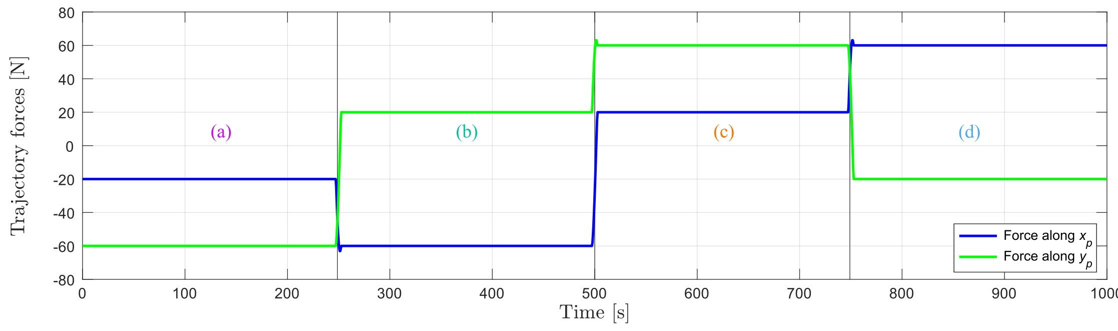


Figure 3.2: Measures of the desired workpiece, machining tool and trajectory tool with workpiece frame $\mathcal{F}_p$. Orientation of velocity vector $\vec{\mathbf{v}}$ (yellow) plus tangential and radial force vectors $\vec{\mathbf{f}}_t$ and $\vec{\mathbf{f}}_r$ (blue and green). The tool trajectory is divided into four parts (a), (b), (c) and (d).



(a) Velocity profiles



(b) Cutting force profiles

Figure 3.3: Velocity and cutting force profiles applied to the machining tool in frame $\mathcal{F}_p$. Each sector is labeled (a), (b), (c) and (d) to match the corresponding trajectory part.

Table 3.1: Robot dimensions plus joint velocity and acceleration limits

NB-module half height r	0.07 m
NB-module tube slope α	15°
Link length	0.2 m
Tool height	0.1 m
Tool offset	45°
Robot + tool total height	1.9 m
Max/min joint velocity	±1.0 rad/s
Max absolute joint acceleration/deceleration	2.0 rad/s^2

Table 3.2: Test trajectory details, velocities and forces exerted on end-effector and time for tracking entire trajectory

Square side	0.5 m
Steps	401
Magnitude velocity vector $\|\|\vec{\mathbf{v}}\|\|_2$	0.002 m/s
Magnitude tangential force vector $\|\|\vec{\mathbf{f}}_t\|\|_2$	60 N
Magnitude radial force vector $\|\|\vec{\mathbf{f}}_r\|\|_2$	20 N
Time	1000 s

Table 3.3: Machine and test implementation details

Operating System	Linux
CPUs number	4
CPU model	Intel Core i7 10th Gen, 1.30GHz
Language	C++
Control frequency	10Hz
Time to track one trajectory	5 s

in Table 3.3. The NB-R1 features, trajectory size and velocity/force magnitudes were provided by Nimbl'Bot.

A convex combination ϵ of the three kinetostatic performance indices is used to rate and compare the NB-R1 kinetostatic performance. It should be noted that η and ρ are bounded between $[0, 1]$ whereas μ is not bounded, $[0, \infty)$. So, it is necessary to bound μ in the range $[0, 1]$ before writing the convex combination. A new index called bounded manipulability ν is defined as:

$$\nu = 1 - \frac{1}{1+\mu}. \tag{3.22}$$

When $\mu = 0$ then $\nu = 0$, and when $\mu = \infty$ then $\nu = 1$. Now, η, ν and ρ are all bounded between 0 and 1 and can then be used in a convex combination:

$$\epsilon(\eta, \nu, \rho) = \lambda_1\eta + \lambda_2\nu + \lambda_3\rho, \tag{3.23}$$

where λ_1, λ_2 and λ_3 are scaling factors. Since all the kinetostatic performance indices are valid in the same range, the weighting factors λ_1, λ_2 and λ_3 are selected equal to $= 1/3$. So, ϵ becomes valid in the same range $[0, 1]$. When $\epsilon = 1$, the robot is in an isotropic configuration, the manipulability hyper-ellipsoid becomes a hyper-sphere and the angle between the joint velocity and torque vectors tends to $0°$.

Four different actions are used in the simulation. When the kinetostatic optimization tasks, namely dexterity, manipulability and RTR, are deactivated, $\mathscr{A}_1$ is the action used to reach the starting pose and $\mathscr{A}_2$ to follow the trajectory. When the

Table 3.4: Details about the task names, control objective types, and hierarchy levels. Symbol (E) represents the equality control objective tasks and (I) the inequality ones. The last four columns list the hierarchy level for each task in actions $\mathscr{A}_1$ (Reach Pose), $\mathscr{A}_2$ (Follow Trajectory), $\mathscr{A}_3$ (Reach Pose Optimized) and $\mathscr{A}_4$ (Follow Trajectory Optimized). When symbol "/" is used, it means that a task is not present in the action and has no hierarchy level.

			Hierarchy levels			
Task	Category	Type	$\mathscr{A}_1$	$\mathscr{A}_2$	$\mathscr{A}_3$	$\mathscr{A}_4$
End-Effector Pose	action oriented	E	1st	/	1st	/
End-Effector Velocity	action oriented	E	/	1st	/	1st
Dexterity	optimization	I	/	/	2nd	2nd
Manipulability	optimization	I	/	/	2nd	2nd
RTR	optimization	I	/	/	2nd	2nd

optimization tasks are activated, $\mathscr{A}_3$ brings the robot to the starting pose and $\mathscr{A}_4$ follows the trajectory. Table 3.4 shows the task details and their hierarchy inside each action.

The tests are developed as follows. The robot is started from a random configuration and reaches the starting point on one trajectory. From here, it tracks the entire trajectory and collects dexterity, bounded manipulability and RTR values at

Algorithm 3.1 Procedure to collect kinetostatic optimized and not optimized results during machining operations

```
Require: Actions $\mathscr{A}_1$ = Reach Pose, $\mathscr{A}_2$ = Follow Trajectory, $\mathscr{A}_3$ = Reach Pose
  Optimized and $\mathscr{A}_4$ = Follow Trajectory Optimized, details in Table 3.4.
Variables: Number of trajectories ($m_t$) is 4 and number of repetitions ($m_r$) is 100.
  Variable $t$ is time instant.
 1: for $i := 1 \to m_t$ do
 2:     for $j := 1 \to m_r$ do
 3:         Randomly initialize starting robot configuration vector $\mathbf{q}_0$.
 4:         Save configuration vector $\mathbf{q}_0$ saved.
 5:         Load $i^{\text{th}}$ trajectory.
 6:         while End-effector not on $i^{\text{th}}$ trajectory starting pose do
 7:             Run action $\mathscr{A}_1$ to reach $i^{\text{th}}$ trajectory starting pose.
 8:         end while
 9:         while End-effector not on $i^{\text{th}}$ trajectory ending pose do
10:             Run action $\mathscr{A}_2$ to move robot to next pose at $t$.
11:             Compute $\epsilon(\mathbf{q}, t)$ for robot configuration $\mathbf{q}$ at $t$.
12:             Save not optimized $\epsilon(\mathbf{q}, t)$, $\eta(\mathbf{q}, t)$, $\mu(\mathbf{q}, t)$, $\rho(\mathbf{q}, t)$ for $j^{\text{th}}$ repetition.
13:         end while
14:         Restart robot in $\mathbf{q}_0$, generated at step 3.
15:         while End-effector not on $i^{\text{th}}$ trajectory starting pose and $\dot{\eta}$,$\dot{\nu}$,$\dot{\rho} > \delta$ do
16:             Run action $\mathscr{A}_3$ to reach $i^{\text{th}}$ trajectory starting pose.
17:         end while
18:         while End-effector not on $i^{\text{th}}$ trajectory ending pose do
19:             Run action $\mathscr{A}_4$ to move robot to next pose at time instant $t$.
20:             Compute $\epsilon(\mathbf{q}, t)$ for robot configuration $\mathbf{q}$ at time instant $t$.
21:             Save optimized $\epsilon(\mathbf{q}, t)$, $\eta(\mathbf{q}, t)$, $\nu(\mathbf{q}, t)$, $\rho(\mathbf{q}, t)$ for $j^{\text{th}}$ repetition.
22:         end while
23:     end for
24: end for
```

each step. These actions are repeated hundred times starting from different random robot configurations to obtain a general pool of results without using the optimization tasks. Then, the same process is repeated with the dexterity, manipulability and RTR tasks, both when approaching the starting point and following the trajectory. The robot is initialized using the same random configurations used in the tests without optimization. Moreover, when the optimization tasks are employed, a monitoring is added while reaching the starting point to check if the control algorithm is still optimizing the robot configuration even though the end-effector frame has already reached the desired pose. So, the robot stats tracking the desired trajectory only when the kinetostatic index velocities $\dot{\eta}$, $\dot{\nu}$ and $\dot{\rho}$ are under a certain threshold $\delta = 10^{-6}$. This methodology is applied to each trajectory. A summary of the methodology is presented in Algorithm 3.1.

3.5 Trajectory tracking test in simulation results

This section describes all the results collected during the simulation tests. The results obtained with and without kinetostatic optimization tasks are compared to demonstrate the improvements made. Figure 3.4 collects the values of ϵ for the robot on the starting poses with and without optimization tasks. For each trajectory 1 to 4, two box plots are shown containing the results obtained by repeating the process without (blue) and with (red) optimization a hundred times. When the dexterity,

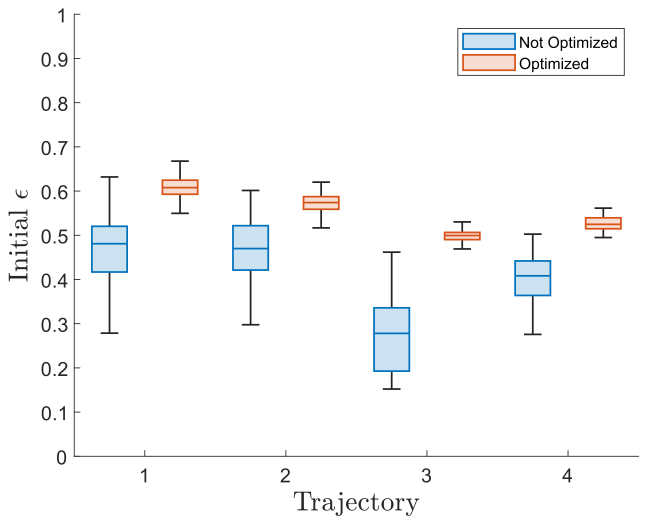


Figure 3.4: Values taken by ϵ at the starting pose of each trajectory (1 to 4, Fig. 3.1) for each one of the hundred repetitions, without (blue) and with (red) kinetostatic optimization

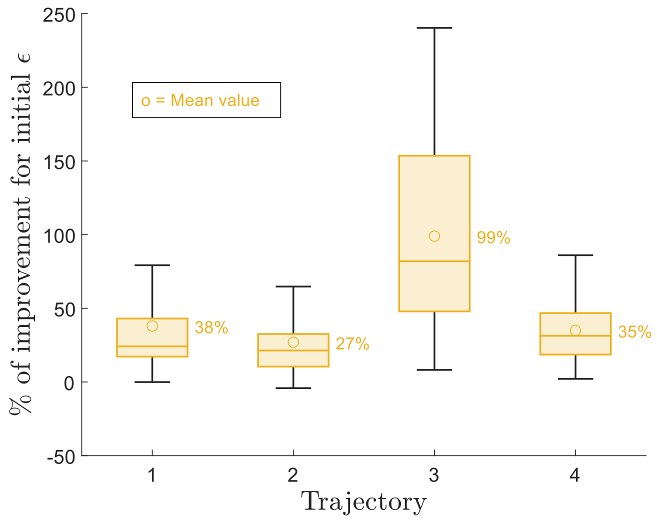


Figure 3.5: Percentages of optimization for ϵ in the starting pose of each trajectory comparing the optimized and non-optimized case for each of the hundred repetitions. The circle $\circ$ highlights the mean percentage of the hundred repetitions.

manipulability and RTR tasks are used, the variance of ϵ is smaller and the minimum and maximum values are high. This means that the optimization tasks always help reaching configurations with high values of ϵ. On the contrary, when the dexterity, manipulability and RTR tasks are removed, the variance of ϵ on the starting poses is bigger. Since the optimization tasks are disabled, the TPIK algorithm runs the robot straightly to the desired pose without performing any optimization on the robot configuration and ϵ can reach higher or lower values. So, the optimization task use provides an improvement to the robot performance without affecting the total simulation time. In fact, the average time for running the TPIK algorithm at each step is 698 μs without the optimization tasks and 744 μs with them. The computational time difference is negligible. Figure 3.5 presents the improvement of ϵ at the starting pose of each trajectory, with and without the optimization tasks for each of the hundred repetitions. The improvement of ϵ are almost always high. For the third trajectory, the range is larger than the other trajectories, over 200% in some cases. This happens because some non-optimized tests on the third trajectory reached very low kinetostatic values compared to the optimized tests, as shown in Fig. 3.4 However, few cases show a negative percentage. This is due to the control algorithm converging to a local maxima when optimizing the metrics. In fact, the optimized control algorithm reaches a local maxima in these few cases while the non-optimized one moves the robot in a configuration that escapes the local maxima, reaching higher kinetostatic performance unintentionally. This behavior happens a few times, which justifies the need to run the algorithm several times to obtain the best robot performance.

Figure 3.6 collects the mean values $\bar{\epsilon}$ of ϵ reached on each trajectory. Again, for each trajectory 1 to 4, there are two box plots containing the results obtained by repeating a hundred times the process without (blue) and with (red) optimization. The optimization tasks lead to a smaller variance for $\bar{\epsilon}$ compared to the non-optimized results. However, the minimum values of $\bar{\epsilon}$ in the non-optimized cases are higher than the minimum ϵ obtained on the non-optimized starting poses. This means that

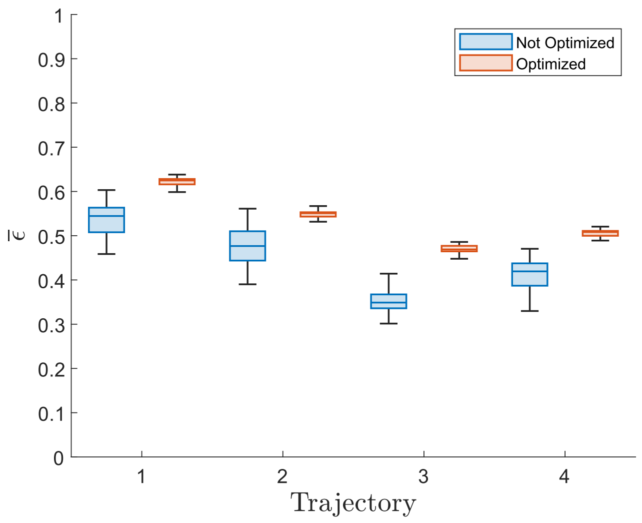


Figure 3.6: Mean values $\bar{\epsilon}$ of ϵ reached along each trajectory (1 to 4, Fig. 3.1) for each one of the hundred repetitions, without (blue) and with (red) kinetostatic optimization

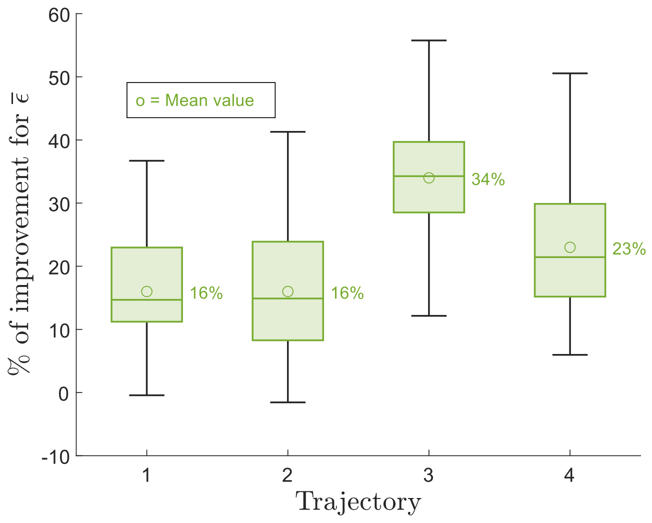
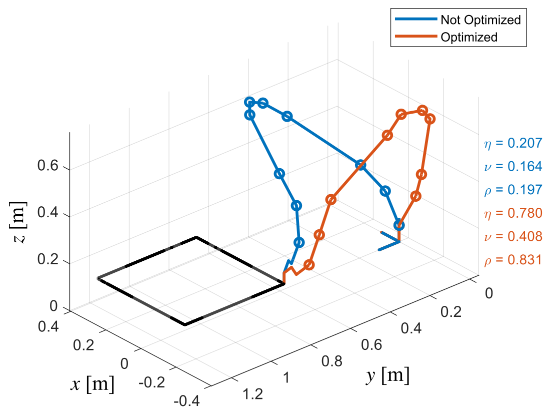


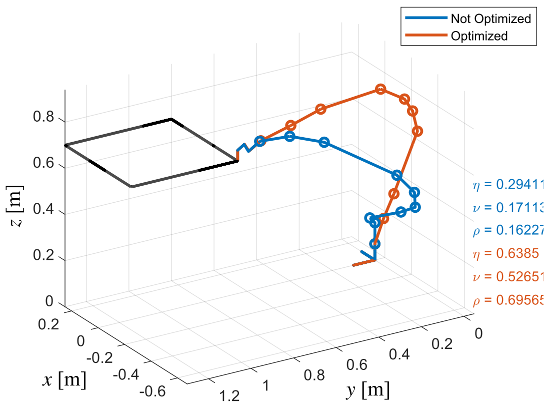
Figure 3.7: Percentages of optimization for mean value $\bar{\epsilon}$ of ϵ on each trajectory comparing the optimized and non-optimized case for each of the hundred repetitions. The circle $\circ$ highlights the mean percentage of the hundred repetitions.

the NB-R1 can maintain good kinetostatic performance on the trajectories even if the optimization tasks are not used and the starting ϵ is low. Moreover, the first trajectory shows the best performance, followed by the second, the third and the fourth. So, this design has better kinetostatic performance when the trajectory is horizontal and closer to the base. Figure 3.7 presents the improvement of ϵ on each trajectory, with and without the optimization tasks for each of the hundred repetitions. Again, it can be noticed that the percentage is negative in few cases due local maxima issue.

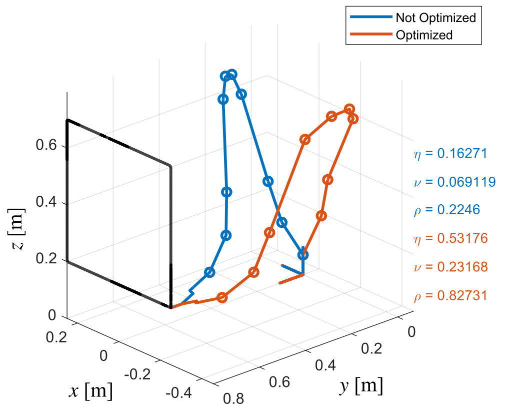
Figure 3.8 shows the robot on the starting pose of each trajectory for the minimum value of ϵ in case of no optimization and the maximum ϵ when optimization tasks were used. The values η, ν and ρ for the optimized and not optimized configurations are shown next to the robots in each figure. In the configurations assumed by the optimized robots, the x pose difference of one NB-module center and the next is



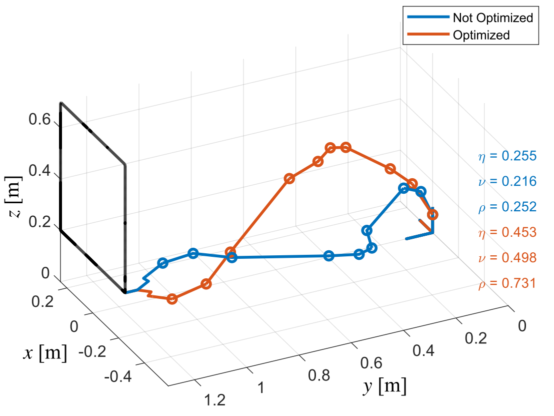
(a) First trajectory



(b) Second trajectory

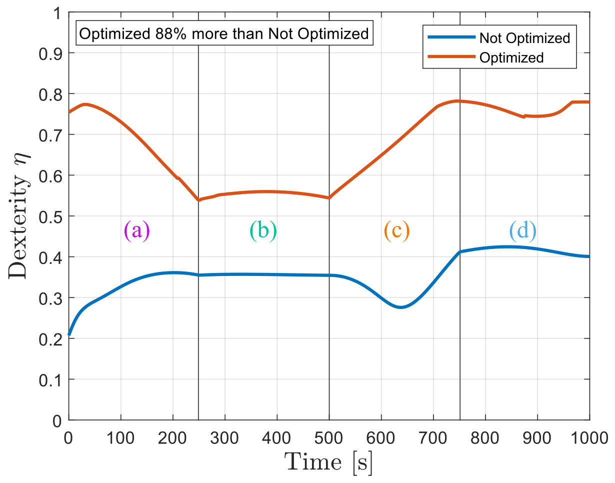


(c) Third trajectory

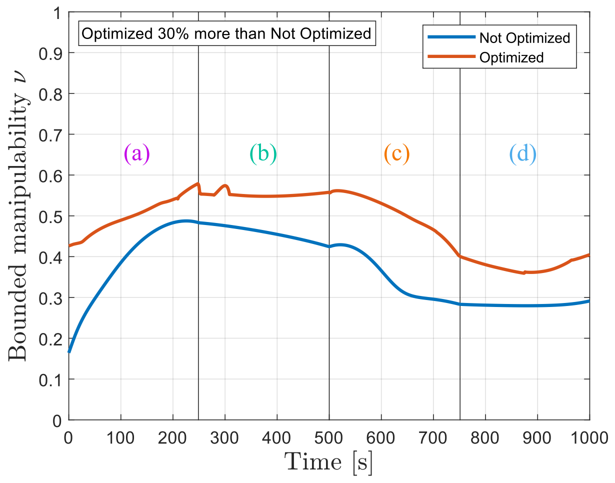


(d) Fourth trajectory

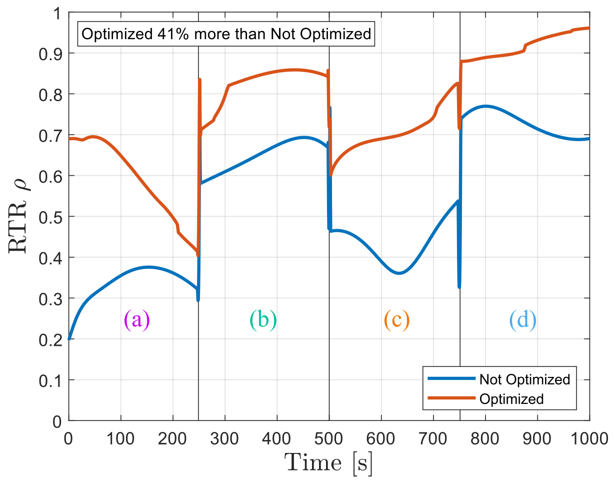
Figure 3.8: Robot in starting configuration on each trajectory for least value of ϵ in case of no optimization (blue) and highest value of ϵ with optimization (red). Values of η, ν and ρ in both cases are shown on the right of each sub-figure.



(a) Dexterity metric η



(b) Bounded manipulability metric ν



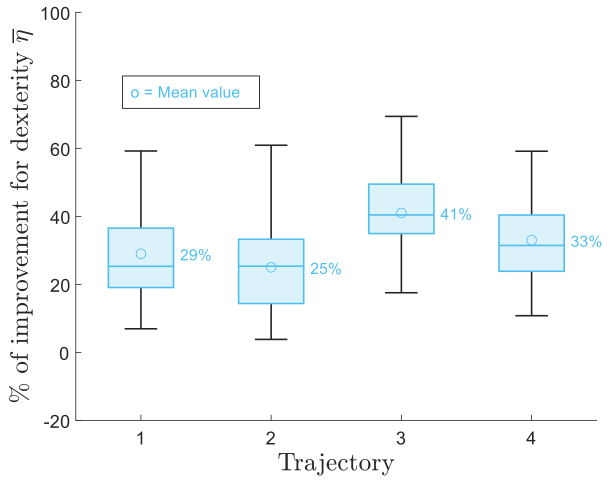
(c) RTR metric ρ

Figure 3.9: Graph of η, ν and ρ while following the first trajectory for least value of $\bar{\epsilon}$ in case of no optimization (blue) and highest value of $\bar{\epsilon}$ with optimization (red). Each sector is labeled (a), (b), (c) and (d) to match the corresponding part of the trajectory.

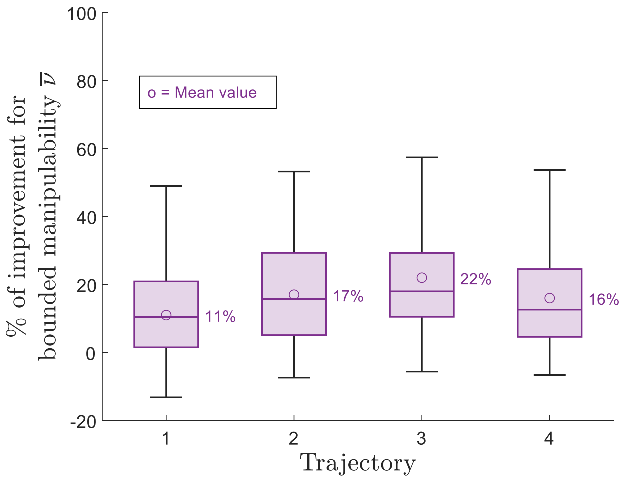
lower than in the non-optimized cases. This leads to a smaller angle between the joint velocity and torque vectors and increases the RTR value. Moreover, the vectors from the joint frames to the end-effector frame are different in length and orientation, giving distinct contributions to the kinematic Jacobian matrix and increasing the dexterity value.

Figure 3.9 shows the robot dexterity, bounded manipulability and RTR profiles along the first trajectory for the minimum value of $\overline{\epsilon}$ in case of no optimization and the maximum $\overline{\epsilon}$ with optimization tasks. The dexterity, bounded manipulability and RTR graphs are divided in the four trajectory sectors (a), (b), (c) and (d). The curves are higher when their tasks are used. The graphs also show the percentage of optimization for each curve. The activation of the dexterity task can improve the performance of almost a 90%. It can also be noticed how the RTR curve has discontinuities in correspondence to the trajectory corners since its value is directly affected by the orientation of the velocity and force vectors applied to the ending tool. Here, only the graphs for the first trajectory are shown since all the other trajectories showed similar behaviors. Comparing the dexterity, bounded manipulability and RTR results on each trajectory with and without optimization repeated a hundred times shows that the optimization tasks averagely increase the dexterity of 32%, the bounded manipulability of 17% and the RTR of 21%. Figure 3.10 shows the improvement percentage of $\overline{\eta}$, $\overline{\nu}$ and $\overline{\rho}$ with and without the optimization tasks on all trajectories and for a hundred repetitions. These percentages demonstrate how the use of optimization tasks generally improves the robot performance. In few cases, the negative percentage issue is met, which appears only for the bounded manipulability and the RTR.

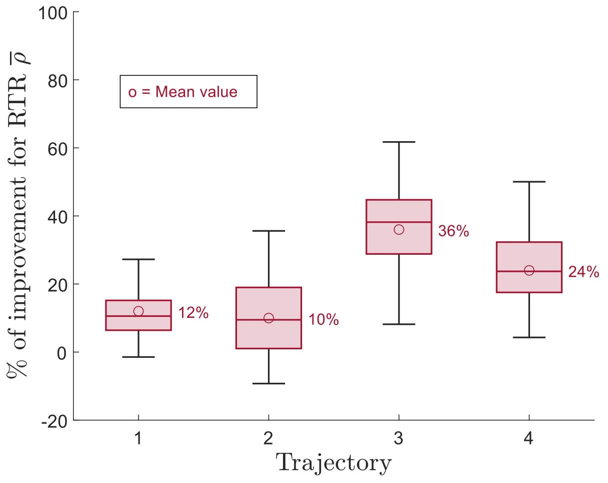
Another consideration that can be pointed out is the relation between the RTR value and the velocity and force vectors orientation. The RTR results shown in Fig. 3.10c are better in (b)-(d) than in (a)-(c) sectors. Taking into account the first and second horizontal trajectories in case of no optimization, the RTR values are in general 23% higher when the robot moves along axis x, (b)-(d) sectors, than along axis y, (a)-(c) sectors. When the RTR task is used, its performance difference between (b)-(d) and (a)-(c) sectors becomes 16% because the task helps maintaining higher values in all the sectors. Then, considering the third and fourth vertical trajectories in case of no optimization, the RTR values are 24% higher when the robot moves along axis x, (b)-(d) sectors, than along axis z, (a)-(c) sectors. When the RTR task is used, its performance difference between (b)-(d) and (a)-(c) sectors becomes 3%. So, the robot has higher RTR performance when the end-effector does



(a) Optimization percentages for mean value $\overline{\eta}$



(b) Optimization percentages for mean value $\overline{\nu}$

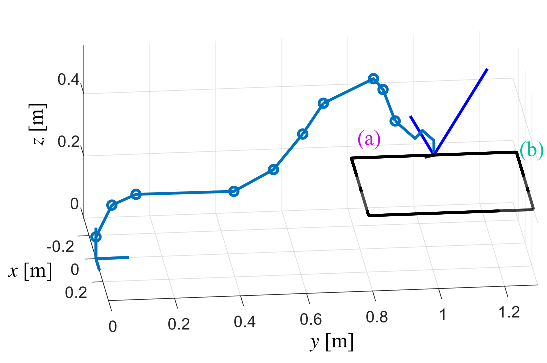


(c) Optimization percentages for mean value $\overline{\rho}$

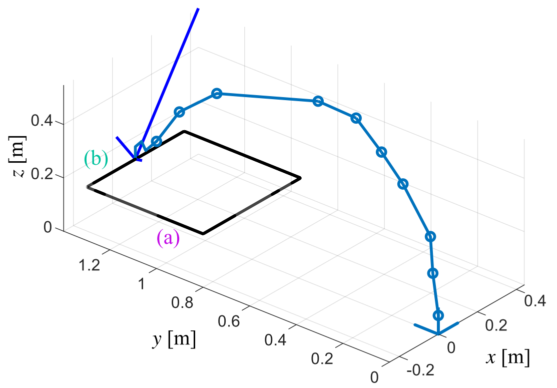
Figure 3.10: Percentages of optimization for mean value of all the optimization metrics on each trajectory comparing the optimized and non-optimized case for each of the hundred repetitions. The circle ○ highlights the mean percentage of the hundred repetitions.

a tangential horizontal movement than radial and vertical movements. A similar behavior can not be noticed in the dexterity or bounded manipulability because it is not affected by the magnitude or orientation of the velocity and force vectors applied to the end-effector.

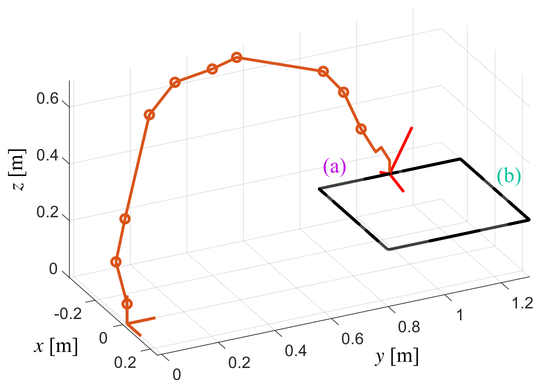
To further investigate the RTR behavior, it is useful to analyze the robot linear kinematic Jacobian matrix singular vectors applied to the end-effector. In this case, only the linear kinematic Jacobian matrix is considered since no end-effector twist or wrench is planned to be applied to the end-effector while tracking the trajectories. By definition, the RTR index is related to the angle between the joint velocity $\dot{\mathbf{q}}$ and torque $\boldsymbol{\tau}$ vectors and, as consequence, the angle between the end-effector twist $\mathbf{t}$ and the wrench $\mathbf{w}$, identifying the effectiveness in producing a desired motion. This concept is also highlighted by the robot linear kinematic Jacobian matrix singular vectors. A singular vector closer to zero means that the robot can no longer perform



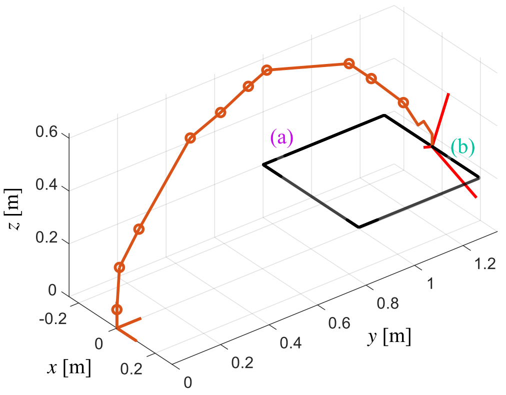
(a) Non-optimized simulation on sector (a)



(b) Non-optimized simulation on sector (b)

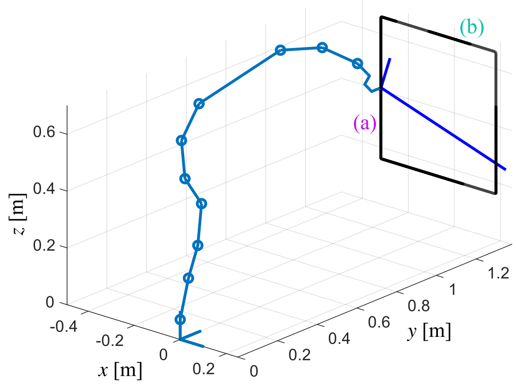


(c) Optimized simulation on sector (a)

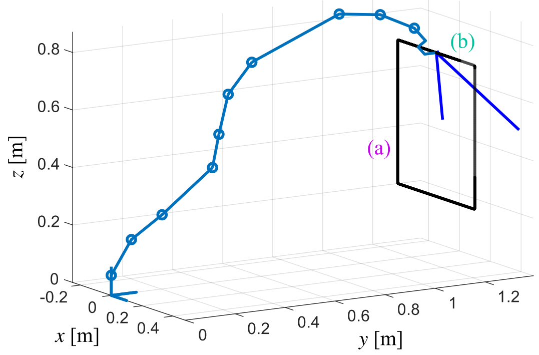


(d) Optimized simulation on sector (b)

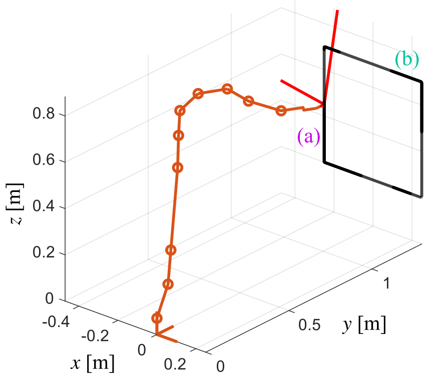
Figure 3.11: Robot configurations and linear kinematic Jacobian matrix singular vectors applied to the end-effector along first trajectory on sectors (a) and (b) for least value of $\bar{\epsilon}$ in case of no optimization (blue) and highest value of $\bar{\epsilon}$ with optimization (red)



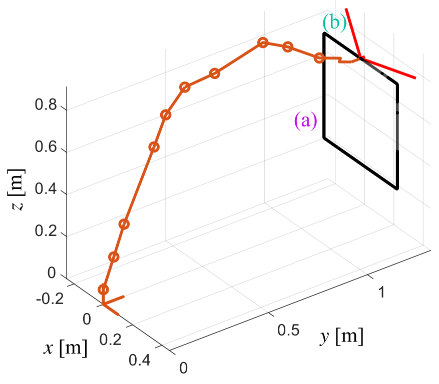
(a) Non-optimized simulation on sector (a)



(b) Non-optimized simulation on sector (b)



(c) Optimized simulation on sector (a)



(d) Optimized simulation on sector (b)

Figure 3.12: Robot configurations and linear kinematic Jacobian matrix singular vectors applied to the end-effector along fourth trajectory on sectors (a) and (b) for least value of $\bar{\epsilon}$ in case of no optimization (blue) and highest value of $\bar{\epsilon}$ with optimization (red)

a movement along the direction identified by the singular vector. On the contrary, a bigger singular vector implies that the robot can freely move along that direction. Figure 3.11 shows the robot configurations and linear kinematic Jacobian matrix singular vectors applied to the end-effector along the first trajectory. The blue robot is related to the least $\bar{\epsilon}$ obtained without optimization and the red ones to the highest $\bar{\epsilon}$ with optimization. Figures 3.11a and 3.11b present the robot and the linear singular vectors for the non-optimized simulation on the sectors (a) and (b), respectively. Figures 3.11c and 3.11d present the robot and the linear singular vectors for the optimized simulation on the sectors (a) and (b), respectively. It can be noticed that one singular vector is always close to zero, the one oriented along axis y. On the contrary, the other two singular vectors are longer. This justifies why the RTR performance are higher when moving along axis x, (b)-(d) sectors, than along axis y, (a)-(c) sectors. Moreover, this vector length difference is high in both the optimized and not optimized cases. This clarifies why the RTR performance are still a 16% higher along axis x than along axis y even if the RTR optimization task is employed while following the horizontal trajectories. Then, Fig. 3.12 shows the robot configurations and linear kinematic Jacobian matrix singular vectors applied to the end-effector along the fourth trajectory. The blue robot is related to the least $\overline{\epsilon}$ obtained without optimization and the red ones to the highest $\overline{\epsilon}$ with optimization. Figures 3.12a and 3.12b present the robot and the linear singular vectors for the non-optimized simulation on the sectors (a) and (b), respectively. Figures 3.12c and 3.12d present the robot and the linear singular vectors for the optimized simulation on the sectors (a) and (b), respectively. In this case, the singular vector oriented along axis y is still close to zero. However, the robot does not have to move along y. In the non-optimized case, the singular vectors when moving along axis x and z clearly show that the robot can apply a more effective movement along axis x. However, this difference decreases in the optimized case dropping the improvement percentage between axis x and z movement from 24% to 3%. It happens because the RTR task helps maintaining higher values in all the trajectory sectors and the performance drop is lower from one sector to the other.

3.6 Task priority based kinematic control conclusions

This chapter described a kinematic control algorithm called Task Priority Inverse Kinematic (TPIK) used to kinematically control the redundant robots, proposed in [SC16, SCWA18, SCWA19]. This kinematic control algorithm is tested on the NB-R1 to exploit its kinematic redundancy to solve simultaneous tasks. Three new tasks are introduced to improve the robot kinetostatic performance. One is based on the dexterity, the second one on the manipulability and the last one on the robot transmission ratio (RTR). The NB-R1 robot tracks a series of trajectories thanks to the TPIK algorithm with and without the kinetostatic optimization tasks. The major limitation of this algorithm is its attraction by local maxima. So, the process needs to be run several times to come up with the best robot configurations for the desired set of tasks and optimization metrics. When the optimization tasks are used, the results clearly show an improvement in the robot performance, both dexterity, manipulability and RTR, without affecting the time consumption. In fact, the average time consumed by the algorithm at each step with or without the optimization tasks is almost equal, the difference is less than 50 μs. To rate the improvement given by the optimization tasks, a linear combination of dexterity, bounded manipulability and RTR is used, named ϵ. When the robot reaches the trajectory starting pose using the optimization tasks, ϵ is averagely 50% higher than the non-optimized case. Along each trajectory, the mean value $\overline{\epsilon}$ is averagely 22% higher in the optimized case compared to the non-optimized one. Moreover, it can be noticed that the kinetostatic performance are affected by the trajectory placement and by the velocity and force vectors orientation. Finally, from the linear kinematic Jacobian matrix singular vectors study it can be deduced that the singular vector along axis y is always closer to zero, reducing the allowed movements along that direction. This reduces the movement effectiveness along the direction y. Part of the work presented in this chapter was published in [GCSL23a].

Chapter 4

Task Priority Based Kinematic Redundant Robot Design Optimization

This chapter proposes a design optimization process for kinematic redundant manipulators. The design optimization process is developed for kinematic redundant robots and employs the TPIK algorithm presented in Chapter 3 [SC16]. The chapter is organized as follow. Section 4.1 describes the existent methods for design optimization of redundant robot and the problems that can arise. Section 4.2 explains the new method for design optimization of robotic manipulators, which is the central topic. Section 4.3 tests the proposed optimization algorithm on the 21 degrees of freedom NB-R1 robot, presented in Chapter 1 and shown in Fig. 1.7 on page 11. Section 4.4 discusses and compares the obtained results. The conclusions about this new design optimization process based on the desired application are presented in Section 4.5.

4.1 Robot design optimization problem

When building a new kinematic redundant robot, an important step is the optimization of its design with respect to relevant performance indices. Angeles, in [Ang92], proposed an approach to design isotropic redundant manipulators by minimizing the condition number of the robot kinematic Jacobian matrix. However, an overall way to optimize redundant robots is to consider global indices. In [KSR$^+$14], the authors optimized a redundant serial manipulator using the global conditioning index. The design can also be optimized through both kinematic and dynamic global indices to obtain a robot with a high global dexterity over its whole workspace, ensuring high dynamic performance and energy efficiency [HKC$^+$17].

It is important to consider the main robot tasks during the robot design optimization process. An example limited to non-redundant manipulators is presented in [PK93]. In [KK93], the authors worked with re-configurable modular manipulator systems to address the problem of task-based robot design. The work presented in [WC18] explored a method to optimize the design parameters and the desired trajectory together, considering the desired task with interesting results. A critical issue in these task-oriented robot optimization processes is the complexity caused by the problem non-linearity. There are many ways to reduce the complexity, for example, breaking down the problem into multiple easier steps [KK93]. In [CGSBK18], the authors adopted a particle swarm optimization algorithm to determine the design parameter values of two collaborative robots. Other solutions employed a grid method [PCY03] or a complex direct search method [ADJM13]. However, none of these techniques benefits from the robot redundancy to perform an optimization as a function of multiple tasks.

In [MDA22], the authors developed a novel method for optimizing manipulator design using kinematic redundancy resolution. The authors use a combination of kinematic redundancy resolution and a multi-objective optimization algorithm. This type of design optimization algorithm involves replacing some design parameters with additional degrees of freedom to optimize in the robot architecture. This allows performing the same task with different robot configurations. Through the Jacobian null-space projection, the optimization algorithm modifies the chosen design parameters together with the robot configuration, solving the main task and some additional performance optimization sub-tasks. This algorithm gave promising results being able to identify some optimal parameter values. However, the algorithm was tested only on a two degrees of freedom non-redundant robot adding other two degrees of freedom for redundancy resolution optimization. Moreover, the proposed process does not present any mechanisms to avoid the kinematic and algorithmic singularities that can arise. Here, the main contribution is a new design optimization process for redundant manipulators with respect to the robot kinetostatic performance and all the tasks that will be executed at run-time. This goal is achieved by employing again the TPIK algorithm, described in Chapter 3, during both design and testing phases. As in [MDA22], virtual prismatic or revolute joints replace the design parameters to be optimized in the candidate generation phase. In this way, the kinematic control algorithm considers the design parameters as extra robot degrees of freedom and their value is optimized accordingly. The dexterity, manipulability and RTR tasks, described in Chapter 3, are included to improve the robot kinetostatic behavior during the design candidate generation and selection phases. The TPIK algorithm identifies some optimal robot design and configuration simultaneously solving the desired tasks.

4.2 Proposed design optimization method

The idea proposed in this chapter is to optimize the design of kinematic redundant manipulators based on the desired application, exploiting the TPIK algorithm already used for the online control. The TPIK framework is specifically designed to optimally control highly redundant robot manipulators, solving simultaneous tasks. In this case, the design optimization process is developed with respect to a tracking trajectory application while optimizing the robot kinetostatic performance. The problem formulation and all the phases that compose the optimization method are presented. In the considered case study, the robot main application requires following a trajectory inside a defined workspace while maximizing its kinetostatic performance. The kinetostatic performance is measured using the dexterity, manipulability and RTR indices.

4.2.1 Problem formulation

The application considered for the design optimization is tracking a set of trajectories that delimits a specific workspace area. While following these trajectories, the robot kinetostatic performance needs to be maximized. So, two principal inputs are defined and given to the optimization algorithm. The first one is a series of p trajectories with desired orientations and velocities. These trajectories delimit the workspace area where the robot will be applied. The second input is the list of tasks that the robot should achieve while tracking the trajectories. The main tasks are related to the end-effector pose and velocity to correctly track the desired trajectories. A safety task is added for joint limit compliance. Three tasks based on dexterity, manipulability and RTR evolution are used to improve the robot kinetostatic performance. Both the dexterity and manipulability are kinematic indices. As mentioned in Chapter 3, maximizing them simultaneously forces the manipulability hyper-ellipsoid to be as big as possible and close to a hyper-sphere. So, ideally, the robot will be able to move with the same high velocity amplification factor in all directions while reducing actuator velocity limits. The RTR task tries to align

Table 4.1: Details about the task names, control objective types, and hierarchy levels. Symbol (E) represents the equality control objective tasks and (I) the inequality ones. The last two columns list the task hierarchies for actions $\mathscr{A}_1$ (Reach Pose) and $\mathscr{A}_2$ (Follow Trajectory). The symbol "/" means that a task is not present in an action.

			Hierarchy level	
Task	Category	Type	$\mathscr{A}_1$	$\mathscr{A}_2$
Joint Limit	system safety	I	1st	1st
End-Effector Pose	action oriented	E	2nd	/
End-Effector Velocity	action oriented	E	/	2nd
Dexterity	optimization	I	3rd	3rd
Manipulability	optimization	I	3rd	3rd
RTR	optimization	I	3rd	3rd

the joint velocity and torque vectors making the robot movement along the desired direction more effective.

Table 4.1 reports all the task information used during the optimization. The dexterity, manipulability and RTR tasks have the same priority level since they have the same relevance in the design optimization process. The optimization algorithm uses the linear combination $\epsilon(\eta, \nu, \rho)$ of the three kinetostatic performance indices to rate and compare the kinetostatic performance of the obtained designs. The computation of $\epsilon(\eta, \nu, \rho)$ was described at Eq. (3.23) on page 49. To recall its formula, ϵ is computed as a function of the dexterity η, bounded manipulability ν and RTR ρ:

$$\epsilon(\eta, \nu, \rho) = \lambda_1 \eta + \lambda_2 \nu + \lambda_3 \rho, \tag{4.1}$$

where the bounded manipulability ν is the manipulability μ limited between $[0, 1]$,

$$\nu = 1 - \frac{1}{1 + \mu}. \tag{4.2}$$

4.2.2 Preliminary phase

The optimization algorithm is divided into two main phases. The first is the candidate generation phase, where several design candidates are generated. The second phase is the candidate selection one, where the robot designs obtained from the optimization process are evaluated and compared. Before these phases, some preliminary steps are necessary. The robot design parameters $\boldsymbol{\zeta}$ to be optimized need to be selected, for example, the link length or the angular offset amplitude between two consecutive joints. Then, controllable virtual joints are inserted in the robot architecture to substitute these design parameters. A prismatic joint substitutes a link to control its length, while a revolute joint replaces an angular offset to modify its angle. So, the design parameter vector $\boldsymbol{\zeta}$ is converted into virtual joint vector $\mathbf{q}_{n_v}$. During the candidate generation phase, the joint variable vector contains both the real and virtual joint positions $\mathbf{q} = [\mathbf{q}_{n_r}, \mathbf{q}_{n_v}] \in \mathbb{R}^n$, with $\mathbf{q}_{n_r} \in \mathbb{R}^{n_r}$, $\mathbf{q}_{n_v} \in \mathbb{R}^{n_v}$ and the robot degrees of freedom is $n = n_r + n_v$. The joint limit task constrains the virtual joint values $\mathbf{q}_{n_v}$ between the desired limits, resulting in limiting the design parameter values $\boldsymbol{\zeta}$.

4.2.3 Candidate generation phase

This phase employs the robot with real and virtual joints $\mathbf{q} = [\mathbf{q}_{n_r}, \mathbf{q}_{n_v}]$. The robot configuration vector $\mathbf{q}$ is randomly initialized. From here, the robot is moved to track all the p trajectories in a random order to ensure more general results. The robot reaches the starting pose of the trajectories and tracks it entirely under the kinematic control of the TPIK algorithm. When the robot finishes tracking a trajectory, it is moved to another one until it follows all the p trajectories. At equidistant time steps t_i on each trajectory, the optimization algorithm saves the virtual joint values $\mathbf{q}_{n_v}$ into the design parameter variable $\boldsymbol{\zeta}^{t_i}$ and its kinetostatic performance value in ϵ^{t_i}. Once the robot has tracked all the p trajectories, the optimization algorithm makes the weighted average $\overline{\boldsymbol{\zeta}}$ of all the collected $\boldsymbol{\zeta}^{t_i}$ using their associated ϵ^{t_i} as weighting factor:

$$\overline{\boldsymbol{\zeta}} = \frac{\epsilon^{t_0}\boldsymbol{\zeta}^{t_0} + \cdots + \epsilon^{t_\mathrm{f}}\boldsymbol{\zeta}^{t_\mathrm{f}}}{\epsilon^{t_0} + \cdots + \epsilon^{t_\mathrm{f}}}, \tag{4.3}$$

where $[t_0, \ldots, t_\mathrm{f}]$ are the equidistant time steps in which the design parameter vector $\boldsymbol{\zeta}$ was saved. This weighted average is useful to give more importance to those design parameters that provided better kinetostatic performance. The average re-

Algorithm 4.1 Candidate generation phase

```
Require: Actions $\mathscr{A}_1$ = Reach Pose and $\mathscr{A}_2$ = Follow Trajectory and trajectory vector $\mathbf{p}$.
 1: for $m_r := 1 \to$ number of repetitions do
 2:     Random initialization of $\mathbf{q} = [\mathbf{q}_{n_r}, \mathbf{q}_{n_v}]$.
 3:     Random shuffle of $\mathbf{p}$.
 4:     for $k := 1 \to p$ do
 5:         Reach starting pose of $\mathbf{p}(k)$ using $\mathscr{A}_1$.
 6:         while Trajectory $\mathbf{p}(k)$ not finished do
 7:             Move to next step on $\mathbf{p}(k)$ using $\mathscr{A}_2$.
 8:             if $t_i$ = equidistant time step then
 9:                 Save $\mathbf{q}_{n_v}$ as $\boldsymbol{\zeta}^{t_i}$.
10:                 Save $\epsilon^{t_i}$.
11:             end if
12:         end while
13:     end for
14:     Compute weighted average $\overline{\boldsymbol{\zeta}}$ for all $[t_0, \dots, t_f]$.
15: end for
```

sult $\overline{\boldsymbol{\zeta}}$ is stored. The process described above is repeated several times. At the end of this phase, the optimization algorithm has collected a list of candidate designs described by $\overline{\boldsymbol{\zeta}}$. Algorithm 4.1 sums up all the steps of the candidate generation phase.

During this phase, the optimization algorithm computes and collects the values of the kinetostatic indices η, ν and ρ for the real robot architecture without any virtual joints. In fact, the additional revolute/prismatic joints alter the kinetostatic performance of the real robotic design. This performance alteration risks damaging the optimization process and obtaining not optimized designs. So, the columns corresponding to the virtual joints $\mathbf{q}_{n_v}$ in $\mathbf{J}_w$ and its derivative $\partial\mathbf{J}_w/\partial q_i$, $\forall q_i \in \mathbf{q}$ are set to zero to correctly compute the kinetostatic metrics and their derivatives

4.2.4 Candidate selection phase

The candidate selection phase tests the designs obtained from the candidate generation phase to identify the best one. Here, the virtual joints are removed from the robot architecture vector and replaced by constant links and offsets whose value was stored inside all the obtained $\overline{\boldsymbol{\zeta}}$. In this way, several new robotic designs are

Algorithm 4.2 Candidate selection phase

```
Require: Actions $\mathscr{A}_1$ = Reach Pose and $\mathscr{A}_2$ = Follow Trajectory and trajectory
   vector $\mathbf{p}$.
 1: for $m_d := 1 \to$ number of optimized designs do
 2:     for $m_r := 1 \to$ number of repetitions do
 3:         Random initialization of $\mathbf{q} = [\mathbf{q}_{n_r}]$.
 4:         for $k := 1 \to p$ do
 5:             Reach starting pose of $\mathbf{p}(k)$ using $\mathscr{A}_1$.
 6:             while Trajectory $\mathbf{p}(k)$ not finished do
 7:                 Move to next step on $\mathbf{p}(k)$ using $\mathscr{A}_2$.
 8:                 Save $\epsilon^{t_i}$ at the time step $t_i$.
 9:             end while
10:         end for
11:         Compute $\hat{\epsilon} = 0.5\ \epsilon_{\text{min}} + 0.25\ (\bar{\epsilon} + \epsilon_{\text{max}})$.
12:     end for
13: end for
14: Select the best design as max($\hat{\epsilon}$).
```

generated. From now on $\mathbf{q} = [\mathbf{q}_{n_r}]$ and $n_v = 0$, and the total amount of degrees of freedom $n = n_r$. For each optimized robotic design, the configuration vector $\mathbf{q}$ is randomly initialized. Then, each robot is moved to the starting pose of one trajectory and tracks it entirely under the kinematic control of the TPIK algorithm. When one robot finishes tracking a trajectory, it is moved to the next one until it has followed all the p trajectories. At each trajectory time step t_i, the optimization algorithm stores the value of ϵ^{t_i} for that robot configuration. This process is repeated several times to obtain more general results. When the robot has tracked all the trajectories, the optimization algorithm computes $\hat{\epsilon}$ as

$$\hat{\epsilon} \triangleq 0.5\ \epsilon_{\text{min}} + 0.25\ (\bar{\epsilon} + \epsilon_{\text{max}}), \tag{4.4}$$

where ϵ_{min}, $\bar{\epsilon}$ and ϵ_{max} are respectively the minimum, mean and maximum of ϵ values along all the p trajectories. The $\hat{\epsilon}$ value is used to compare the designs and identify the best one. The process is repeated for all the designs obtained from the previous phase. The design that has the highest $\hat{\epsilon}$ is identified as the best one. Algorithm 4.2 sums up all the steps of the candidate selection phase.

4.3 Optimization test set up description

The design optimization process proposed here is tested on the NB-R1 robot. The following section describes the optimized robot design parameters and their features. Then, the trajectories employed for the optimization are described in terms of size and placement. Moreover, the velocities and forces required to follow them are listed. The machine and implementation details remained the same ones proposed in Chapter 3 and shown in Table 3.3 on page 48.

4.3.1 Robot under study

The employed robot for testing the design optimization algorithm is the NB-R1, shown in Fig. 1.7. In this case, the dimensions of the NB-module are constant and not included in the optimization. Their values are the same of the ones shown in Table 4.2. The optimized robot design parameters are the link lengths l_1 and l_2 and the amplitude of three angular offsets β_1, β_2 and β_3, shown in Fig. 4.1. The angular offsets are respectively inserted between the first link and first mechanism of the elbow, the second link and first mechanism of the wrist, and between the second-to-last and last mechanisms of the wrist. The angles β_1, β_2 and β_3 rotate about axis x, depicted in red in Fig. 4.1. An ending tool is mounted on the last revolute joint of the robot. The design parameters of the tool are inserted in the

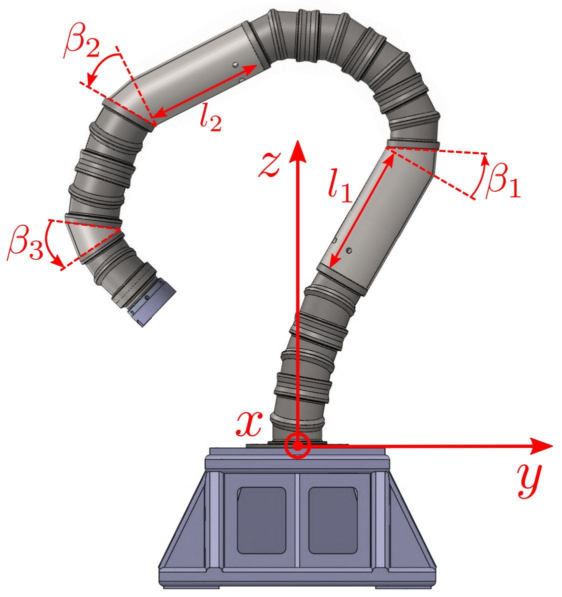


Figure 4.1: Robot design parameters under optimization: link lengths l_1 and l_2, angular offsets β_1, β_2 and β_3

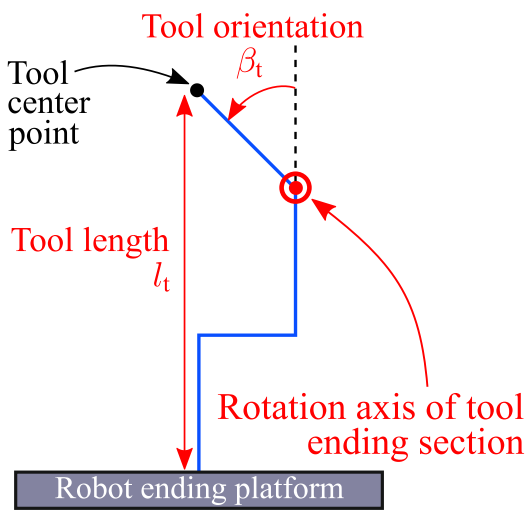


Figure 4.2: Ending tool design parameters under optimization: tool length l_t and orientation β_t

Table 4.2: NB-module dimensions plus real and virtual joint details

NB-module half height r	0.07 m
NB-module tube slope α	15°
Max/min revolute joint velocities	±1.0 rad/s
Max absolute revolute joint accelerations/decelerations	2.5 rad/s^2
Max/min prismatic joint velocities	±1.0 m/s
Max absolute prismatic joint accelerations/decelerations	2.5 m/s^2

optimized variables, namely its length l_t and orientation β_t, shown in Fig. 4.2. So, the number of virtual joints is $n_v = 7$ and the robot with virtual joints has 28 degrees of freedom, i.e. $n_r + n_v = 28$. Table 4.2 gives the main parameter values and limits of the robot.

The design parameters under optimization are limited to avoid too massive results with no sense. However, the limits are large enough trying to not over-constrain the optimization process. The link lengths l_1 and l_2 are constrained in the range $[0, 3]$ m. The angular offset amplitude β_1 is constrained in $[-3\pi/4, \pi/4]$ rad and β_2 and β_3 in $[-3\pi/4, 3\pi/4]$ rad. The tool length l_t is valid in the range $[0, 0.5]$ m, and the tool orientation offsets β_t in $[0, 3\pi/4]$ rad. These ranges are used by the joint limits task to constrain the virtual joints.

4.3.2 Employed trajectories

The robot application requires tracking horizontal and vertical trajectories in a cube whose sides are 0.7 m $\times$ 0.7 m long and centered in $(x, y, z) = (0.0, 0.75, 0.65)$ m. The orientation of the end-effector is expressed in terms of roll ϕ, pitch θ and yaw ψ. The angle values for the horizontal trajectories are $(\phi, \theta) = (\pi, 0)$, $\forall\ \psi \in [-\pi, \pi]$, and for the vertical trajectories $(\phi, \theta) = (\pi/2, 0)$, $\forall\ \psi \in [-\pi, \pi]$.

Figure 4.3 shows the $p = 4$ trajectories used during the candidate generation and selection phases of the optimization process. The two horizontal trajectories, green and magenta, have the corners placed in $x = (-0.35, 0.35)$ m and $y = (0.4, 1.1)$ m at the height of $z = 0.3$ m, green trajectory, and $z = 1.0$ m, magenta trajectory. The two vertical trajectories, blue and black, have the corners placed in $x = (-0.35, 0.35)$ m and $z = (0.3, 1.0)$ m at the depth of $y = 0.4$ m, blue trajectory, and $y = -1.1$ m, black trajectory. The small arrows along the trajectories express axis z orientation of the end-effector frame while following the trajectories. The trajectories are planned to cut a squared shape using a machining tool and the measures are shown in Fig. 4.4. Figure 4.4 also shows the orientation of the velocity vector $\vec{\mathbf{v}}$ and the tangential and radial force vectors $\vec{\mathbf{f}}_t$ and $\vec{\mathbf{f}}_r$ applied on the machining tool. The magnitudes of $\vec{\mathbf{v}}$, $\vec{\mathbf{f}}_t$ and $\vec{\mathbf{f}}_r$ are constant along the entire trajectory. The profiles of $\vec{\mathbf{v}}$, $\vec{\mathbf{f}}_t$ and $\vec{\mathbf{f}}_r$ are depicted in Fig. 4.5. The gravity force and the cutting one

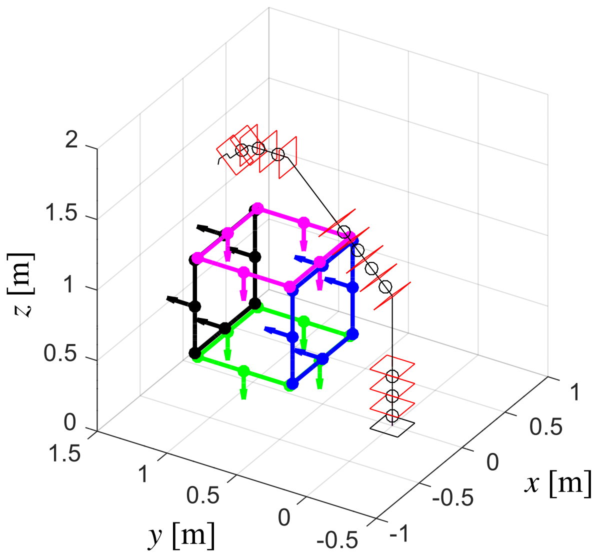


Figure 4.3: The four trajectories used in the optimization process and a version of Nimbl'Bot robot of size: $l_1 = 0.5$ m, $l_2 = 0.5$ m, $\beta_1 = \pi/4$ rad, $\beta_2 = \pi/4$ rad, $\beta_3 = \pi/4$ rad, $l_t = 0.1$ m and $\beta_t = \pi/4$ rad

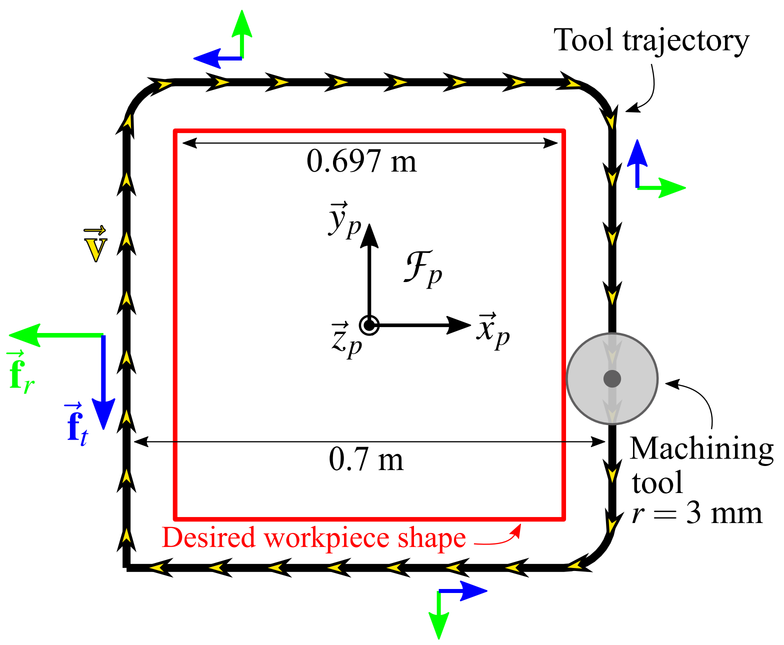
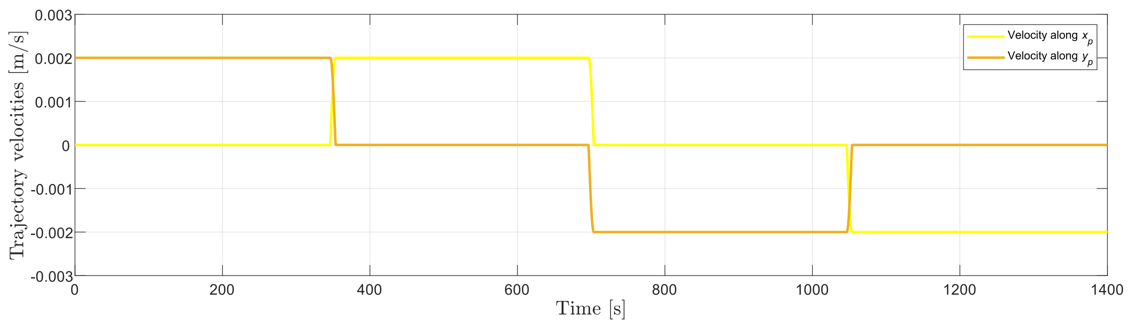
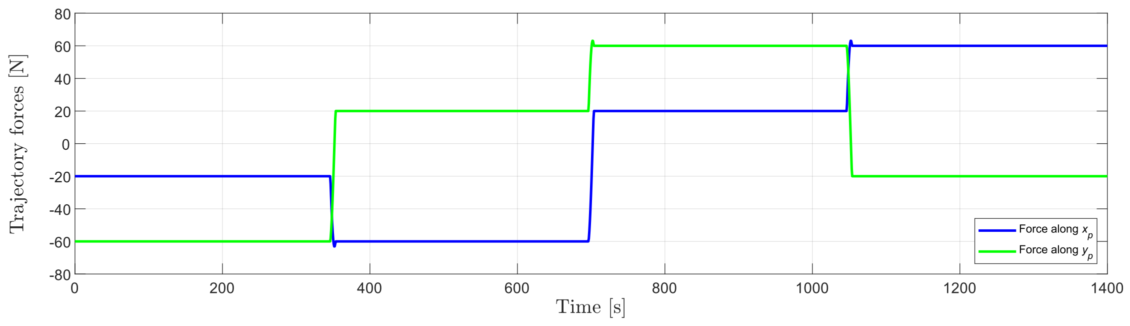


Figure 4.4: Measures of the desired workpiece, machining tool and trajectory tool with workpiece frame $\mathcal{F}_p$ in optimization test. Orientation of velocity vector $\vec{\mathbf{v}}$ (yellow) plus tangential and radial force vectors $\vec{\mathbf{f}}_t$ and $\vec{\mathbf{f}}_r$ (blue and green).



(a) Velocity profiles



(b) Cutting force profiles

Figure 4.5: Velocity and cutting force profiles applied to the machining tool in frame $\mathcal{F}_p$ during optimization tests

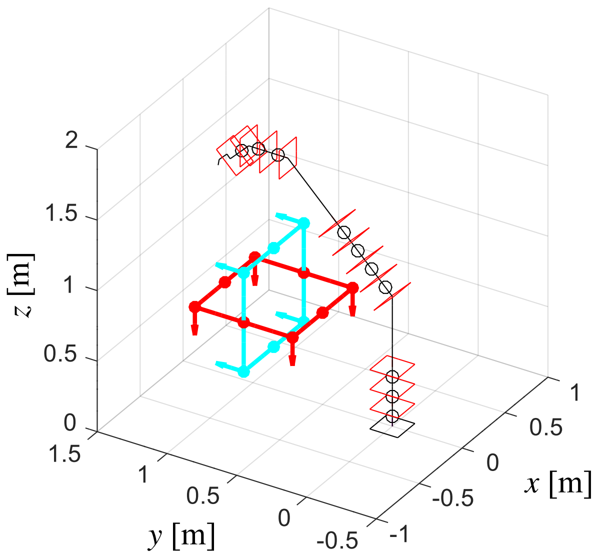


Figure 4.6: The two trajectories used to validate the designs obtained from the optimization process and a version of Nimbl'Bot robot of size: $l_1 = 0.5$ m, $l_2 = 0.5$ m, $\beta_1 = \pi/4$ rad, $\beta_2 = \pi/4$ rad, $\beta_3 = \pi/4$ rad, $l_t = 0.1$ m and $\beta_t = \pi/4$ rad

Table 4.3: Trajectory details in optimization test

Square side	0.7 m
Steps	560
Magnitude velocity vector $\|\|\vec{\mathbf{v}}\|\|_2$	0.002 m/s
Magnitude tangential force vector $\|\|\vec{\mathbf{f}}_t\|\|_2$	60 N
Magnitude radial force vector $\|\|\vec{\mathbf{f}}_r\|\|_2$	20 N
Time	1400 s

along $\vec{z}_p$ are neglected in this work. These four trajectories were chosen for the design optimization because they describe the cube where the robot is supposed to work in the real world. Then, the kinetostatic performance of the best and worst designs are compared on two new trajectories, one horizontal and one vertical, placed inside the cubic workspace area. These trajectories are used to confirm the optimization process ability to identify optimal kinetostatic designs for a desired application in a specific workspace area. Figure 4.6 shows the two trajectories used for testing the design obtained from the optimization process. The horizontal trajectory has the corners placed in $x = (-0.35, 0.35)$ m and $y = (0.4, 1.1)$ m at the height of $z = 0.65$ m, red trajectory. The vertical trajectory has the corners placed in $x = (-0.35, 0.35)$ m and $z = (0.3, 1.0)$ m at the depth of $y = 0.75$ m, cyan trajectory. Table 4.3 gives