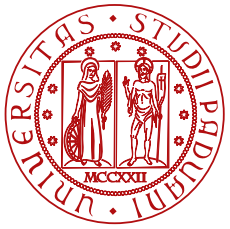


Università degli Studi di Padova

Dipartimento di Ingegneria dell'Informazione

Master Thesis in Control System Engineering

Modeling, identification, and control of a robotic arm for aerial manipulation

Master Candidate

Matteo De Facci

Student ID 2092382

Supervisor

Prof. Angelo Cenedese

University of Padova

Co-supervisor

Dott. Marco Cognetti

LAAS-CNRS

Co-supervisor

Dott. Gianluca Corsini

IRISA/CNRS-Inria

Academic Year
2023/2024

Abstract

Reaching high or inaccessible areas can be challenging, dangerous, or inefficient for human workers, even with appropriate equipment. Aerial vehicles have become a well-established solution, offering reliability, customizability, and scalability. Unlike ground vehicles, they have the advantage of a larger operational workspace. However, their effectiveness is limited by the absence of dexterous tools that would enable to physically manipulate the environment. Tasks such as pick-and-place operations, surface monitoring, and other contact-based activities require a "dexterity tool" that should be rigidly attached under the drone. For example, this tool can span from a static perch to a robotic manipulator. While the former has the main advantage to not significantly increase the robot weight, it does not introduce any additional degrees of freedom, that would allow the drone to perform a task in different ways, optimizing criteria such as manipulability. For this reason, the objective of this thesis relies in the conception and the control of a custom-designed robotic arm, specifically designed for aerial vehicles. We reason for a custom-design arm is due to the lack of commercially-available solutions in the market, since the existing manipulators are often bulky, energy-intensive. On the control side, while grounded manipulators often prioritize kinematic control over dynamic modeling, tasks involving environmental interaction or complex systems necessitate the control at the dynamic level. This is crucial for accurately predicting and controlling system behavior during contact tasks.

In summary, this thesis focuses on the modeling and control of a lightweight manipulator, specifically designed for drones, with particular attention to deriving an accurate dynamic model. Experimental validation is conducted using a custom-built robotic arm designed to be mounted on a hexa-rotor drone.

Sommario

Raggiungere aree alte o inaccessibili può essere difficile, pericoloso o inefficiente per gli operatori umani, anche con attrezzature adeguate. I veicoli aerei sono diventati una soluzione consolidata, che offre affidabilità, personalizzazione e scalabilità. Tuttavia, la loro efficacia è limitata dall'assenza di strumenti che permettano di manipolare fisicamente l'ambiente. Compiti come le operazioni di pick-and-place, il monitoraggio di superfici e altre attività basate sul contatto richiedono uno strumento rigidamente posto sotto il drone. Ad esempio, questo strumento può andare da un'asta statica a un manipolatore robotico. Se da un lato il primo ha il vantaggio principale di non aumentare significativamente il peso del robot, dall'altro non introduce alcun grado di libertà aggiuntivo, che permetterebbe al drone di eseguire un compito in modi diversi, ottimizzando criteri come la manipolabilità. Per questo motivo, l'obiettivo di questa tesi si basa sulla concezione e sul controllo di un braccio robotico progettato su misura, specificamente pensato per i veicoli aerei. La ragione di un braccio progettato su misura è dovuta alla mancanza di soluzioni disponibili sul mercato, poiché i manipolatori esistenti sono spesso ingombranti e ad alto consumo energetico. Per quanto riguarda il controllo, mentre i manipolatori a terra spesso privilegiano il controllo cinematico rispetto alla modellazione dinamica, i compiti che prevedono l'interazione con l'ambiente o con sistemi complessi richiedono il controllo a livello dinamico. Questo è fondamentale per prevedere e controllare accuratamente il comportamento del sistema durante i compiti di contatto.

In sintesi, questa tesi si concentra sulla modellazione e sul controllo di un manipolatore leggero, progettato specificamente per i droni, con particolare attenzione alla derivazione di un modello dinamico accurato. La validazione sperimentale è stata condotta utilizzando un braccio robotico progettato per essere montato su un drone esa-rotor.

Contents

List of Figures

List of Tables

List of Code Snippets

List of Acronyms

LAAS Laboratory for Analysis and Architecture of Systems

dof degree-of-freedom

DH Denavit-Hartenberg

NE Newton-Euler

EoM Equation of Motion

LLF Low-Level Feedback

LMI Linear Matrix Inequality

SDP Semi-Definite Programming

SLSQP Sequential Least Squares Quadratic Programming

SA Simulated Annealing

PI Proportional-Integral

RNEA Recursive Newton-Euler Algorithm

UMVE Unbiased Minimum Variance of Error

DFT Discrete Fourier Transform

LPF Low Pass Filter

CSV Comma Separated Values

BLDC BrushLess Direct Current

SPM Surface Permanent Magnet

FOC Field Oriented Control

ESCs Electronic Speed Controllers

CAN Controller Area Network

SPI Serial Peripheral Interface

URDF Unified Robotic Description Format

1 Introduction

Aerial vehicles have become a widely adopted solution due to their reliability, versatility, and scalability. Unlike ground vehicles, they provide the benefit of a significantly larger operational workspace. By integrating a manipulator, the range of tasks that the vehicle can perform is significantly expanded. As there are currently no commercially available manipulators designed for aerial manipulation, the decision was made to develop a custom-designed manipulator specifically for interaction with the environment.

Understanding the dynamics of the robot is crucial for model-based control, allowing for high-performance motion and precise force interactions. Additionally, the coupling effects between two torque-controlled systems, such as the drone and the arm, are significant and must be considered in the control design.

The robot dynamics is influenced by its geometry as well as rigid-body parameters, such as link inertia and friction. Dynamic parameters can be identified using computer-aided design (CAD) software, through experimental testing of individual robot components, or by employing regression methods based on dynamic models. Dynamic parameters have physical meaning and are therefore constrained by physically values, which are taken into consideration during the estimation process.

The manipulator was designed at LAAS and manufactured using PLA through a 3D printing process. A four-degree-of-freedom configuration was selected as a compromise between system complexity and dexterity. Given the maneuverability constraints of the drone, a fully-actuated arm was deemed excessive and would have unnecessarily increased the overall weight.

The kinematic and dynamic control algorithms were primarily developed in Python, as it allows for easy portability across different platforms and provides access to many well-optimized libraries. For the motor control boards, where the low-level feedback loops are implemented, C++ was chosen as the programming language to accommodate the microcontroller's requirements.

This document is organized as follows. Section 2 introduces the theoretical background of rigid body modeling and control, followed by a detailed description of the linear parameter formulation and the procedure for estimating a feasible set of parameters (Section 3). Section 4 presents the analysis of joint motion and the mechanical components, as well as the implemented motor control algorithms, including an examination of the joint parameters. Section 5 discusses the Gazebo robotics simulator and the corresponding developed model, along with the experiments conducted for data collection and the subsequent analysis of the results.

2

Theoretical Framework

2.1 Manipulator model

The term rigid body refers to a solid object which, even when subjected to external forces, deformation is negligible. In a rigid body, the distance between all internal points remains constant regardless of the application of external forces or moments. Rigid bodies can be either discrete, i.e. a finite union of masses, or with a continuous distribution of mass.

The position of a rigid body refers to all the particles of which it is composed. Since the body parts cannot move between them, it is possible to represent the object by its center of mass. Mechanical calculations are often simplified when expressed relative to the center of mass, a hypothetical point where an object's entire mass is assumed to be concentrated, allowing for easier visualization of its motion. This is the point to which a force may be applied to cause a linear acceleration without an angular acceleration.

A rigid body, unlike a material point, can be associated with a reference coordinate system. The origin of this system is arbitrary, though it is commonly placed at the center of mass. The rigid boy is completely described by its position and orientation with respect to a reference frame.

The linear position can be represented by a vector, expressed in the chosen coordinate system, usually pointing to the center of mass of the object. The orientation on the other hand has several way to be mathematically described: a set of three Euler angles, a rotation matrix or a quaternion.

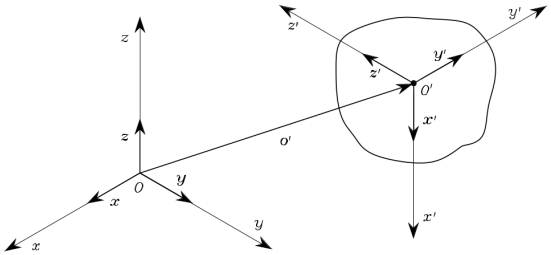


Figure 2.1: Position and orientation of two coordinate systems which could represent the world and body relation.

In general, the motion of a rigid body results in time-varying changes to both its position and orientation. From a kinematic perspective, these changes are described as translation and rotation, mathematically represented as a roto-translation between two coordinate systems: the body frame and the world frame.

2.2 Kinematic model

In the following theory, we assume that the arm consist in a open kinematic chain, namely there is just one sequence of joints and links that connect the base with the end effector. In the *open kinematic chain* every joint provides the structure with a single degree-of-freedom (dof), namely $n_{joints} = n_{dof}$.

2.2.1 Related Spaces

The position and orientation, potentially as functions of time, must be specified to define the task of the end-effector. This task depends on the specific application of the manipulator, and, in general, we refer to the *operational space* as the space in which the manipulator's task is defined.

In contrast, the joint variables represent the degrees of freedom available to the user for controlling the end-effector. The *joint space* (or configuration space) is then defined as the space to which the vector of joint variables $\mathbf{q} = \begin{bmatrix} q_1, \ldots, q_n \end{bmatrix}^T$ belongs.

The parametrization of the end-effector pose x_e defines the dimensionality of the operational space, $\mathbb{R}^m$. As previously mentioned, the pose embeds both the position and orientation of an object. By specifying the orientation using a triplet of Euler angles, which constitutes a *minimal representation*, it follows that $m \leq 6$. Specifically, three components represent the position $\mathbf{p_e}$, while the

remaining three describe the orientation $\mathbf{\Phi_e}$.

$$\mathbf{x}_e = \begin{bmatrix} \mathbf{p_e} \\ \mathbf{\Phi_e} \end{bmatrix} \qquad \mathbf{p}_e = \begin{bmatrix} p_{e_x} \\ p_{e_y} \\ p_{e_z} \end{bmatrix} \in \mathbb{R}^3$$

The orientation vector $\mathbf{\Phi}_e$ is no further specified since it depends on the specific choice triplet of angle. Also in many cases the computation of the three components of $\mathbf{\Phi}_e$ cannot be performed in close form but goes through the computation of the end-effector rotation matrix.

There are instances in which only a subset of the operational space is of interest, and in such cases, we refer to this as the *task space*. For instance, if the objective of the end-effector is positioning in space, only the Cartesian variables $\mathbf{p}_e$ are considered.

2.2.2 Direct kinematic

The direct or forward kinematics equations allow the *position* and *orientation* of the end-effector frame to be expressed as function of the *joint variables* with respect to the base frame.

In order to simplify the computation of the end-effector pose a good approach is to divide the problem into a combination of simpler sub-problems. In this case, the process can be decomposed into a sequence of sub steps, where each step relates the elementary motion of each link to the preceding one. Specifically, a sequence of reference frames can be placed approximately at each joint and the relative motion between them can be characterized using homogeneous transformation matrices.

$$A_i^{i-1}(q_i) = \left[\begin{array}{ccc|c} & \boldsymbol{R}_i^{i-1}(q_i) & & \mathbf{o}_i^{i-1} \\ \hline 0 & 0 & 0 & 1 \end{array}\right] \tag{2.1}$$

$$\boldsymbol{T}_n^0(\mathbf{q}) = \boldsymbol{A}_1^0(q_1)\dots\boldsymbol{A}_n^{n-1}(q_n)$$

In equation 2.1, $A_i^{i-1}(q_i)$ is the homogeneous transformation between two consecutive frames, it depends only on the value of the joint between the two coordinate systems. $\boldsymbol{R}_i^{i-1}(q_i)$ is a rotation matrix, while $\mathbf{o}_i^{i-1}$ is the translation vector between the frame origins. The composition of the elementary motion, gives $\boldsymbol{T}_n^0(\mathbf{q})$: the homogeneous transformation matrix from the last frame to the base one. At this stage, given a fixed joint configuration, the pose of the end-effector can be determined.

The direct kinematic function 2.2, represent a mapping from the joint space

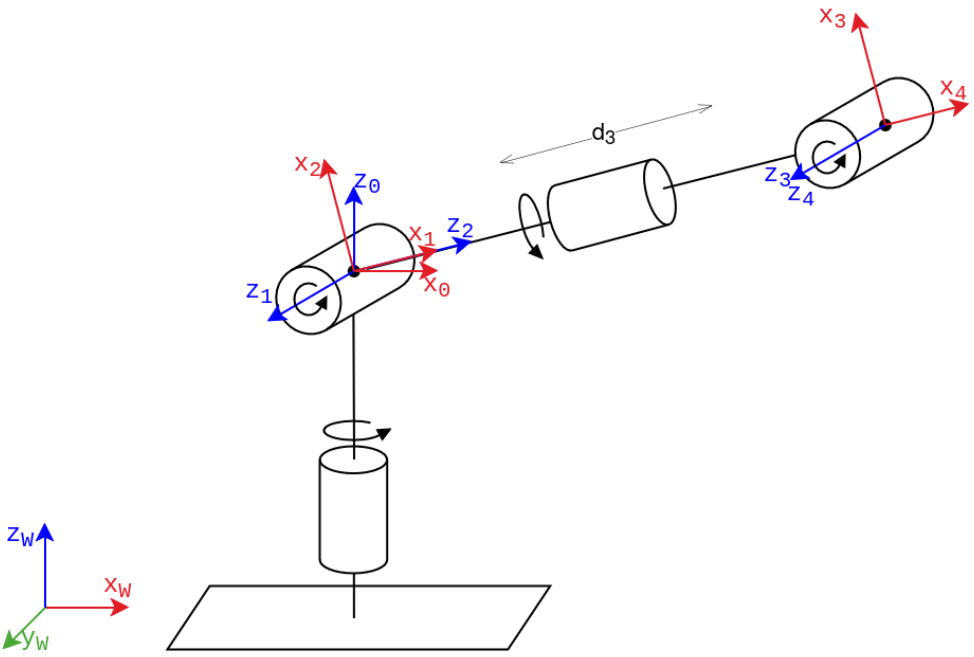


Figure 2.2: Schematic of the manipulator with joint reference frames, following the Denavit-Hartenberg (DH) convention. The reference frames consist of orthonormal bases, following the right-hand rule.

to the operational space.

$$\begin{aligned} \mathcal{K} : \mathbb{R}^n &\longrightarrow \mathbb{R}^m \\ \mathbf{q} &\longrightarrow \begin{bmatrix} \mathbf{p_e} \\ \mathbf{\Phi_e} \end{bmatrix} = \mathbf{x}_e \end{aligned} \tag{2.2}$$

This function varies depending on the specific application and the task of interest for the manipulator.

In this paper we are interested only on the position of the end-effector (so only on the translation component of $\boldsymbol{T}_n^0(\mathbf{q})$) as orientation tasks are performed by the drone.

The position, orientation, and number of these coordinate systems are arbitrary; however, adopting a systematic approach facilitates the construction of a clearer model and prevents the inclusion of excessive frames. In the literature, it is common practice to position reference frames along the kinematic chain following the Denavit-Hartenberg (DH) convention. Due to space and content constraints, this systematic procedure will not be detailed in this document, but a comprehensive explanation is available at [4].

In the designed manipulator only revolute joints are included, again because the drone performs the moving. Hence the joint variables enters in the DH-table (Table 2.1) as the angle parameter θ_i, which is the angle between two consecutive x-axis as shown in Figure 2.2.

As it is possible to observe from table 2.1, the only *kinematic parameter* related to this structure is d_3, namely the length of the arm. In the specific case

joint	a_i	d_i	α_i	θ_i
1	0	0	$\pi/2$	q_1
2	0	0	$\pi/2$	$q_2 + \pi/2$
3	0	d_3	$\pi/2$	q_3
4	0	0	0	q_4

Table 2.1: DH parameter of the experimental manipulator.

$d_3 = 0.65m$.

2.3 Dynamic model

The dynamic model of a rigid body provides the relation between generalized forces acting on the robot and the motion of the same. The development of a manipulator's dynamic model is essential for motion simulation, structural analysis, and the design of control algorithms.

Simulating the motion of a manipulator enables the evaluation of control strategies and motion planning techniques without the need for a physically available system. Furthermore, the analysis of the dynamic model is instrumental also in the mechanical design of prototype arms. Calculating the forces and torques necessary to execute typical movements provides valuable insights for the design of joints, transmissions, and actuators.

The primary assumption in this document is that the agents can be modeled as a composition of one or more rigid bodies: physical entities that do not undergo deformation or, more realistically, experience deformations so minimal that their effects can be neglected.

The equation of motion can be derived in two formalisms: Newton-Euler (NE) formulation and Lagrange formulation. In the following, only the latter approach will be presented, as it provides a systematic framework that facilitates direct analysis for control tasks. However, due to its recursive formulation, the NE principle is better suited for real time applications, and is the method implemented in the code of this project. Notably, both formalisms yield the same dynamic model.

2.4 Lagrange formulation

The equation of motion can be derived in a systematic way, *independently* of the reference coordinate frame [4, pp.247-259].

At first a set on *generalized coordinates* must be chosen to describe the *n-dimensional* system. In the case of a manipulator these variables are the joint positions, namely q_i, $i = 1, \ldots, n$ where $\mathbf{q} = \begin{bmatrix} q_1, \ldots, q_n \end{bmatrix}^T \in \mathbb{R}^n$. The *Lagrangian*

formalism exploits the kinetic energy $\mathcal{K}$ and the potential energy $\mathcal{U}$ as function of the generalized coordinates (and derivatives) in the **Lagrangian** $\mathcal{L}$.

$$\mathcal{L}(\mathbf{q}, \dot{\mathbf{q}}) = \mathcal{K}(\mathbf{q}, \dot{\mathbf{q}}) - \mathcal{U}(\mathbf{q}) \tag{2.3}$$

The Lagrange equations can then be computed and in their compact form, they are expressed as follows:

$$\frac{d}{dt}\left(\frac{\partial \mathcal{L}}{\partial \dot{\mathbf{q}}}\right)^T - \left(\frac{\partial \mathcal{L}}{\partial \mathbf{q}}\right)^T = \xi \tag{2.4}$$

where the vector ξ describes *non conservative* torques and forces doing work at each manipulator joint. In general, ξ may represent any force acting on the manipulator, i.e. actuation torques, the friction effects, aerodynamic effects and the interaction with environment forces.

For a rigid body, the kinetic and potential energy can be explicitly calculated for each link and corresponding joint motor. For simplicity, motors are assumed to have zero mass, making their contribution as *independent* rigid bodies negligible. In other words, they are assumed to exhibit no inertia or gyroscopic effects and, consequently, do not appear as independent terms in the Lagrangian. However, their mass is incorporated into the total mass of the respective link.

For standard mechanical systems (composition of rigid bodies), the kinetic energy can be proven to have be quadratic form of the type $\mathcal{K} = \frac{1}{2}\dot{\mathbf{q}}^T \boldsymbol{B}(\mathbf{q})\dot{\mathbf{q}}$, where $\boldsymbol{B}(\mathbf{q})$ is the *inertia matrix* and it can be shown to be a $(n \times n)$ symmetric and *positive-definite* matrix, independently on the generalized vector $\mathbf{q}$. Notably, the potential energy depends only on the joint position $\mathbf{q}$.

Taking the derivatives required by Lagrange equations in 2.4 and rearranging the centrifugal and Coriolis terms into matrix form, it yields

$$\boldsymbol{B}(\mathbf{q})\ddot{\mathbf{q}} + \boldsymbol{C}(\mathbf{q}, \dot{\mathbf{q}})\dot{\mathbf{q}} + \mathbf{g}(\mathbf{q}) = \xi \tag{2.5}$$

The matrix $\boldsymbol{C}(\mathbf{q}, \dot{\mathbf{q}})$ is a suitable $(n \times n)$ matrix whose entries c_{ij} satisfy a certain equation, resulting from the Lagrange equations. Occasionally, to simplify notation, the Coriolis and gravitational effects are embedded into a single vector: $\boldsymbol{C}(\dot{\mathbf{q}}, \mathbf{q})\dot{\mathbf{q}} + \mathbf{g}(\mathbf{q}) = \mathbf{n}(\dot{\mathbf{q}}, \mathbf{q})$.

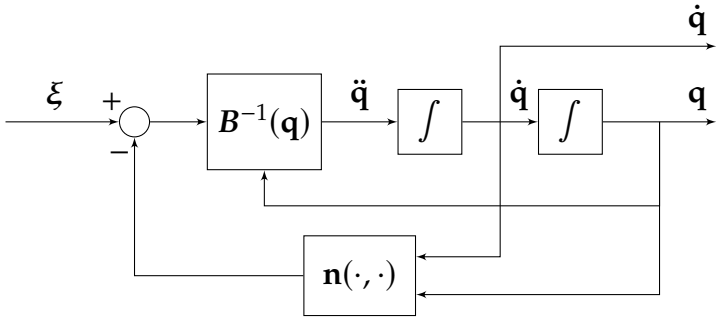


Figure 2.3: Representative diagram of the robot dynamics: the integrators provide a mathematical representation of the system's internal processes. The joint torques serve as inputs, while the joint positions and velocities are outputs, characterizing the system's behavior.

Regarding the *generalized force vector* ξ it takes into account for the actuation torques, and may also includes environmental interaction forces. If the manipulator's end-effector is in contact with the environment, a portion of the actuation torques is used to *balance* the torque induced at the joints by these contact forces. Denoting by $\mathbf{h}_e$ the vector of forces and moments exerted by the end-effector, the resulting joint torques are given by

$$\boldsymbol{\tau}_\mathbf{e} = \boldsymbol{J}^T(\mathbf{q})\mathbf{h}_e$$

where the *configuration dependent* matrix $\boldsymbol{J}(\mathbf{q})$ is the *geometric Jacobian* deriving from the differential kinematics.

2.4.1 Joint Friction

The vector ξ can also be used to incorporate friction effects. Friction is a complex *non-linear* phenomena, especially during motion reversal. However, the following non-linear formulation in the joint velocity, is often an acceptable simplification for many robotics applications

$$\tau_{f_i} = f_{v_i}\dot{q}_i + f_{s_i}\,\mathrm{sgn}\,\dot{q}_i \tag{2.6}$$

This formulation has the clear advantage to be linear with the parameters f_{v_i}, f_{s_i}: property that will be exploited later in the document.

In equation 2.5, it is possible to include the friction contributions, yielding complete model as shown in equation 2.7.

$$\boldsymbol{B}(\mathbf{q})\ddot{\mathbf{q}} + \boldsymbol{C}(\mathbf{q},\dot{\mathbf{q}})\dot{\mathbf{q}} + \mathbf{g}(\mathbf{q}) + \boldsymbol{F}_v\,\dot{\mathbf{q}} + \boldsymbol{F}_s\,\mathrm{sgn}(\dot{\mathbf{q}}) = \boldsymbol{\tau} - \boldsymbol{J}^T(\mathbf{q})\,\mathbf{h}_e \tag{2.7}$$

where $\boldsymbol{F}_v = diag\left\{f_{v_1},\ldots,f_{v_n}\right\}$ is the viscous friction matrix and $\boldsymbol{F}_s = diag\left\{f_{s_1},\ldots,f_{s_n}\right\}$ is the Coulomb or static friction matrix.

Equation 2.7 is named *Equation of Motion (EoM)* as it provides the relation between the motion of the body with the generalized force contributions.

Figure 2.3 schematically illustrates the behavior of the rigid body when a given torque vector $\boldsymbol{\xi}$ is applied.

2.4.2 Direct and inverse dynamics

Given the dynamic model equation in the form of 2.7, two frameworks related to system dynamics can be considered:

- the **direct dynamics** problem consists in determining the system's motion, expressed through the generalized coordinates $\mathbf{q}, \dot{\mathbf{q}}, \ddot{\mathbf{q}}$, given the input torque vector $\boldsymbol{\xi}$ and the system's initial conditions. This approach is particularly useful for simulation purposes.
- Conversely, the **inverse dynamics** problem takes the motion variables as input and calculates the required torque to achieve the desired trajectory, making it well-suited for control applications.

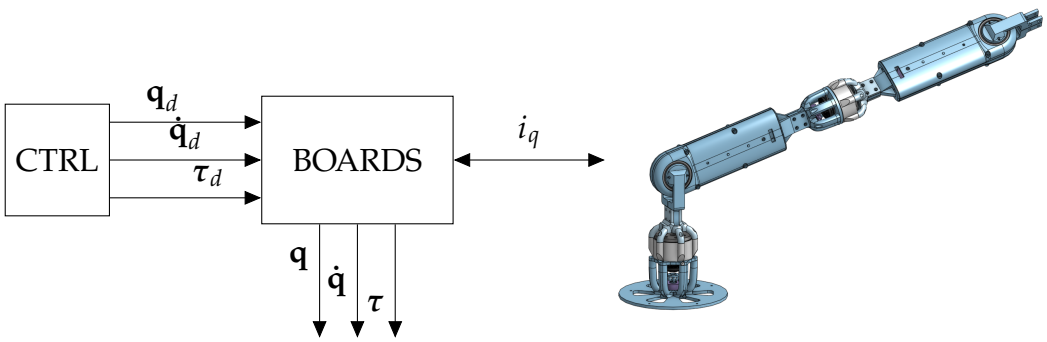


Figure 2.4: Control Scheme: signal flows to and from the manipulator. The joints are controlled via current, although the control boards also support position and velocity commands. Additionally, the boards provide measurements of position, velocity, and torque, which can be utilized for feedback control.

2.5 Basic controllers

A variety of joint space control techniques are presented, which can be classified into two categories. *Decentralized* control schemes involve controlling each manipulator joint independently of the others, whereas *centralized* control schemes account for the dynamic interaction effects between the joints.

Task specifications, such as end-effector motion and applied forces, are typically defined in the operational space, while control actions, such as the generalized forces of the joint actuators, are executed in the joint space. This distinction naturally gives rise to two general control schemes: joint space control and operational space control. In both schemes, the control structure incorporates closed-loop feedback to exploit its advantages, including robustness to modeling uncertainties and the mitigation of disturbance effects. For general manipulator applications, the reference trajectory is typically defined in the *operational* space, namely $\mathbf{x}_d$. However in the *joint space* control problem this motion has to be transformed in the corresponding motion $\mathbf{q}_d$ in the joint space which is what can be tracked by the controller.

Although the operational space variables $\mathbf{x}_e$ are not directly controlled, and the pose moves according to the manipulator structure. It is clear that any structural uncertainties, such as construction tolerances, lack of calibration, gear backlash, or elasticity, as well as any imprecision in the knowledge of the end-effector's pose relative to the object being manipulated, result in reduced accuracy of the operational space variables.

On the other hand the *operational space* control problem relies on the the feedback loop directly in the operational space coordinates. Its conceptual advantage lies in the ability to act directly on the operational space variables. However, this advantage is largely theoretical, as operational space variables are often not measured directly but are instead obtained by evaluating direct

kinematics functions based on the measured joint space variables.

First will be discussed an implementation of the *decentralized control*, since it will be useful in the parameter estimation section, then some dynamic controllers will be presented.

2.5.1 Decentralized kinematic control

Kinematic control involves designing the controller while neglecting the robot's dynamics. To compensate for this omission, a Low-Level Feedback (LLF) loop is necessary which ensures that the commanded velocities are accurately tracked.

When a manipulator is actuated by electric motors with high-ratio reduction gears, the gears tend to linearize the system dynamics and decouple the joints, thereby mitigating non-linearity effects. However, this advantage comes at the cost of joint friction, elasticity, and backlash, which can impose greater limitations on system performance than configuration-dependent inertia, Coriolis forces, centrifugal forces, and other factors.

The *decentralized control* strategy ignores the overall dynamics of the system and establishes n independent control schemes, as illustrated in Figure 2.5, where each joint is considered as single-input/single-output system. The non linearities are treated as disturbances in the control loop. Then the control must be properly selected and tuned to reject them.

To guide selection of the controller structure, start noticing that an effective rejection of the disturbance d on the output $\dot{q}_i$ is ensured by:

- a large value of the amplier gain before the point of intervention of the disturbance,
- the presence of integral action in the controller helps to eliminate the effect of the gravitational component on the output at steady state, where these effects remain constant.

These requisites clearly suggest the use of a Proportional-Integral (PI) control action in the forward path whose transfer function is

$$C_V(s) = K_P + K_I \frac{1}{s}$$

Even though the system dynamics, as described in Section 2.4, is clearly *non-linear* and *coupled*, by implementing a low-level feedback we can approximate the scheme of Figure 2.5 as an integrator. The inner loop, or the main controller is implemented in the driver boards. In this way the control bandwidth is higher enough to consider it as analog loop.

Once the controllers are tuned the low level loop, of Figure 2.5 is then seen as a simple integrator for each joint. Then the kinematic control introduced before can be implemented, as in section 2.5.2 and 2.5.3.

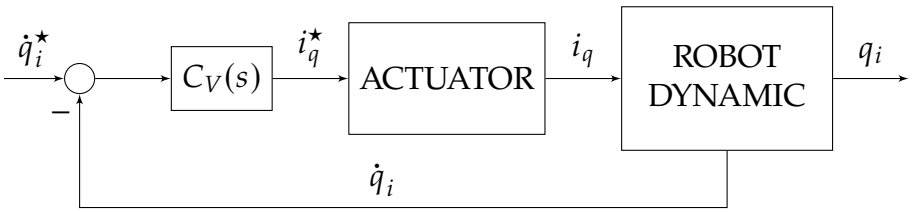


Figure 2.5: Decentralized control scheme. This structure is implemented for every joint i. Due to the high bandwidth of the motor current, the control loop can be approximately considered as analog.

In the context of this project, the trajectory is already defined in the joint space, thus eliminating the need for a *reference generator*, as would be required in the general case.

A brief description of two possible kinematic control strategies, exploiting the above-mentioned approximation are now presented.

2.5.2 Kinematic control of joint motion

In this case as input the Cartesian trajectory of the end effector $\mathbf{p}_d$ and $\dot{\mathbf{p}}_d$ is considered. The trajectory needs to be translated to the joint space to serve as the system's input.

To achieve this, the inverse differential kinematic theory must be exploited, which is what the block $J^{-1}(\mathbf{q}_d)$ of scheme 2.6 is designed to do. The reason of the nomenclature comes from the derivation of the direct kinematic. Indeed the time derivative of relation 2.2 is

$$\dot{\mathbf{x}}_e = \begin{bmatrix} \dot{\mathbf{p}}_e \\ \dot{\boldsymbol{\Phi}}_e \end{bmatrix} = \begin{bmatrix} J_p(\mathbf{q}) \\ J_\Phi(\mathbf{q}) \end{bmatrix} \dot{\mathbf{q}} = J_K(\mathbf{q})\,\dot{\mathbf{q}} \tag{2.8}$$

where $J_K(\mathbf{q})$ is the analytical Jacobian of $\mathcal{K}(\mathbf{q})$. Given that we are considering only the Cartesian variables $\dot{\mathbf{p}}_e$, the relation to invert is $\dot{\mathbf{p}}_e = J_p(\mathbf{q})\,\dot{\mathbf{q}}$. This operation is generally *non-invertible* and requires careful considerations. For simplicity in the schematic 2.6, the inversion is represented using standard matrix notation.

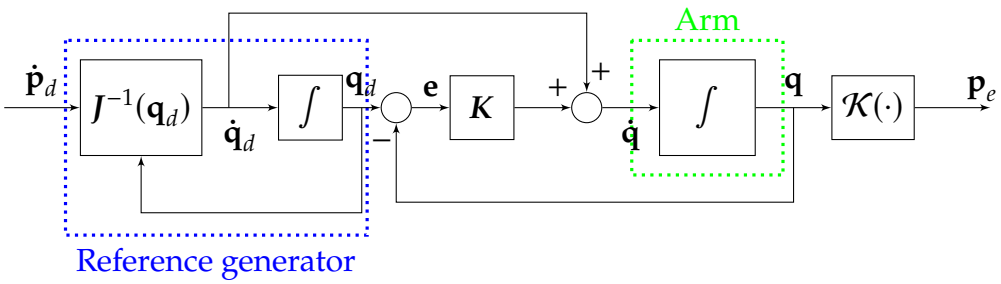


Figure 2.6: Kinematic control scheme of joint motion. The arm dynamic is approximated as an integrator, assuming the presence of the LLF. The end effector position is then computed through direct kinematic.

The *reference generator* exploit the kinematic relation between joint velocity and velocity of Cartesian variables. It is performed over the *desired* trajectory, hence it can be computed *offline*.

The error converges to zero if the gain matrix $\boldsymbol{K} > 0$. Indeed, from the scheme 2.6, it holds:

$$\dot{\mathbf{e}} = \dot{\mathbf{q}}_d - \dot{\mathbf{q}} = \dot{\mathbf{q}}_d - (\dot{\mathbf{q}}_d + \boldsymbol{K}(\mathbf{q}_d - \mathbf{q})) = -\boldsymbol{K}\mathbf{e} \qquad \mathbf{e} = \mathbf{q}_d - \mathbf{q}$$

Thus, the evolution of the error $\mathbf{e}$ is described by a first-order differential equation, with an exponential decay rate determined by the gain matrix $\boldsymbol{K}$.

2.5.3 Kinematic control of Cartesian motion

The second approach involves compensating for the error that arises in the Cartesian space. In this case, the reference generator functions is an integrator that provides the position trajectory in accordance with $\dot{\mathbf{p}}_d$. Typically, precise knowledge of the end-effector's position is unavailable and must therefore be determined through direct kinematics. The scheme of Figure 2.7 shows the general view of this approach.

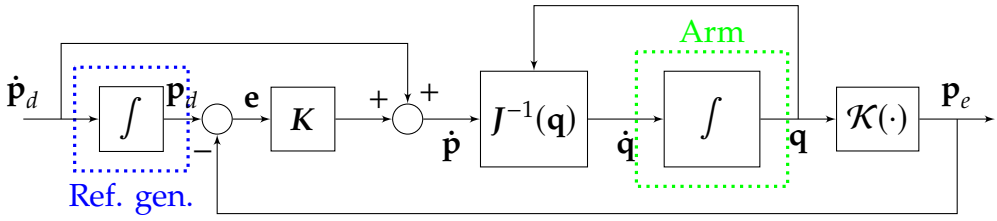


Figure 2.7: Kinematic control scheme of Cartesian motion. The conversion is inherently integrated into the feedback loop, as the sensors measure the joint state, and direct kinematics transform this data into Cartesian space.

Similarly to the previous case if $\boldsymbol{K} > 0$ the system has exponential error convergence. The limitation of this approach arises from the need for real-time

computation of the Jacobian inverse. Singularities must be foreseen and carefully managed. When singularity issues related to the desired motion become relevant, it is necessary to account for the robot's dynamic terms and dissipative effects.

2.6 Dynamic controllers

For applications requiring high operational speeds, varying payload conditions, or direct-drive actuators, the nonlinear coupling terms significantly affect system performance and can no longer be treated as mere disturbances. In such cases, it is recommended to design control algorithms that leverage a detailed understanding of the manipulator's dynamics to compensate for these nonlinear coupling effects. *Centralized algorithms* take advantage of a detailed knowledge of the manipulator dynamics to compensate for the non-linear coupling terms. In this approach the manipulator is not a set of n independent system but it is a *multi-variable, non-linear system*, where the inputs are the $n - joint$ torques and the output the $n - joint$ positions.

On the other hand, in order to generate compensating torques for the nonlinear terms the knowledge of the dynamic model is required, to design a model-based non-linear control approach. This, in turn, creates the requirement to estimate the model parameters.

2.6.1 Position control

Initially a position control approach is introduced. Given a desired constant position $\mathbf{q}_d$ the objective is to design a controller ensuring *global asymptotic stability* of the posture.

The reference position can also be assigned in the operational space, then an inverse kinematic step must be performed before-hand. Alternatively, the error feedback, as will be introduced later[1], can also be viewed as the error in the task space, assuming the availability of sensors to measure the end-effector's pose.

To simplify the procedure, it is assumed that there is no *interaction with the environment*, i.e., $\mathbf{h}_e = \mathbf{0}$. Additionally, based on the dynamic model in equation 2.7, we assume the absence of *static friction*, such that $\mathbf{F}_s = \mathbf{0}$.

The determination of the control input that stabilizes the system around the equilibrium posture is based on the *Lyapunov direct method*. The equilibrium state should consist of the desired joint position with zero joint velocity. A possible

[1]While the control law remains similar, its interpretation and the convergence proof differ.

definition is:

$$\mathbf{x}_{eq} \coloneqq \begin{bmatrix} \tilde{\mathbf{q}} \\ \dot{\mathbf{q}} \end{bmatrix} \qquad \tilde{\mathbf{q}} \coloneqq \mathbf{q}_d - \mathbf{q} \tag{2.9}$$

where $\tilde{\mathbf{q}}$ is the error in joint space.

The candidate Lyapunov function should be strictly positive for all the $\mathbf{x} \neq \mathbf{0}$, and zero in the origin. The condition $V(\mathbf{x}) > 0$ is not sufficient to guarantee asymptotic stability; it is also necessary that $\dot{V}(\mathbf{x}) < 0$ for all $\mathbf{x} \neq \mathbf{0}$. A valid Lyapunov function candidate is

$$V(\tilde{\mathbf{q}}, \dot{\mathbf{q}}) = \frac{1}{2}\dot{\mathbf{q}}^T \boldsymbol{B}(\mathbf{q})\dot{\mathbf{q}} + \frac{1}{2}\tilde{\mathbf{q}}^T \boldsymbol{K}_P \tilde{\mathbf{q}} \tag{2.10}$$

where $\boldsymbol{K}_P$ is a $(n \times n)$ symmetric positive definite matrix. The inertia matrix $\boldsymbol{B}$ is symmetric and positive definite as well, ensuring the positivity of V.

Differentiating equation 2.10 with respect to time, it yields

$$\dot{V}(\tilde{\mathbf{q}}, \dot{\mathbf{q}}) = \dot{\mathbf{q}}^T \boldsymbol{B}(\mathbf{q})\ddot{\mathbf{q}} + \frac{1}{2}\dot{\mathbf{q}}^T \dot{\boldsymbol{B}}(\mathbf{q})\dot{\mathbf{q}} - \dot{\mathbf{q}}^T \boldsymbol{K}_P \tilde{\mathbf{q}} \tag{2.11}$$

where the constant desired joint position is utilized, specifically $\dot{\tilde{\boldsymbol{q}}} = -\dot{\mathbf{q}}$. At this point the system dynamics can be included. Specifically, based on equation 2.7 and the assumptions outlined, the term $\boldsymbol{B}(\mathbf{q})\ddot{\mathbf{q}}$ can be exploited and substituted into $\dot{V}$. After rearranging the terms, the expression becomes:

$$\dot{V}(\tilde{\mathbf{q}}, \dot{\mathbf{q}}) = \frac{1}{2}\dot{\mathbf{q}}^T \left(\dot{\boldsymbol{B}}(\mathbf{q}) - 2\boldsymbol{C}(\mathbf{q}, \dot{\mathbf{q}})\right) \dot{\mathbf{q}} - \dot{\mathbf{q}}^T \boldsymbol{F}_v \dot{\mathbf{q}} + \dot{\mathbf{q}}^T \left(\mathbf{u} - \mathbf{g}(\mathbf{q}) - \boldsymbol{K}_P \tilde{\mathbf{q}}\right) \tag{2.12}$$

It is possible to define the matrix $\boldsymbol{N}(\mathbf{q}, \dot{\mathbf{q}}) = \dot{\boldsymbol{B}}(\mathbf{q}) - 2\boldsymbol{C}(\mathbf{q}, \dot{\mathbf{q}})$. From the conservation of energy principle [4, p.259] it holds that

$$\dot{\mathbf{q}}^T \boldsymbol{N}(\mathbf{q}, \dot{\mathbf{q}})\dot{\mathbf{q}} = 0 \qquad \forall \dot{\mathbf{q}} \tag{2.13}$$

The energy derivative function $\dot{V}$ can be simplified, and with the choice of the input as in equation 2.14 it can be state $\dot{V} < 0$.

$$\mathbf{u} = \mathbf{g}(\mathbf{q}) + \boldsymbol{K}_P \tilde{\mathbf{q}} + \boldsymbol{K}_D \dot{\tilde{\boldsymbol{q}}} \tag{2.14}$$

With this choice of input, it cannot be directly inferred from the Lyapunov function that $\tilde{\mathbf{q}} = 0$ is contained within the equilibrium subspace. Indeed the only condition appearing from $\dot{V}$ on the state $\left[\tilde{\mathbf{q}}^T, \dot{\mathbf{q}}^T\right]^T$ is $\dot{\mathbf{q}} = \mathbf{0}$, as shown in equation 2.15.

$$\dot{V}(\tilde{\mathbf{q}}, \dot{\mathbf{q}}) = -\dot{\mathbf{q}}^T \left(\boldsymbol{F}_v + \boldsymbol{K}_D\right) \dot{\mathbf{q}} \leq 0 \qquad \dot{V} = 0 \iff \dot{\mathbf{q}} = \mathbf{0},\ \forall \tilde{\mathbf{q}} \tag{2.15}$$

Regardless that the provided input u is asymptotically stabilizing the system.

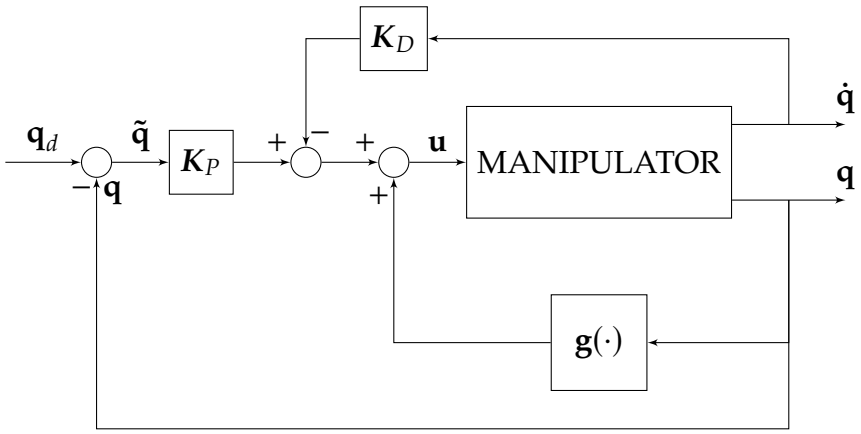


Figure 2.8: Dynamic position control scheme: *gravity compensation*. The bias term $\mathbf{g}(\mathbf{q})$ is calculated and incorporated as a real-time feedforward action. The controller gains are typically diagonal matrices. In particular, to ensure asymptotic stability, it is required that $\boldsymbol{K}_P \succ 0$, $\boldsymbol{K}_D \succ 0$.

By substituting equation 2.14 into the dynamic equation 2.7, it follows that:

$$\boldsymbol{B}(\mathbf{q})\ddot{\mathbf{q}} + \boldsymbol{C}(\mathbf{q}, \dot{\mathbf{q}})\dot{\mathbf{q}} + \mathbf{g}(\mathbf{q}) + \boldsymbol{F}_v\, \dot{\mathbf{q}} = \mathbf{g}(\mathbf{q}) + \boldsymbol{K}_P\tilde{\mathbf{q}} - \boldsymbol{K}_D\dot{\mathbf{q}} \tag{2.16}$$

where at the equilibrium subspace obtained in equation 2.15, where in particular $\dot{\mathbf{q}} = 0$ and accordingly $\ddot{\mathbf{q}} = 0$, most of the terms are simplified and it remains $\tilde{\mathbf{q}} = 0$. The scheme implementing this controller is illustrated in Figure 2.8.

The presence of viscous friction determine the rate of energy dissipation. Due to the uncertainty in the friction term $\boldsymbol{F}_v$, a derivative component has been introduced. The gain $\boldsymbol{K}_D$ enhances the convergence to the equilibrium. The inclusion of the derivative term $\boldsymbol{K}_D$ also contributes to increasing $|\dot{V}|$ along the system trajectories.

The selection of control gains $\boldsymbol{K}_P$, $\boldsymbol{K}_D$ influences the robot's behavior during transient phases and settling times. Defining optimal values across the entire workspace is generally challenging.

Given position control as the objective, the dynamic model is exploited only to compensate the gravity force at each position. For this reason, the presented controller is often referred to as *gravity compensation*, as at equilibrium, where $\tilde{\mathbf{q}} = \mathbf{0}$, $\dot{\mathbf{q}} = \mathbf{0}$, the input is non-zero but corresponds to $\mathbf{u} = \mathbf{g}(\mathbf{q})$.

Notably, the control law remains applicable in discrete time, as all terms are static. The velocity $\dot{\mathbf{q}}$ is measured at each discrete time step. In this implementation, it is challenging to estimate the *rise time* and *settling time* of the system in advance, as these values depend on various factors such as the masses, control gains, initial and target positions, and other dynamic influences.

2.6.2 Trajectory tracking

In this section trajectory controllers will be presented. The reference trajectory is assumed to be provided in joint space, consisting of position, velocity, and acceleration functions 2.17.

$$\mathbf{q}_d(t) \quad \dot{\mathbf{q}}_d(t) \quad \ddot{\mathbf{q}}_d(t) \tag{2.17}$$

If performance is not a critical concern and the target trajectory is relatively slow compared to the manipulator's dynamics, it may be feasible to implement a trajectory tracking algorithm as a sequence of position tracking tasks.

Otherwise, in the general trajectory tracking problem, the complete dynamics must be exploited. Even in an ideal scenario where the *initial conditions* are perfectly known, they may not align with the desired ones, necessitating the use of a closed-loop control strategy.

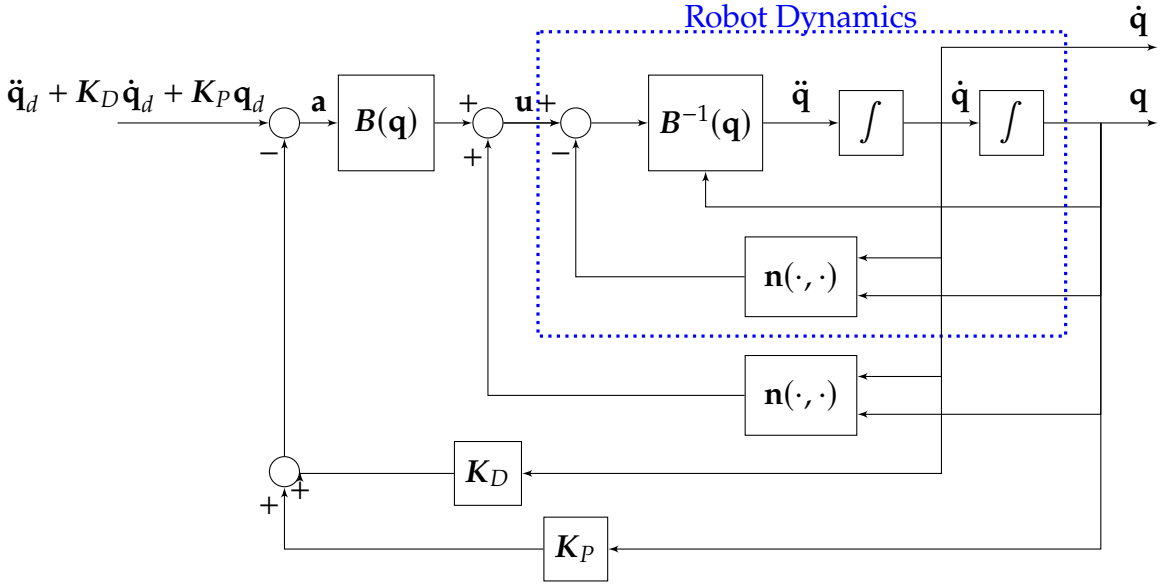


Figure 2.9: Dynamic trajectory controller. It consists of two feedback loops: an inner loop for feedback linearization (FBL) and an outer loop for stabilizing the overall system. Once again, to ensure asymptotic stability, the gains have to be positive definite matrices.

Similarly to the position tracking section we assume absence of forces interacting with the environment, i.e., $\mathbf{h}_e = \mathbf{0}$. and no Coulomb friction forces are acting on the robot. For the sake of clarity, the dynamic model equation 2.7 is presented below with a simplified notation.

$$\begin{aligned} \mathbf{u} &= B(\mathbf{q})\ddot{\mathbf{q}} + C(\mathbf{q},\dot{\mathbf{q}})\dot{\mathbf{q}} + \mathbf{g}(\mathbf{q}) + F_v\,\dot{\mathbf{q}} + F_s\ \mathrm{sgn}(\dot{\mathbf{q}}) \\ \mathbf{u} &= B(\mathbf{q})\ddot{\mathbf{q}} + \mathbf{n}(\mathbf{q},\dot{\mathbf{q}}) \end{aligned} \tag{2.18}$$

The core concept of this controller relies on the precise knowledge of the dy-

namic model and the ability to compute the non linear feedback in *real-time*. The approach involves utilizing the acceleration vector as the manipulator's output reference and adapting the controller to transform the system into a linear structure, enabling the application of a PI/PID control scheme.

$$\ddot{\mathbf{q}} = \boldsymbol{B}^{-1}(\mathbf{q})\left[\mathbf{u} - \mathbf{n}(\mathbf{q}, \dot{\mathbf{q}})\right] \tag{2.19}$$

From the above equation, it is possible to compute an initial term that can be applied to achieve a linearized form. Specifically, the choice presented in equation 2.20 linearizes the system 2.19, resulting in a second-order differential equation of the form $\ddot{\mathbf{q}} = \mathbf{a}$.

$$\mathbf{u} = \boldsymbol{B}(\mathbf{q})\,\mathbf{a} + \mathbf{n}(\mathbf{q}, \dot{\mathbf{q}}) \tag{2.20}$$

At this point, the acceleration vector $\mathbf{a}$ becomes a degree of freedom in designing the controller. In the ideal case, setting $\mathbf{a} = \ddot{\mathbf{q}}_d$ provides a *feedforward action* that enables the tracking of the desired trajectory.

The resulting system $\ddot{\mathbf{q}} = \mathbf{a}$ can be treated as linear, so a *Proportional-Derivative action* can be implemented.

$$\mathbf{u} = \boldsymbol{B}(\mathbf{q})\left[\ddot{\mathbf{q}}_d + \boldsymbol{K}_P\left(\mathbf{q}_d - \mathbf{q}\right) + \boldsymbol{K}_D\left(\dot{\mathbf{q}}_d - \dot{\mathbf{q}}\right)\right] + \mathbf{n}(\mathbf{q}, \dot{\mathbf{q}}) \tag{2.21}$$

In this way the control law 2.21 ensures asymptotical stability if $\boldsymbol{K}_P \succ 0$, $\boldsymbol{K}_D \succ 0$. Indeed it comes the second order differential equation 2.22, namely the *error dynamics* governed by the gains coefficients $\boldsymbol{K}_P$, $\boldsymbol{K}_D$. Such an error arises only if $\mathbf{q}(0)$ and $\dot{\mathbf{q}}(0)$ differ from zero, and it converges to zero at a rate determined by the selected gain matrices $\boldsymbol{K}_P$ and $\boldsymbol{K}_D$.

$$\ddot{\tilde{\boldsymbol{q}}} + \boldsymbol{K}_D\dot{\tilde{\boldsymbol{q}}} + \boldsymbol{K}_P\tilde{\boldsymbol{q}} = \mathbf{0} \tag{2.22}$$

The control law 2.21 is depicted in the scheme of Figure 2.9.

Under feedback linearization control, the robot has a dynamic behavior that is *invariant*, *linear* and *decoupled* in its whole workspace ($\forall(\mathbf{q}, \dot{\mathbf{q}})$). If the control gains are chosen as positive and diagonal matrices the second-order differential equation 2.22 becomes a set of n independent differential equations.

A drawback of this approach is the reliance on the precise knowledge of the dynamic model. In practice, the model is often known with a certain degree of uncertainty due to incomplete information about the manipulator's mechanical parameters, the presence of unmodeled dynamics, and the dependence on end-effector payloads, which are not precisely known and therefore cannot be perfectly compensated.

3

Dynamic parameters

As with many other applications, the primary assumption in designing a controller is the knowledge of the system. While it is impossible to achieve a perfect model, what is required is a sufficiently accurate model for control tasks, where uncertainties can be compensated. This chapter outlines the procedure adopted for the estimation of dynamic parameters.

Computing these parameters based on the design data of the mechanical structure is a complex task. CAD modeling techniques can be employed to compute the inertial parameters of various components (such as links, actuators, and transmissions) based on their geometry and the materials used. However, the estimates derived from these techniques are often imprecise due to the simplifications typically introduced in geometric modeling. Additionally, complex dynamic effects, such as joint friction, cannot be accurately accounted for using this approach.

A heuristic approach could involve disassembling the various components of the manipulator and conducting a series of measurements to determine the inertial parameters. Nevertheless, this technique is challenging to implement and may pose difficulties in accurately measuring the relevant quantities, not to mention the inability to estimate any effects arising from the interaction between links.

3.1 Linear formulation of the problem

The dynamic model written in Lagrangian formulation has the property to be *linear* with respect to a suitable set of *dynamic parameters* [4, pp.259-264]. The Lagrangian equation 2.5 can be reformulated in the linear form as in equation 3.1. Within this context the *generalized forces* ξ comprises only the effort (force or

torque) at the joints.

$$\boldsymbol{B}(\mathbf{q})\ddot{\mathbf{q}} + \boldsymbol{C}(\mathbf{q}, \dot{\mathbf{q}})\dot{\mathbf{q}} + \mathbf{g}(\mathbf{q}) = \boldsymbol{Y}(\mathbf{q}, \dot{\mathbf{q}}, \ddot{\mathbf{q}})\,\boldsymbol{\pi} = \boldsymbol{\tau} \tag{3.1}$$

The matrix $\boldsymbol{Y}(\mathbf{q}, \dot{\mathbf{q}}, \ddot{\mathbf{q}})$ is referred as *regressor matrix* and it map the elements from the parameter's space $\mathbb{R}^p$ to the joint effort space $\mathbb{R}^n$. The regressor matrix incorporates the information related to the system's motion, as it is dependent on the current positions, velocities, and accelerations of the *generalized coordinates* (in this case, the trajectory of joints).

The entries of the matrix $\boldsymbol{Y}(\mathbf{q}, \dot{\mathbf{q}}, \ddot{\mathbf{q}})$ are *non-linear* functions of position, velocity and acceleration of the joints and can be retrieved as in the general procedure 2.4 throughout the Lagrangian formulation. In fact they can be derived from the computation of the kinetic and potential energy link-wise, but with a reformulation of the vectors in the link reference frames.

The primary issue of the standard formulation is the presence of the *center of mass coordinates* and *inertia components*, which enter the expression for kinetic energy in a nonlinear way. By applying the Huygens-Steiner theorem (equation 3.2), the inertia tensor of the link, originally expressed with respect to its center of mass ($\boldsymbol{I}_{\boldsymbol{\ell}_i}^{\boldsymbol{\ell}_i}$), can be re-expressed relative to the link frame origin. In this way it is possible to incorporate both terms in a unique matrix thus making it linearly parameterizable.

$$\bar{\boldsymbol{I}}_{\boldsymbol{\ell}_i}^{\boldsymbol{\ell}_i} = \boldsymbol{I}_{\boldsymbol{\ell}_i}^{\boldsymbol{\ell}_i} + m_i \boldsymbol{S}^T(\mathbf{c}_{i,C_i}^i)\boldsymbol{S}(\mathbf{c}_{i,C_i}^i) \tag{3.2}$$

where $\boldsymbol{S}(\cdot)$ is the skew-symmetric matrix operator,

$$\boldsymbol{S}(\mathbf{r}) = \begin{bmatrix} 0 & -r_z & r_y \\ r_z & 0 & -r_x \\ -r_y & r_x & 0 \end{bmatrix} \quad \text{for} \quad \mathbf{r} = \begin{bmatrix} r_x \\ r_y \\ r_z \end{bmatrix}$$

In equation 3.2, $\boldsymbol{I}_{\boldsymbol{\ell}_i}^{\boldsymbol{\ell}_i}$ is the inertia tensor relative to link i center of mass, m_i its mass and $\boldsymbol{S}(\mathbf{c}_{i,C_i}^i)$ the skew symmetric matrix related to the relative distance, expressed in the link frame coordinates, of the center of mass and the frame origin itself.

The positive definiteness of the inertia matrix $\boldsymbol{I}_{\boldsymbol{\ell}_i}^{\boldsymbol{\ell}_i}$ ensures that the kinetic energy of the system, which depends on the inertia matrix, is always positive for any non-zero angular velocity. In general the inertia matrix is always positive definite when expressed with respect to the center of mass, but when expressed with respect to other points, its positive definiteness is no longer guaranteed, and the matrix may become indefinite, depending on the location and mass distribution. In this case though, the new reference system is not arbitrary, but it is the link frame, parallel with the center of mass one. It yields then that $\bar{\boldsymbol{I}}_{\boldsymbol{\ell}_i}^{\boldsymbol{\ell}_i} > 0$. Indeed it is a summation of a positive definite matrix $\boldsymbol{I}_{\boldsymbol{\ell}_i}^{\boldsymbol{\ell}_i}$ and a positive

semidefinite matrix $m_i \boldsymbol{S}^T(\mathbf{c}^i_{i,C_i})\boldsymbol{S}(\mathbf{c}^i_{i,C_i})$. Moreover, given that m_i is a *positive* scalar, for every vector $\mathbf{x}$, it holds

$$\mathbf{x}^T \left(\boldsymbol{S}^T \boldsymbol{S} \right) \mathbf{x} = ||\boldsymbol{S}\mathbf{x}||^2 \geq 0$$

This shows that $\boldsymbol{S}^T\boldsymbol{S}$ is positive *semidefinite* since $\mathbf{x}\boldsymbol{S}^T\boldsymbol{S}\mathbf{x} \geq 0$ for any $\mathbf{x}$.

Each link is described by a p-dimensional parameter vector $\pi_i \in \mathcal{P} \subset \mathbb{R}^p$ whose components are a non-linear combination of the dynamic quantities of the link, namely: mass and inertia moments of *first* and *second* order. Typically, the space in which the vector resides is a **subset** of $\mathbb{R}^p$, as not all elements of this space can represent a valid parameter vector due to certain constraints imposed over it.

Assuming, for the moment, the absence of external forces due to environmental contact and neglecting the contribution of friction to the dynamics, the parameter vector for each link takes the following form:

$$\boldsymbol{\pi_i} = \left[m_i,\ m_i c_{x,i},\ m_i c_{y,i},\ m_i c_{z,i},\ \bar{I}_{i,xx},\ \bar{I}_{i,xy},\ \bar{I}_{i,yy},\ \bar{I}_{i,xz},\ \bar{I}_{i,yz},\ \bar{I}_{i,zz} \right]^T \quad \boldsymbol{\pi} = \begin{bmatrix} \boldsymbol{\pi_1} \\ \vdots \\ \boldsymbol{\pi_n} \end{bmatrix} \tag{3.3}$$

In this representation, the number of unknowns for each joint consists of ten parameters, denoted as $p = 10$. The inertia components $\bar{I}_{xyz}$ define a unique quadratic and symmetric matrix $\bar{\boldsymbol{I}}_{\ell_i}$, which represents the inertia tensor of link ℓ_i, expressed relative to the origin of the link frame itself.

In the parameter vector $\boldsymbol{\pi}_i$, defined above, the relative distance $\mathbf{c}^i_{i,C_i}$ between the link and center of mass frames enters as the *first order momentum*. To simplify the notation equation 3.2 can be reformulated to be directly dependent on the parameter vector's entries. From 3.2, and recognizing that $m_i > 0$, it yields

$$\bar{\boldsymbol{I}}^{\boldsymbol{\ell_i}}_{\boldsymbol{\ell_i}} = \boldsymbol{I}^{\boldsymbol{\ell_i}}_{\boldsymbol{\ell_i}} + \boldsymbol{S}^T(m_i \mathbf{c}^i_{i,C_i})\boldsymbol{S}(m_i \mathbf{c}^i_{i,C_i})$$

where $m_i \mathbf{c}^i_{i,C_i} = \left[m_i c_{x,i},\ m_i c_{y,i},\ m_i c_{z,i} \right]^T$ is the first moment of inertia of link ℓ_i, namely the components two to four of $\boldsymbol{\pi}_i$. For the parameter set of link ℓ_i to be feasible, the barycentric inertia must satisfy $\boldsymbol{I}^{\boldsymbol{\ell_i}}_{\boldsymbol{\ell_i}} > 0$. Accordingly, based on the aforementioned considerations, it follows that:

$$\bar{\boldsymbol{I}}^{\boldsymbol{\ell_i}}_{\boldsymbol{\ell_i}} - \boldsymbol{S}^T(m_i \mathbf{c}^i_{i,C_i})\boldsymbol{S}(m_i \mathbf{c}^i_{i,C_i}) > 0 \tag{3.4}$$

It is important to emphasize the property of positive definiteness, as it will be utilized in the following section.

As outlined in section 2.4, the most convenient approach to describe the

friction in the joints is through a linear formulation. At this stage, it is beneficial to incorporate Coulomb and viscous friction parameters into the parameter vector. A possible extension of the aforementioned vector $\boldsymbol{\pi}$ is represented by $\boldsymbol{\theta}$ in equation 3.5.

Additionally, the regressor matrix must be adjusted to align with the order of the new parameter vector. Assuming that the friction in each joint is independent of the states of the other joints, the matrices $\boldsymbol{F_v}$ and $\boldsymbol{F_s}$, representing viscous and static friction respectively, are diagonal. Considering the linear model of the form: $\tau_i = f_{v_i}\dot{q}_i + f_{s_i}\operatorname{sgn}\dot{q}_i$, then it is possible to modify the regressor matrix as described in equation 3.6. This adjustment corresponds to the extended parameter vector $\boldsymbol{\theta}$ in equation 3.5.

$$\boldsymbol{\theta} = \left[\boldsymbol{\pi_1}^T \dots \boldsymbol{\pi_n}^T,\ f_{v_1} \dots f_{v_n},\ f_{s_1} \dots f_{s_n}\right] \tag{3.5}$$

$$\boldsymbol{\Phi}(\mathbf{q}, \dot{\mathbf{q}}, \ddot{\mathbf{q}}) = \left[\begin{array}{c|ccc|ccc} & \dot{q}_1 & & & \operatorname{sgn}\dot{q}_1 & & \\ \boldsymbol{Y}(\mathbf{q}, \dot{\mathbf{q}}, \ddot{\mathbf{q}}) & & \ddots & & & \ddots & \\ & & & \dot{q}_n & & & \operatorname{sgn}\dot{q}_n \end{array}\right] \tag{3.6}$$

The system dynamics, incorporating the friction effects as described in Equation 2.7, can be expressed as: $\boldsymbol{\tau} = \boldsymbol{\Phi}(\mathbf{q}, \dot{\mathbf{q}}, \ddot{\mathbf{q}})\,\boldsymbol{\theta}$ which can clearly be embedded into a linear regression framework for parameter estimation. At this stage, the dimension of $\boldsymbol{\theta}$ is np, where

$$p = 12 \tag{3.7}$$

and n represents the number of degrees of freedom of the manipulator. For the regressor matrix, it holds: $\boldsymbol{\Phi} \in \mathbb{R}^{n \times np}$.

3.2 Optimization problem

The selection of a parameter estimation method involves a trade-off between accuracy and implementation complexity. Least squares (LS) parameter estimation is a non-iterative approach that determines parameter estimates in a single step using singular value decomposition.

3.2.1 Identifiability

Consider the linear regressor model of the form $\mathbf{y} = \boldsymbol{\Phi}\boldsymbol{\theta} + \mathbf{w}$, where $\mathbf{w} \sim \mathcal{N}(0, \boldsymbol{\Sigma})$ (with $\boldsymbol{\Sigma} > 0$), $\boldsymbol{\Phi} \in \mathbb{R}^{M \times np}$ with $M \geq np$ then we can assert that global identifiability is achieved if the Fisher information matrix is invertible. In this specific case, the Fisher information matrix can be computed in closed form as $\boldsymbol{I}(\boldsymbol{\theta}) = \boldsymbol{\Phi}^T\boldsymbol{\Sigma}^{-1}\boldsymbol{\Phi}$ which is clearly invertible if and only if the regressor matrix $\boldsymbol{\Phi}$ is *full column rank*.

To achieve this condition, having fixed the size of the parameter vector, it is convenient to increase the number of samples to be collected, while maintaining a trade-off with the complexity of the problem.

$$\hat{\boldsymbol{\theta}}_{ML} \equiv \hat{\boldsymbol{\theta}}_{LS} = \underset{\theta}{\operatorname{argmin}} \|\mathbf{y} - \boldsymbol{\Phi}\,\theta\|^2_{\Sigma^{-1}}$$

Under the assumption of Gaussian noise, the least-squares estimation provides the same estimator as the maximum likelihood method, thus ensuring to be an Unbiased Minimum Variance of Error (UMVE) estimator.

On the other hand, in the general case (non linear problem and non Gaussian noise), $\hat{\boldsymbol{\theta}}_{LS}$ is a *minimum variance* over the class of the *unbiased* and *linear* estimators of $\boldsymbol{\theta}$.

The knowledge of the noise covariance matrix makes it possible to discriminate data according to its accuracy. In fact, weighting with the inverse of the covariance matrix ensures that torque measurements with higher noise are less influential in parameter estimation.

3.2.2 Least square problem

Given the lack of prior information on the variance of the measurement error, and the fact that the parameters have physical significance and require feasibility constraints, a constrained least-squares problem was preferred.

The identification procedure is performed by collecting $M >> np$ joint torque, position and velocity samples so as to avoid *ill-conditioning* of matrix $\bar{\boldsymbol{\Phi}}$. The samples are collected along a trajectory, as described in the following sections, while the acceleration samples are computed through differentiation. For each sample $(\boldsymbol{\tau}_k,\ \mathbf{q}_k,\ \dot{\mathbf{q}}_k,\ \ddot{\mathbf{q}}_k)$ with $k = 1, \ldots, M$ we have

$$\boldsymbol{\tau}_k = \boldsymbol{\Phi}(\mathbf{q}_k, \dot{\mathbf{q}}_k, \ddot{\mathbf{q}}_k)\,\boldsymbol{\theta}$$

By stacking the collected samples into a matrix formulation the data takes the following form

$$\begin{bmatrix} \boldsymbol{\tau}_1 \\ \vdots \\ \boldsymbol{\tau}_M \end{bmatrix} = \begin{bmatrix} \boldsymbol{\Phi}(\mathbf{q}_1, \dot{\mathbf{q}}_1, \ddot{\mathbf{q}}_1) \\ \vdots \\ \boldsymbol{\Phi}(\mathbf{q}_M, \dot{\mathbf{q}}_M, \ddot{\mathbf{q}}_M) \end{bmatrix} \boldsymbol{\theta} = \bar{\boldsymbol{\Phi}}(\mathbf{q}, \dot{\mathbf{q}}, \ddot{\mathbf{q}})\,\boldsymbol{\theta} = \bar{\boldsymbol{\tau}} \tag{3.8}$$

The solution to the ordinary least squares problem with a non-square matrix $\bar{\boldsymbol{\Phi}}$ can be obtained through pseudo-inversion:

$$\hat{\boldsymbol{\theta}} = \bar{\boldsymbol{\Phi}}^{\dagger}\,\bar{\boldsymbol{\tau}}$$

Utilizing this procedure does not guarantee to obtain an absolute minimum, as it is possible to get stuck in local minima. Furthermore, this solution does not guarantee feasibility, as certain parameters are constrained by physics laws.

3.2.3 Physical consistency

In general the solution vector $\hat{\boldsymbol{\theta}}$ may be physically inconsistent, as negative masses or non positive definite inertia matrices. This of course does not reflect reality, but may be due to measurement noise, local minima, model errors.

Following the condition 3.4, the set of all feasibility constraints, for each link ℓ_i, can be written as

$$\begin{cases} m_i > 0 \\ \bar{\boldsymbol{I}}_{\ell_i} - \boldsymbol{S}^T(m_i \mathbf{c}_i)\boldsymbol{S}(m_i \mathbf{c}_i) \succ 0 \end{cases} \tag{3.9}$$

The set of all feasible link parameter vectors, called $\mathcal{P}$, can then be defined as

$$\mathcal{P} := \left\{ \boldsymbol{\pi} \in \mathbb{R}^{10n} \;:\; m_i > 0, \bar{\boldsymbol{I}}_{\ell_i} - \boldsymbol{S}^T(m_i \mathbf{c}_i)\boldsymbol{S}(m_i \mathbf{c}_i) \succ 0 \,|\, i = 1, \ldots, n \right\} \tag{3.10}$$

In order to overcome the feasibility problem of the solution, constraints are then introduced on the components of the parameter vector. In this case, given the friction model described previously (refer to 2.6), the positivity of its coefficients is imposed as an additional constraint. The implemented algorithm is the following non-linear constrained optimization problem

$$\begin{aligned} \hat{\boldsymbol{\theta}}_{LS} = \quad & \underset{\theta}{\operatorname{argmin}} \quad \|\bar{\boldsymbol{\tau}} - \bar{\boldsymbol{\Phi}}\,\boldsymbol{\theta}\|^2 \\ & \text{s.t.} \quad \boldsymbol{I}_{\ell_i}^{\ell_i} > 0 \qquad\qquad i = 1, \ldots, n \\ & \qquad\; m_{i_{min}} < m_i < m_{i_{max}} \quad i = 1, \ldots, n \\ & \qquad\; f_{v_i} > 0 \qquad\qquad i = 1, \ldots, n \\ & \qquad\; f_{s_i} > 0 \qquad\qquad i = 1, \ldots, n \end{aligned} \tag{3.11}$$

The imposed conditions on masses and friction coefficients have a clear physical meaning: masses cannot be negative (nor zero), and the friction effect must oppose the motion. Given that it is possible to weigh the links while they are unmounted, a range of valid mass values is defined as a constraint, rather than simply imposing the positivity of the masses.

Whether the solution requires it, additional constraints can be introduced, such as: total weigh bound, positivity of the diagonal components of the inertia matrix, or limits on the center of mass coordinates.

Additional constraints are indeed implemented. Since the centers of mass for each link must lie within their respective convex hulls, bounds on the distance vector (specified by $r_{i_{LB}}$ and $r_{i_{LB}}$) are applied. Consequently, limits on the first

moment of inertia are also incorporated.

$$\begin{cases} m_i \mathbf{c}_i - m_i \mathbf{r}_{i_{LB}} & \geq 0 \qquad i = 1, \ldots, n \\ -m_i \mathbf{c}_i + m_i \mathbf{r}_{i_{UB}} & \geq 0 \qquad i = 1, \ldots, n \end{cases} \tag{3.12}$$

A constraint on the total mass of the robot is also imposed, expressed as follows:

$$m_{tot_{min}} \leq \sum_i m_i \leq m_{tot_{max}} \tag{3.13}$$

The idea behind imposing these constraints is to reduce the solution set $\mathcal{P}$, thereby preventing the occurrence of local minima or unfeasible solutions. However, this approach may lead to the problem becoming unsolvable; in such cases, a constraint relaxation procedure will be implemented, or an alternative dataset will be utilized.

Regarding the inertia matrices of each link, special attention is required, as the property of positive definiteness **cannot** be expressed in linear form. Despite this, it is possible to formulate the constraints the form of a Linear Matrix Inequality (LMI), a common framework in control theory [5]. This formulation implies that the set defined by conditions of physical feasibility is inherently convex, thereby enabling the achievement of global optima in optimization problems. Consequently, this permits the formulation of Semi-Definite Programming (SDP) problems, for which effective and fast-solving techniques are well established.

Furthermore, this constraint is not imposed directly on the matrix derived from the parameter vector $\boldsymbol{\pi}$ but must instead be applied to the matrix expressed with respect to the center of mass. In the vector $\boldsymbol{\pi}_i$ the inertia matrix referenced in the link frame, obtained through Huygens-Steiner theorem 3.2, which can be expressed as follows:

$$\bar{\boldsymbol{I}}_{\ell_i}^{\ell_i} = \begin{bmatrix} I_{i,xx} + m_i(c_{y,i}^2 + c_{z,i}^2) & -I_{i,xy} - m_i c_{x,i} c_{y,i} & -I_{i,xz} - m_i c_{x,i} c_{z,i} \\ * & I_{i,yy} + m_i(c_{x,i}^2 + c_{z,i}^2) & -I_{i,yz} - m_i c_{y,i} c_{z,i} \\ * & * & I_{i,zz} + m_i(c_{x,i}^2 + c_{y,i}^2) \end{bmatrix} \tag{3.14}$$

where $\boldsymbol{I}_{\ell_i}^{\ell_i}$ is expressed in the coordinate frame i, but it represents the inertia tensor relative to the center of mass of link i. On matrix $\boldsymbol{I}_{\ell_i}^{\ell_i}$ the constraint 3.15 holds, which is then what will be imposed in the optimization problem 3.11.

$$\boldsymbol{I}_{\ell_i}^{\ell_i} = \begin{bmatrix} I_{i,xx} & I_{i,xy} & I_{i,xz} \\ I_{i,xy} & I_{i,yy} & I_{i,yz} \\ I_{i,xz} & I_{i,yz} & I_{i,zz} \end{bmatrix} \succ 0 \tag{3.15}$$

3.3 Optimization algorithms

The manifold generated by the cost function of the problem 3.11 contains multiple local minima and therefore starting from different initial conditions helps to span as much as possible the cost function manifold.

Different algorithms have been implemented to address the problem, which has a linear cost but with non linear constraints. Since the optimization problem is scalar, under non-linear constraints the Sequential Least Squares Quadratic Programming (SLSQP) method was used. By running several runs of the algorithm from different, random starting points, the choice falls on the least-cost solution.

Following the idea of the optimization algorithm proposed in [1], a second solver using the Simulated Annealing (SA) technique has been employed. SA search for an optimal solution by exploring the solution space and probabilistically accepting worst solutions to escape local minima. It is particularly effective for problems characterized by a large number of local minima. Constraints are incorporated into the optimization process by penalizing them as part of the objective function. Unlike gradient-based methods SA can escape local minima, making it well-suited for non-convex optimization problems.

3.4 Trajectory parametrization

In theory, the invertibility of the square matrix $\bar{\mathbf{\Phi}}^T \bar{\mathbf{\Phi}}$ is sufficient to ensure identifiability. However, in practice, some parameters may be poorly identified due to inadequately exciting input.

It can generally be concluded that the selection of the trajectory should favor polynomial forms that are sufficiently rich to enable an accurate evaluation of the identifiable parameters [3]. This corresponds to ensuring a low condition number of the matrix $\bar{\mathbf{\Phi}}$ along the trajectory.

The choice of the trajectory is a trade-off between several aspects:

1. The joint positions should cover the majority of the manipulator's reachable workspace and be executed at varying accelerations to accurately estimate the moments of inertia.
2. On the other side, such trajectories should *not excite* any unmodeled dynamic effects, such as joint elasticity or link flexibility, as these effects could lead to unreliable estimates of the dynamic parameters.
3. The complexity of parametrization and tracking plays a significant role. The former can increase the computational burden on the code, while the latter may impact the performance of the arm.
4. *Periodic* and *band-limited* signals are more accurate to process, increasing the accuracy of the estimates.

Starting from the last point, the most immediate signal providing a limited bandwidth are trigonometric functions, whose Fourier transforms are frequency pulses.

$$q_i(t) = q_{i,0} + \sum_{k=1}^{N} \left(\alpha_{i,k} \sin(k\omega_0 t) + \beta_{i,k} \cos(k\omega_0 t)\right) \tag{3.16}$$

Equation 3.16, shows the components of the implemented desired trajectory, namely $\mathbf{q}_d(t) = \left[q_1(t), \ldots, q_n(t)\right]^T$.

Since the manipulator can be controlled through velocity commands the actual provided signal will be $\dot{\mathbf{q}}$ whose components can be arithmetically derived as in equation 3.17.

$$\dot{q}_i(t) = \sum_{k=1}^{N} \left(\alpha_{i,k} k\omega_0 \sin(k\omega_0 t) - \beta_{i,k} k\omega_0 \cos(k\omega_0 t)\right) \tag{3.17}$$

3.4.1 Amplitude coefficients

The coefficients characterizing the trajectory $\alpha_{i,k}$ and $\beta_{i,k}$ define the amplitude of the signal, along with the bias term $q_{i,0}$. These coefficients can be determined either solving a non linear optimization problem with constraints imposed by the robot motion, or through trial and error. In this context, they are randomly selected to meet specific criteria:

- a safety check is performed based on the joint positions to ensure that the links do not come into contact with each other.
- Direct kinematics is employed to verify the absence of contact with the ground or the external environment.
- Verify whether the corresponding velocities 3.17 remain within the safe limits for motor operation.

The position farthest from the ground for the arm is the vertical one, which, according to the DH parametrization outlined in Section 2.2.2, corresponds to setting the second joint bias to 90 degrees, in addition to which a random term is added.

3.4.2 Frequency coefficients

Equation 3.16 explicitly shows the relation between the spectral deltas. Specifically $\omega_0 = 2\pi f_0$ defines the fundamental harmonic, while as k increase we have harmonic multiples of this. The term N is an arbitrary factor that defines the number of spectral lines (or frequency deltas) that will appear in the frequency domain.

Different tests have been performed varying both f_0 and N. It has been noted that a good compromise is $f_0 = 0.1\,Hz$ and $N = 5$. In particular $f_0 = 0.1\,Hz$ for

all joints, which correspond to an angular frequency of $\omega_0 = \frac{2\pi}{10}\, rad/s$, is chosen to be an integer multiple of the sampling frequency.

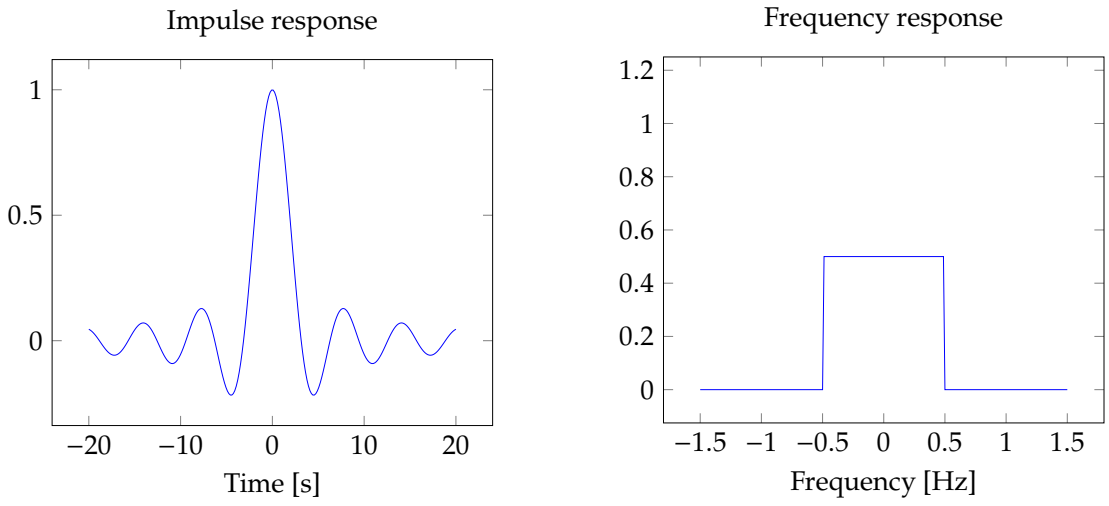
The selection of the fundamental frequency of the trajectory is arbitrary. However, it is important to note that, in combination with the number of spectral lines N, it determines a highest harmonic at Nf_0. Given that the sampling time cannot be improved, it is crucial to consider, as per Shannon's theorem, that the sampling frequency must satisfy $f_{sampling} > 2Nf_0$ to ensure an accurate reconstruction of the signal.

In the case under study, the sampling frequency is set to $f_{sampling} = 1kHz$ while the highest harmonic is $2f_0N = 1Hz$ which clearly satisfies the requirements of the sampling theorem thereby avoiding aliasing effects.

3.5 Filtering trajectory

As discussed in previous chapters, it is assumed that the low-level controllers are properly tuned. The low-level feedback will be utilized to execute trajectories without relying on the dynamic model of the arm. Consequently, velocity commands will be issued to the joints, and it is expected that these commands will be accurately tracked.

Interconnection between links, vibrations and mechanical disturbances, encoder's noise and many other factors introduce noise in the actual joint trajectories that will clearly deviate from the commanded ones. The noise introduced is challenging to model and, for general trajectories, difficult to effectively filter. However, selecting control commands with a known frequency spectrum facilitates the filtering process, making noise mitigation more manageable.



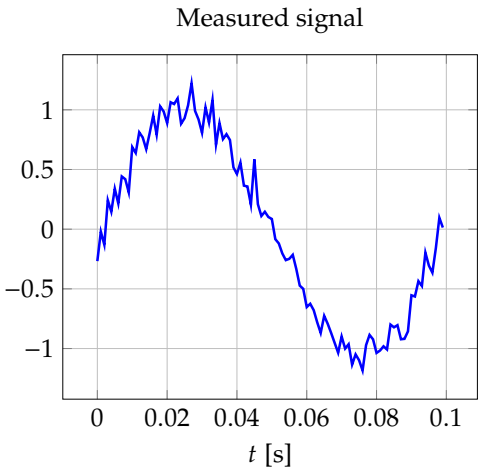
(a) Time domain $sinc(x)$ function.

(b) Rectangular window in the frequency domain.

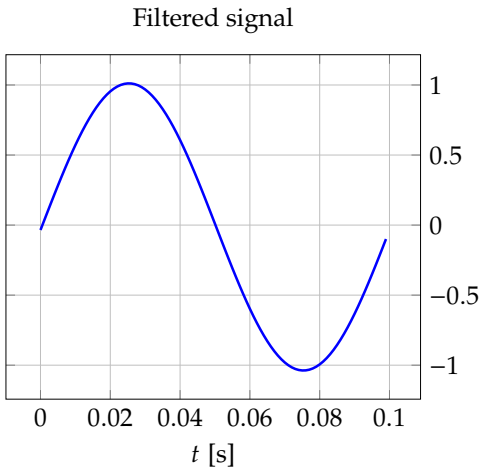
Figure 3.1: Impulse response and frequency response of an ideal LPF.

The motion of the arm will present the same harmonics spectrum as that of the commanded trajectories but the measurements will introduce more and different harmonics then the expected ones. About the order of those harmonics they are expected to be decreasing in frequency, since the behavior of mechanical systems typically exhibit band-limited behavior, and the controller itself operates within a limited bandwidth.

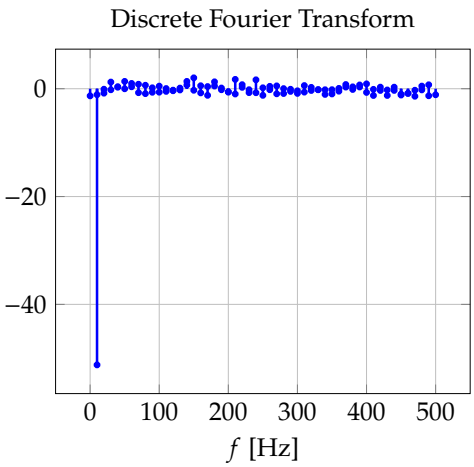
The pre-processing step is meant to clean the measured signals before the identification step. By way of example in Figure 3.2, is depicted a simplified version of the expected behavior.



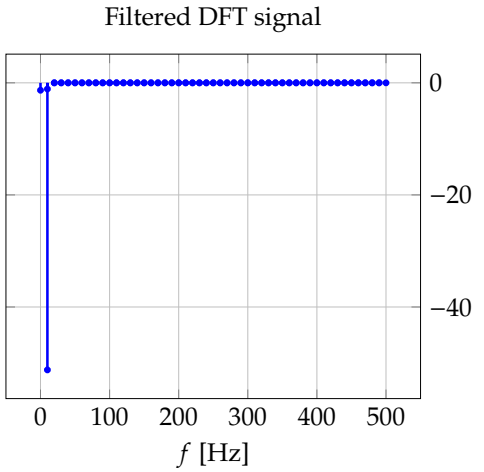
(a) Original noisy signal. $\sin(\omega t)$ with $\omega = 2\pi 10\,[rad/s]$



(b) Anti-transformed filtered signal.



(c) DFT of the original signal.



(d) Rectangular window filter.

Figure 3.2: Filtering process in the frequency domain: the example signal consists of a single frequency, meaning that the relevant information is concentrated in the first peak of the frequency spectrum. A rectangular window filter is subsequently applied to isolate this frequency.

The measured signal is assumed to exhibit addictive Gaussian noise (Figure 3.2a) which has spectral components distributed throughout the frequency range. In this context, the simplest filter that can be implemented is a rectan-

gular window. However, certain precautions must be observed when using this filter, particularly because it is a **non causal** filter, requiring future samples in the time domain. In this case, this limitation does not pose an issue, as the signal is processed post-collection, meaning the entire signal is available for analysis.

The rectangular filter nullifies all frequencies outside the range of interest. In the example, with a signal frequency of $f_0 = 10\,[Hz]$, the filter is configured to eliminate all frequencies above $f_f = 11.5\,[Hz]$. The resulting frequency-domain representation is shown in Figure 3.2d. Figure 3.2b illustrates the reconstructed signal after the inverse transformation step.

The results are not perfectly free of noise because it cannot be removed from the selected frequencies.

With regard to filtering the torque measurements, there is no prior knowledge of the resulting signal's frequency spectrum, unlike the previous case. In general, a low-pass filter can be applied, with the bandwidth and attenuation tuned according to the spectrum of the measured data.

3.5.1 Acceleration derivation

Numerical differentiation amplifies the noise present in the measurements, though the acceleration is computed starting from the filtered version of the velocity. The numerical differentiation is performed through second order approximation of the continuous time derivative, by means of python built-in routine.

An alternative approach involves operating in the frequency domain and multiplying the filtered spectrum by the continuous-time frequency response of a pure differentiator, represented by $j\omega$. While this convolution is suitable for continuous-time signals, it necessitates careful consideration during implementation in discrete-time.

3.6 Payload estimation

Finally, it is worth noting that the technique described above can be extended to identify the dynamic parameters of an unknown payload at the manipulators end-effector. In this scenario, the payload is treated as a structural modification of the last link, allowing for the identification of the modified link's dynamic parameters. If a force sensor is available at the manipulators wrist, it becomes feasible to directly characterize the payloads dynamic parameters based on the force sensor measurements.

4 Hardware

When the joints are not mounted in the arm it is possible to estimate the parameters characterizing the motors and the gear boxes. Since the two radial gear boxes are printed from the same CAD file, it is reasonable to assume that the parameters are more or less the same. In practice of course this is not true indeed many factors may change the parameters, such as: the printing precision, the manual assembling, the motor shaft and the motor itself (even if it is the same model).

Since the final goal is to drive the entire body via torque commands at the joints, it is necessary to have a relation between the commanded signal to the motors and the applied torque at the joint level. The presence of gears affect this relation too, in fact in order to describe this map one has to extrapolate as much information as possible from the transmission chain.

The friction parameters that will be retrieved in this section will be used as initial guess also for the full arm model. The tests are performed with single motor in a test bench, so each joint is not solicited by the mass and the inertia of the others. Hence the derived values will not be congruent with those of the total model but a reasonable starting point.

4.1 SPM brushless motors

As discussed in previous chapters the efficiency is a key-point of the project. The choice to use brushless motors is related to weight and efficiency reasons. Indeed those kind motors are typically more efficient than dc motors. On the other hand a more complex and expensive control electronics is required.

Brushless motors, also called BrushLess Direct Current (BLDC) motors, are synchronous motors controlled by periodic voltage waves in the three phases. There are different type of brushless motors, but the choice addressed to Surface

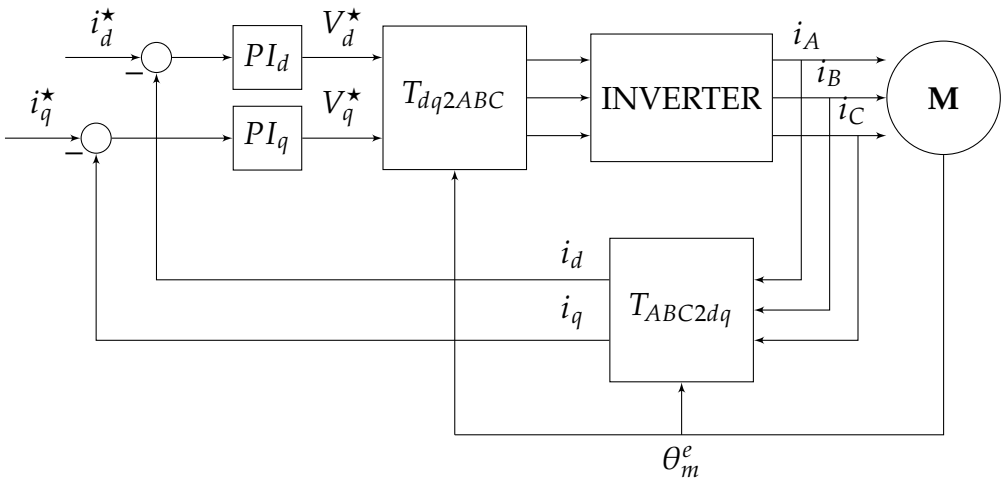


Figure 4.1: FOC control scheme for brushless motors.

Permanent Magnet (SPM) motors. Those are characterized by the presence of p permanent magnets in the rotor and n coils in the stator. In BLDC motors there is no contact between the stator and the rotor, indeed the power is transferred via the magnetic field induced by the current flowing in the stator's windings. In order to obtain a constant torque along a rotor's turn the coils are divided in three groups, called phases. Those are physically separated of $\frac{2\pi}{3}\,rad$ but also the commanded current signal are phase shifted of $\frac{2\pi}{3}\,rad$.

In a synchronous electric motor the torque is generated by a misalignment between the magnetic fields of stator and rotor. In some way the brush commutator contacts are replaced by an electronic sensor that detects the angular position of the rotor allowing synchronisation. The angular frequency of the sinusoidal current injected in the stator phases determine the angular speed of the rotor. The current in the three phases is, with the knowledge of the angular position of the rotor, transformed in the so called *synchronous frame*, a subset of the complex domain. In this complex space the real and imaginary axes are called **direct** and **quadrature** axes respectively. In this domain it is possible to implement the so called Field Oriented Control (FOC). Given that the new reference frame embed information on rotation, sinusoidal quantities are mapped into constant values, hence the control law comes straight forward.

In general a *Proportional-Integral* control is implemented in both direct and quadrature variables as showed in Figure 4.1. Even though the chain made of transformations, inverter and motor is not linear there are techniques that allow a simplified study of the problem.

For classical SPM motors, where the magnets are mounted externally the rotor, the instantaneous power input to the motor is obtained as the sum of the instantaneous power of the single phases, namely: $P_{in} = v_A i_A + v_B i_B + v_C i_C$. The equivalent version of this formula in the synchronous reference frame is

equation 4.1.

$$P_{in}(t) = \frac{3}{2}\left[v_q i_q + v_d i_d\right] \tag{4.1}$$

According to the electrical equation of the motor, the input power can be expressed as the sum of three terms:

- P_J: the joule losses in the winding resistance. In particular $P_J = \frac{3}{2}R(i_d^2 + i_q^2)$.
- P_W: the power associated with the variation of the magnetizing energy stored in the magnetic circuit.
- P_{em}: the remaining power, which is converted in mechanical power output of the motor. $P_{em} = \tau_{em}\,\omega_m$

From the motor's electrical equation, it can be concluded that the torque generated is linearly proportional to the current: equation 4.2. Further details are provided in Section A.1 of the appendix.

$$\tau_{em} = K_i\, i_q \tag{4.2}$$

Where ω_m is the electrical angular velocity[1] of the motor, τ_{em} is the output torque, i_q is the current in the **quadrature** axis. K_i is the torque constant, it depends on physical parameters of the motor. In particular $K_i = \frac{3}{2}p\Lambda_m$, where p is the number of pole pairs and Λ_m is the flux linkage due to the permanent magnets. In other words the electromagnetic torque is given by the interaction between the flux linkage produced by the PMs on the rotor and the stator current component in quadrature. The current in the **direct** axis i_d does not play any role in the output torque, but it still affect the joule losses P_J. In normal flux configurations, it is convenient to set the reference $i_d^\star$ to zero in the control loop of Figure 4.1.

In general commercial BLDC motors are characterized by the K_v value and the number of rotor pair poles. The K_v rating of a brushless motor is the ratio of the motors unloaded speed (in rpm) to the **peak** voltage on the wires connected to the coils (in Volts). From this constant it is possible to determine the torque constant of the motor.

Assuming that the low level controller keeps the current in the *direct axis* at zero and for low quadrature axis current, then the torque constant K_i can be computed from the motor constant K_v with the formula 4.3.

$$K_i = \frac{3}{2}\,\frac{1}{\sqrt{3}}\,\frac{60}{2\pi}\,\frac{1}{K_v} \tag{4.3}$$

[1]Electrical speed refers to the speed of the rotating magnetic field as opposed to mechanical speed being the actual speed of the rotor. The electrical speed is number of polar pairs times the mechanical speed.

	p	$K_v \left[\frac{rpm}{V}\right]$	$K_i \left[\frac{Nm}{A}\right]$	$I_{max}[A]$	$\tau_{max}[Nm]$
mn4004	12	300	0.027	9.0	0.243
mn4006	12	380	0.022	16.0	0.352
mn5006	14	300	0.027	20.0	0.54

Table 4.1: Electrical parameters of the motors used in the project: p denotes the number of pole pairs, and together with K_v (motor velocity constant) and I_{max} (maximum current), these specifications can be obtained from the datasheet.

In the case of study the motors are of two different types. The motor constant K_v comes from the motor' specifications while the torque constant is computed via equation 4.3.

Controller gains

The SPI message is characterized by three gain fields, namely: K_p, K_d, K_i. The imposed control law is reported in equation 4.4, where the FF term is a possible bias current value, also named feed-forward.

$$i_q(t) = K_p\,\tilde{\theta}(t) + K_d\,\tilde{\omega}(t) + K_i \int_0^t \tilde{\theta} dt + FF \tag{4.4}$$

where: $\tilde{\theta} = \theta_{ref} - \theta_{meas}$ and $\tilde{\omega} = \omega_{ref} - \omega_{meas}$ are position and velocity errors. The gain values are meant to design a complete PID over the position reference, but they can be used also to implement a PI control over the velocity reference.

Indeed equation 4.4 can be seen as a **PI** control for velocity tracking as equation 4.5

$$I_q(s) = \left(K_p + \frac{K_i}{s}\right)\Delta\Omega(s)$$

Which in the time domain is equivalent to:

$$i_q(t) = K_p\,\tilde{\omega}(t) + K_i \int_0^t \tilde{\omega}(\tau)\,d\tau = K_p\,\tilde{\omega}(t) + K_i\tilde{\theta}(t) \tag{4.5}$$

imposing $K_i = 0$ and $FF = 0$ in equation 4.4, it is equivalent to 4.5 with a different nomenclature. Despite the different names, the coefficients have the same meaning.

4.2 Gearbox

The torque produced by the bare motors is not enough to the purpose, even in maximum current condition, which is not ideal. A gearbox is then placed in between the motor and the load in order to increase the output torque.

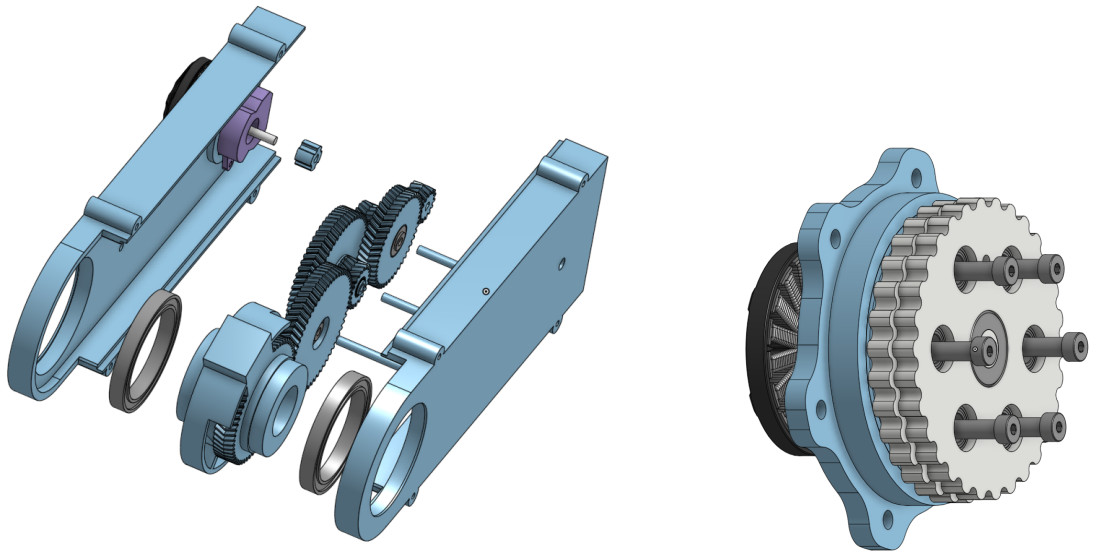
The gearbox reduction ratio is what characterizes the torque increase. This

value is constant and it is defined as the ratio between the output and motor speed, namely: $k_g = \frac{\omega_L}{\omega_M}$. Under this definition it is necessary to have $k_g < 1$ to increase the output torque. In the following, it is convenient to distinguish the output torque as the load (L) torque, while referring to the input as the motor (M) torque.

$$\begin{cases} \omega_L = k_g\, \omega_M & (4.6) \\ \tau_L = \eta_g \dfrac{\tau_M}{k_g} & (4.7) \end{cases}$$

In practical applications of gearboxes, the input power is not fully transmitted to the output due to inherent losses within the gear mechanism. By applying power balance equations that account for the gear's efficiency η_g, the resulting torque equation 4.7 can be derived.

In the project two types of gears were used: herringbone and cycloidal gear. The former type has been used for radial joints with a gear ratio of $k_g = \frac{1}{27}$, while the latter has been used for the axial joints with $k_g = \frac{1}{31}$. As the *cycloidal gear* transmits motion between the two internal discs through friction, the friction is expected to be higher than the herringbone gear. However, the former type offers greater compactness and makes it easier the achievement of higher reduction ratios.



(a) Herringbone gears used for radial joints. $k_g = \frac{1}{27}$

(b) Cycloidal gears used for axial joints. $k_g = \frac{1}{31}$

Figure 4.2: Types of gearboxes used in the manipulator.

4.3 Control board and Encoder

Regarding the components of the control scheme, a custom-designed board developed in-house was selected, enabling the simultaneous control of two

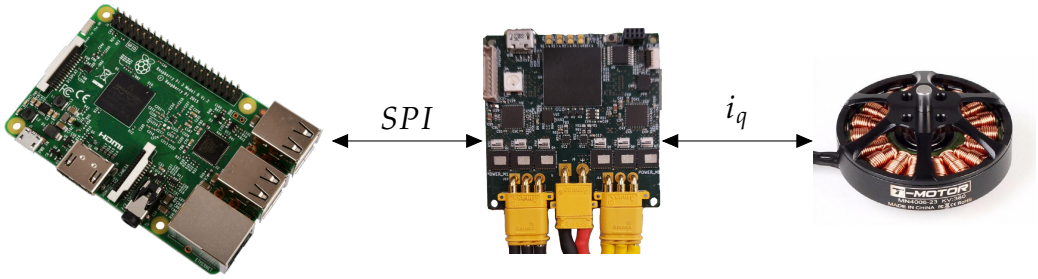


Figure 4.3: Simplified motor control chain diagram: The control algorithm is executed on the Raspberry Pi, which transmits commands to the brushless motor control board via the *SPI* protocol. The control board then regulates the motors by applying the desired current.

motors. Due to the three-phase nature of the system, the control board must incorporate a microcontroller capable of managing the motor's low-level operations. Since the final objective is to control the motor via torque commands, the board must support this functionality. Moreover certain experiments require position or velocity control of the motors. Standard Electronic Speed Controllers (ESCs) for drones, which also typically use brushless motors, rely on the relationship between speed and propeller thrust, thereby prevent the need for torque feedback.

The project utilized custom boards developed in-house at the LAAS-CNRS laboratory. Each board is capable of controlling a pair of motors. The firmware was modified to meet the specific requirements of the project, particularly by incorporating the capability to monitor the input power of the motors. This feature was added to enable energy consumption monitoring and to support certain experimental procedures.

This board operates with an internal execution frequency of 40 kHz, meaning that the internally implemented controller has a discretization step of only a few microseconds. The microprocessor, motors board, communication protocol determine the maximum sampling rate of the system.

In this case, a Raspberry Pi 4 was selected as the board to implement the controller. The communication protocol utilized is Serial Peripheral Interface (SPI), with custom messages designed to meet the specific requirements of the project and they have been implemented in order to match between all the parts of the communication line.

With the selected hardware, a sampling rate of $f_s = 1\,kHz$ was achieved. The cascading of boards, however, reduces the performance of the protocol. If required, the sampling frequency can be improved by switching to a Controller Area Network (CAN) protocol.

The motor's driver has an open source firmware that allows to customize all the features of the low level control loop. In this way the *decentralized controller*

described in Section 3 was set by sending the reference trajectory command.

The encoders are of the capacitive type and are mounted *on-axis* with the motor. The adopted encoders output $N = 4096$ pulses per turn, so the resolution is $r = \frac{360°}{4096} = 0.088°$.

4.4 Motor parameters

The connection between the gearbox and the mechanical load is assumed to be rigid (i.e. no elasticity is present in between the two elements); then it is reasonable to assume that the actual gear ratio coefficient corresponds to the anticipated value.

For each pair of motor and gearbox it is necessary to estimate both the static and dynamic friction. To conduct the experiment without external sensors at the end of the chain, we assume that the mechanical power input to the gearbox is fully transmitted to the output. Under this hypothesis equation 4.7 becomes $\tau_L = \frac{1}{k_g}\tau_M$.

Any knowledge of motor and gearbox efficiencies would not have brought any practical advantage, as it would not have been included in the controller. This is because, by way of example, estimating the efficiency of the individual motor proved to be non-constant depending on the load.

The experiments are performed with the gearboxes in an unloaded condition, and the current applied by the controller is measured. To achieve this a low-level feedback loop is implemented on the motor control board. A PI controller is used to track a reference velocity, and the measured phase current is considered to represent the torque exerted by the motor at the output.

4.4.1 Friction estimation

To set up the experiment, it is necessary to establish a model that accurately describes the friction. In general, for a motor, the torque equation can be seen at the motor side or at the load side (after the gearbox). On the load side, any externally applied torque can be modeled as τ_d.

$$\begin{cases} J_m \dfrac{d\omega_m}{dt} + b_m \omega_m = \tau_m - \tau'_L & (4.8) \\ J_L \dfrac{d\omega_L}{dt} + b_L \omega_L = \tau_L - \tau_d & (4.9) \end{cases}$$

The gears in use exhibit significant friction effects, thus the free parameter τ_d is used to model the *Coulomb* friction within the mechanical transmission. The static friction can be modelled as a constant resistant torque which opposes to

τ_m, ω_m	motor side torque and speed
τ_L, ω_L	load side torque and speed
τ'_L	the load torque seen at motor side. In particular $\tau_L = \frac{\eta_g}{k_g}\,\tau'_L$
τ_d	disturbance torque applied to the load inertia
J_m, b_m	inertia and viscous friction at motor side
J_L, b_L	load moment of inertia and viscous friction
f	static (or Coulomb) friction
k_g	gearbox ratio
i_q, K_i	quadrature current and motor torque constant

Table 4.2: Motor and gearbox: symbols definitions.

the load movement, namely:

$$\tau_d = f\ \mathrm{sgn}(\omega_L) \quad f > 0 \tag{4.10}$$

Since all measurable quantities are at *motor side*, it is more practical to adopt this convention moving forward. Equations 4.8, 4.9 can be combined into equation 4.11 through the application of equation 4.7.

$$J_{eq}\frac{d\omega_m}{dt} + b_{eq}\omega_m + f\ k_g\ \mathrm{sgn}(\omega_m) = \tau_m \tag{4.11}$$

where:

$$J_{eq} = J_m + J_L k_g^2 \qquad b_{eq} = b_m + b_L k_g^2$$

are the equivalent inertia and friction parameters seen at *motor side*.

For the purpose of the project the knowledge of the inertia is irrelevant; therefore, the experiment is designed to eliminate its influence. A possible way to measure the friction parameters is to impose a *constant velocity* and remove all external loads, in this way the inertia contribution can be neglected.

Given that it is not available a direct measure of the motor torque, but the latter is proportional to the current; equation 4.11 can be reformulated into 4.12. This scenario can be formulated as *least square* problem, whose unknowns are b_{eq} and f.

$$K_i\, i_q = b_{eq}\omega_m + f\ k_g\,\mathrm{sgn}(\omega_m) = \begin{bmatrix}\omega_m, & k_g\,\mathrm{sgn}(\omega_m)\end{bmatrix} = \begin{bmatrix} b_{eq} \\ f \end{bmatrix} \tag{4.12}$$

A regressor matrix $\mathbf{\Phi}$ and the corresponding torque vector $\mathbf{\Gamma}$ are constructed from measurements taken at various velocities. To ensure the identifiability of parameters within a linear regression framework, the regressor matrix must be

	joint type	$b_{eq}\,[Nms]$	$f\,[Nm]$
mn4004	axial	$2.42e^{-6}$	0.122
mn4006	radial	$3.96e^{-6}$	0.103
mn5006	axial	$6.45e^{-6}$	0.247

Table 4.3: Results of the estimated friction coefficients. As expected the static friction of the axial joints, which mount cycloidal gears, is higher than the radial one.

full column rank.

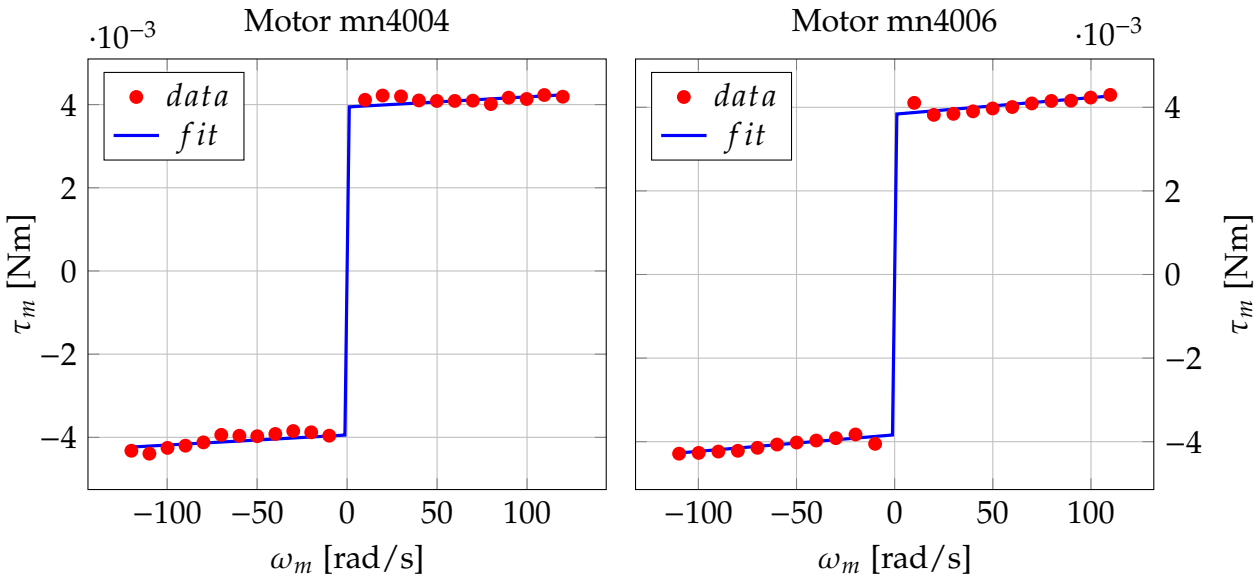
$$\mathbf{\Gamma} = \begin{bmatrix} \tau_{m_1} \\ \vdots \\ \tau_{m_N} \end{bmatrix} = \begin{bmatrix} \omega_{m_1},\ k_g\,\mathrm{sgn}(\omega_{m_1}) \\ \vdots \\ \omega_{m_N},\ k_g\,\mathrm{sgn}(\omega_{m_N}) \end{bmatrix} \begin{bmatrix} b_{eq} \\ f \end{bmatrix} = \mathbf{\Phi}\,\boldsymbol{\mu} \tag{4.13}$$

In this context, given the structure of the regressor matrix, measuring two distinct velocities would be sufficient to obtain a solution to the problem. However, to ensure a more reliable result, multiple samples at varying speeds are collected.

The solution is given by equation 4.14. In Figure 4.4, the behavior of the individual joints as a function of varying velocity can be observed. The selected linear model effectively represents the friction behavior. Indeed the fitted curves represent the estimated friction based on the collected data, namely $\hat{\tau}_m = \hat{b}\omega_m + \hat{f}\,k_g\,\mathrm{sgn}(\omega_m)$. These results, presented in table 4.3, should be considered valid for speed values within the measured range, as it is challenging to predict the behavior of friction at higher velocities.

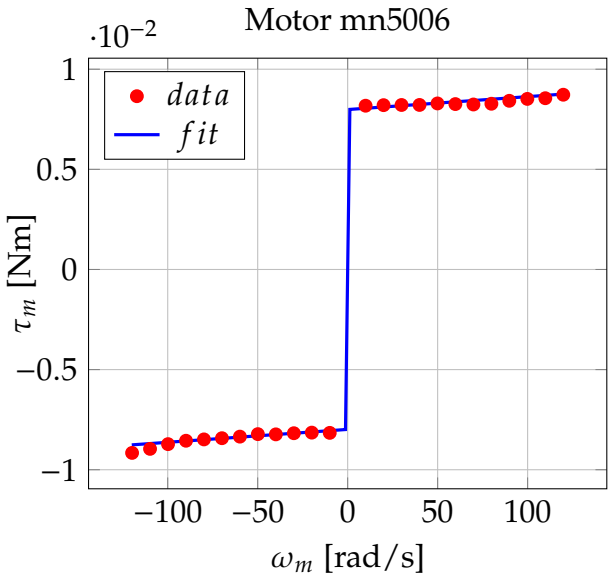
$$\begin{bmatrix} \hat{b}_{eq} \\ \hat{f} \end{bmatrix} = \hat{\boldsymbol{\mu}} = \left(\mathbf{\Phi}^T\,\mathbf{\Phi}\right)^{-1}\mathbf{\Phi}^T\mathbf{\Gamma} \tag{4.14}$$

From Table 4.3 it is possible to conclude that cycloidal gears, adopted for axial joints, present the higher Coulomb friction effort. Furthermore, the larger the motor, the greater the viscous friction. Specifically, the mn5006 motor exhibits the highest value.



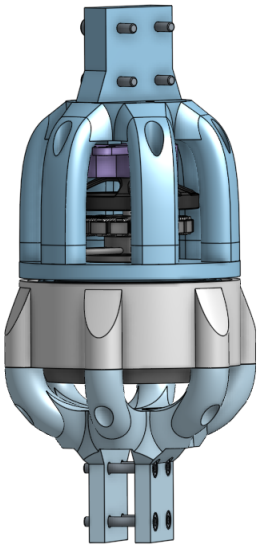
(a) Cycloidal gear for axial joint.

(b) Herringbone gear for radial joint.

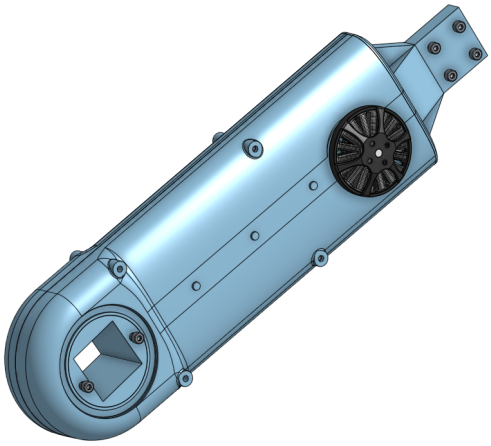


(c) Cycloidal gear for base axial joint.

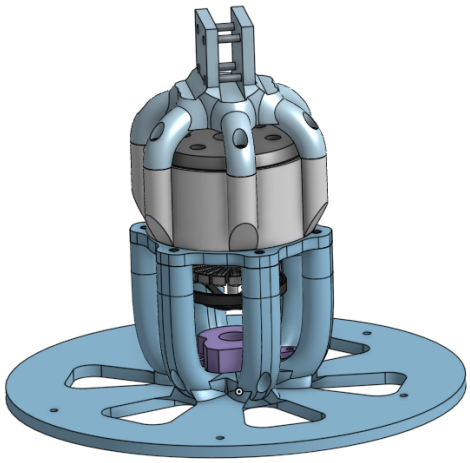
Figure 4.4: Measured data and least square fit of motor torque at the specified velocity. The fit is the best linear approximation of the overall friction effort. It represent: $\tau_m = \hat{b}\omega_m + \hat{f}k_g \operatorname{sgn}(\omega_m)$.



(a) Axial joint, motor mn4004. In the experimental arm this design is placed as third joint.



(b) Radial joint with motor mn4006. This joint structure is used for joints two and four.



(c) Axial joint, with of motor mn5006 used as base joint.

Figure 4.5: Types of joints used in the manipulator. The terminal connector permits the interchange of the order, which must be correspondingly updated in the simulation model.

5 Analysis

In the following sections, an overview of the simulation environment and the associated background algorithms is provided.

Subsequently, the results of the real-world experiments are presented and discussed, including the definition of a comparison metric to assess the accuracy of the estimates.

Finally, a simulation of the dynamic trajectory controller is presented.

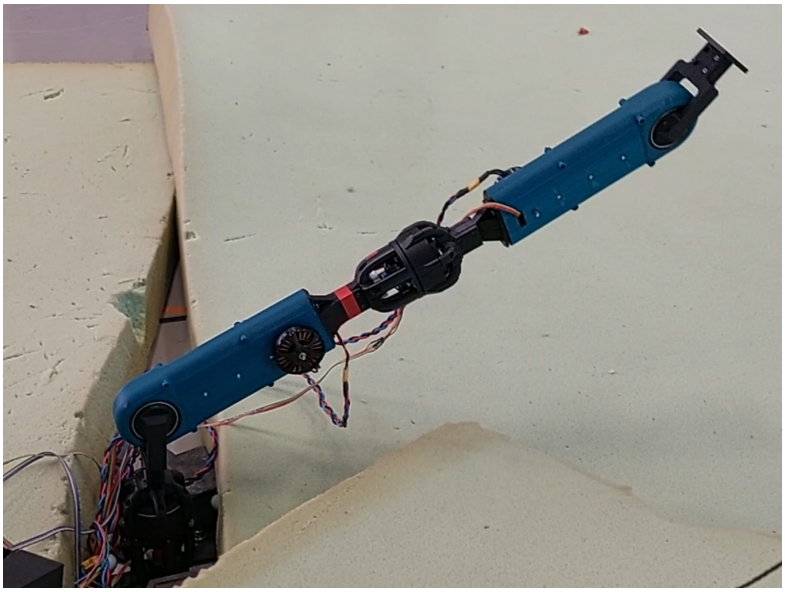


Figure 5.1: The physical manipulator positioned within the arena for safety testing.

5.1 Real time dynamic algorithm

As discussed in Section 2.4, the Lagrangian approach is not ideal for real-time applications. It provides a closed-form, analytical solution that is dependent on n, the number of generalized variables. An alternative derivation of the

dynamics is achieved through the Newton-Euler (NE) formalism. This method provides a recursive approach, where the motion of each link is coupled with that of the others via kinematic relationships for velocities and accelerations.

The NE formalism is based on a balance of all the forces acting on the generic link of the manipulator. This balance can be computed recursively using the Recursive Newton-Euler Algorithm (RNEA), which consists of two primary stages: a *forward recursion* and a *backward recursion*. In the forward recursion, **velocities** and **accelerations** propagate sequentially from the first rigid body to the last, determining each link's kinematic quantities based on those of its predecessor in the chain. Conversely, in the backward recursion, **forces** and **torques** are propagated back from the last rigid body, typically the robot's end effector, to the first link.

In this project, a recursive formulation has been implemented to meet real-time performance requirements. To obtain good performances in computing both the dynamics and their derivatives, the *Pinocchio* library [2] has been utilized. *Pinocchio* is one of the most efficient libraries for computing the dynamics of articulated bodies moreover has the advantage to be open-source.

The library allows also the computation of the regressor matrix, as in equation 3.1.

5.2 Simulation environment

In order to test the controllers before the application in the real manipulator it is useful to use a simulator. In theory, a tool that solves non-linear differential equations would be sufficient to describe the motion of the arm. However there are some drawbacks with this types of simulators.

- Typically, the visual interface only allows the observation quantity plots rather than the actual motion of the bodies, which can make interpreting the system's behavior more challenging.
- These simulators highly dependent on the specified model equations, meaning they do not simulate physics but rather evolves the system from initial conditions and inputs. Conversely, a physics engine is a computational tool that approximates the simulation of physical systems, incorporating collision detection and rigid body dynamics.
- It is hard to extend in projects with multi-bodies.

Based on these considerations, it has been chosen to implement the manipulator in *Gazebo*, an open-source robotics simulator. Gazebo supports multiple high-performance physics engines, including ODE, Bullet, and DART.

An advantage of using Gazebo, lies in the integration with ROS and other customable middleware. This simplifies the change between the simulation and

the real system, as there is no need to re-write the functions that send messages to the various components.

In Gazebo, a model is defined in a Unified Robotic Description Format (URDF) file, where the body is composed of *links*: elements that describe physical properties of each individual body within the model. Each link is specified with the following three fields which will influence the simulation.

- **Collision**. A collision element defines the geometry used by the simulation engine for collision detection and contact force propagation. These shapes should remain simple to reduce computational complexity.
- **Visual**. As the name suggests, this field specifies the visualized meshes. Mesh files can be imported from CAD software to enhance model realism, though this field does not influence the simulation itself.
- **Inertial**. The inertial element describes the dynamic properties of the link, such as mass and inertia matrix.

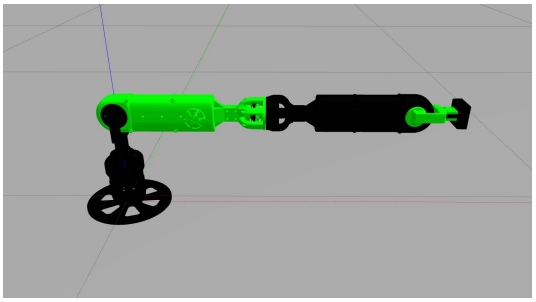
Then *joint* elements connects two links and characterize the relationship between them, along with other features. An example of the URDF file is provided in Appendix Section A.4.

In the inertia field, parameters such as mass, inertia, and the *pose* of the center of mass of each link can be defined. Moments of inertia are directly proportional to mass; however, they vary non-linearly with size. Furthermore, constraints on the relative values of the principal moments make estimating moments of inertia considerably more challenging than determining mass or center of mass location. This complexity justifies using specialized software tools for calculating moments of inertia, which is why this information is typically derived from the CAD model of the body, once the material properties are specified. These parameters are, however, estimates; thus, the model will be refined using parameters obtained from the optimization process.

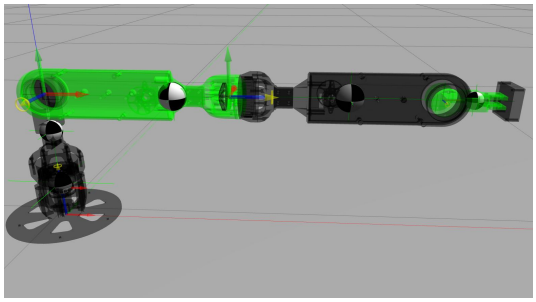
In Figure 5.2c, boxes surrounding each link are shown, with each boxs center aligned to the specified center of mass of its respective link. The sizes and orientations of these boxes represent unit-mass boxes designed to exhibit the **same inertial behavior** as their corresponding links. This is useful for debugging the inertial parameters.

5.3 Parameter estimation results

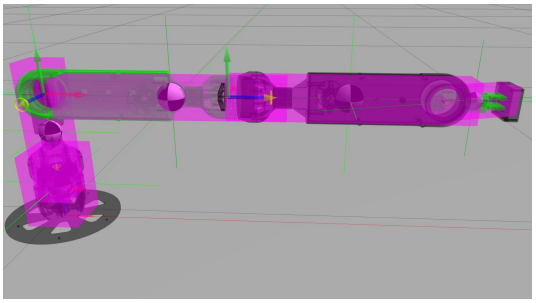
As outlined in Section 3.4, the trajectory is generated randomly. It is subsequently verified to satisfy workspace coverage and velocity variability conditions within the Gazebo simulator. Although the dynamic model is not required for arm control, the CAD parameters are included in the models URDF file for simulation purposes.



(a) The arm is in *neutral* position, where $q_i = 0$ for $i = 1, \ldots, 4$. The colors indicate individual links.



(b) In this representation, the centers of mass are marked by black and white spheres within the links, with the sphere size proportional to the mass of each link. The joint frames are also highlighted.



(c) The inertia of each link is represented by a box centered at the link's center of mass (CoM), approximately fitting within the links geometry.

Figure 5.2: Manipulator model in Gazebo simulator.

The robot performs the identification trajectory for $20\,s$. At a sampling frequency of $1\,kHz$, with a total of $M = 20,000$ samples for position, velocity and torque for each joint. With this setup, the regressor matrix $\bar{\mathbf{\Phi}}$ has dimensions $(80,000 \times 48)$, corresponding to the number of parameters, $np = 48$ (defined in 3.7).

Since computing the singular value decomposition (SVD) of a large rectangular matrix, such as $\bar{\mathbf{\Phi}}$ is computationally inefficient, the relationship between the singular values of $\bar{\mathbf{\Phi}}$ and $\bar{\mathbf{\Phi}}^T\bar{\mathbf{\Phi}}$ has been exploited, as outlined in equation 5.1.

$$\begin{aligned} \sigma\left(\bar{\mathbf{\Phi}}^T\bar{\mathbf{\Phi}}\right) &= \sigma_1^2, \ldots, \sigma_r^2 \\ \sigma\left(\bar{\mathbf{\Phi}}\right) &= \sigma_1, \ldots, \sigma_r \end{aligned} \tag{5.1}$$

where $\sigma(\boldsymbol{A})$ represent the set of singular values of $\boldsymbol{A}$ and $r = rank(\bar{\mathbf{\Phi}})$. In the experiment, the regression matrix has rank $r = 30$. The maximum and minimum singular values of $\bar{\mathbf{\Phi}}^T\bar{\mathbf{\Phi}}$ are preserved after the application of the square root operation, as consequence of the monotonicity property of the square root function. Consequently, the condition number can be computed as in equation 5.2.

For the identification trajectory, the computed condition number is $c = 47$. Therefore the regressor matrix can be considered *well-conditioned* (Gautier and Khalil, 1992).

$$c = \frac{\sigma_{max}}{\sigma_{min}} \tag{5.2}$$

5.3.1 Identification trajectory

As the excitation trajectory, the one depicted in Figure 5.3 has been selected. For each link, the desired and resulting trajectories are shown. Additionally, Section A.2 of the Appendix presents the executed position trajectory and the corresponding differentiated acceleration. As observed in Figure 5.3, the first joint, and particularly the second, exhibit a larger tracking error, especially during motion reversals. This discrepancy results from the arm's weight, which, in rapid direction changes, pushes the respective motors to their current limits, preventing a smooth behavior. Additionally, the connectors between consecutive links, subject to tolerance variations from 3D printing, introduce a slight slack. Nonetheless, the measured velocities remain essential in calculating the regressor, though these factors also impact the exerted torque. Hence, the measured velocities are utilized in the computation of the regressor instead of the theoretical values, despite these effects also being reflected in the exerted torque.

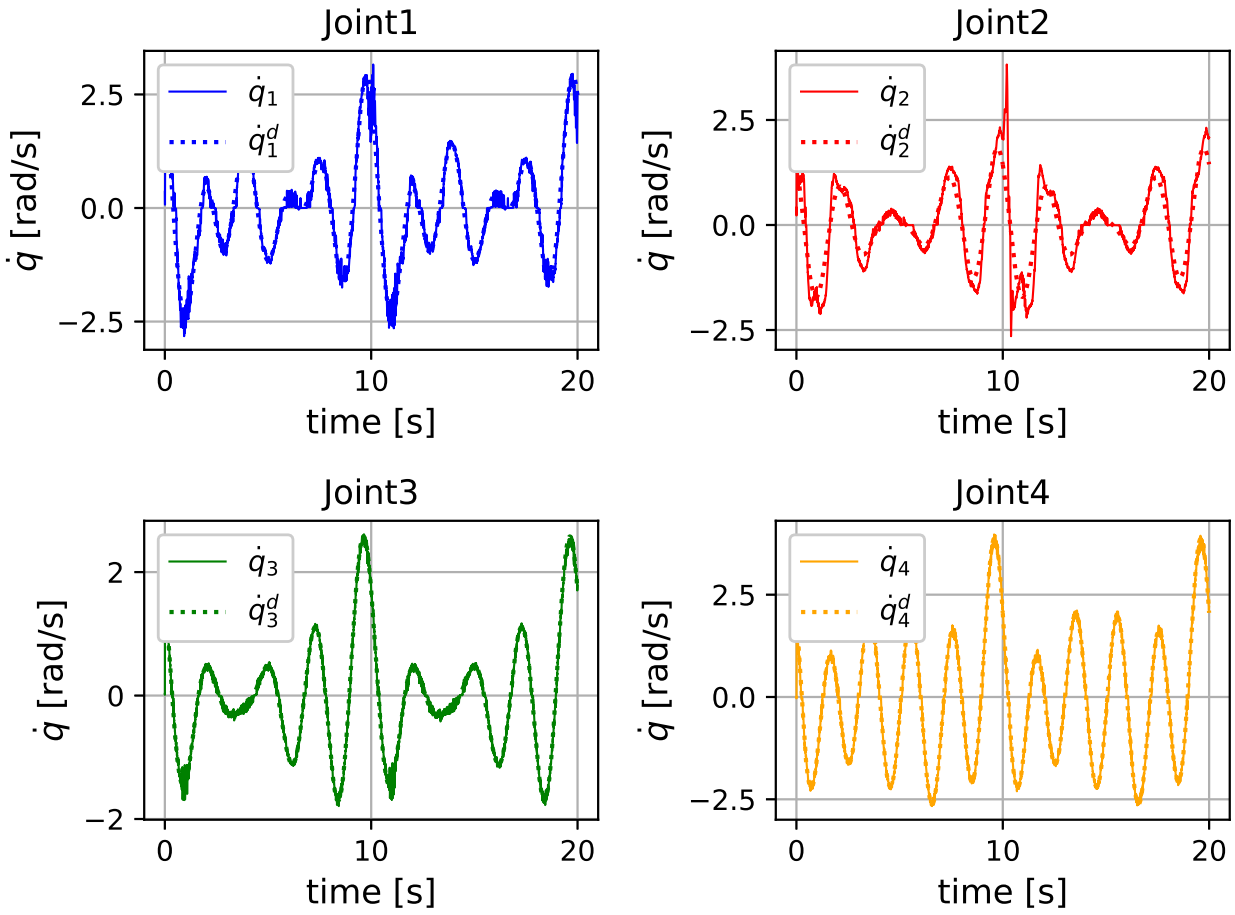


Figure 5.3: Actual and desired joint velocities for the identification trajectory are shown. Due to their greater mass, the first two joints track the trajectory with lower accuracy compared to the final two joints.

5.3.2 Estimation assessment

To validate the estimated parameters, a validation trajectory is generated using the same techniques applied to the identification trajectory, though with different initial conditions. Consequently, the validation trajectory maintains the same harmonic components but exhibits different amplitudes. Also in the validation case the duration of the experiment is of $20\,s$.

To evaluate the accuracy of the torque predictions, a performance metric is introduced. Based on the approach outlined in [5, eq.(84)], the torque prediction error percentage is defined as follows:

$$\epsilon_\tau = 100 \cdot \frac{\|\bar{\tau} - \bar{\mathbf{\Phi}}\,\hat{\theta}\|}{\|\bar{\tau}\|} \tag{5.3}$$

where $\hat{\theta}$ is the estimated parameter vector.

	$\hat{\theta}_{CAD}$	$\hat{\theta}_{SLSQP}$	$\hat{\theta}_{SA}$
Identification	41.0%	35.7%	25.8%
Validation	45.4%	40.3%	34.2%

Table 5.1: Relative error percentages, calculated using equation 5.3, of the predicted torque for identification and validation trajectories.

In Section A.3 are reported the parameters obtained in the estimation via CAD software and via the optimization processes.

Figure 5.4 shows plots of the measured torque and the predicted one, using $\hat{\theta}_{SA}$ estimate. It can be observed that for the first two joints predicted torque tracks the measured ones well. In contrast, for the last two joints, a noticeable degradation in tracking performance is present. This discrepancy is attributed to the smaller torque values, as the inertial parameters for these links are relatively small (or at least smaller in comparison to the parameters of the first links).

On the other side Figure 5.5 presents the behavior of the estimated model in predicting a different trajectory torque. The observed behavior is consistent with the results discussed for the identification trajectory, although the first joint exhibits increased tracking noise, likely due to higher noise levels in the measurements. This observation is further supported by the values of ϵ_τ presented in Table 5.1.

The error of the torque predictions is presented in Table 5.1. It can be observed that, for both the identification and validation trajectories, the error is relatively high. This can primarily be attributed to the physical and assembly characteristics of the manipulator. Factors such as the assembly process, 3D printing tolerances, the use of plastic materials, and mounting screws introduce

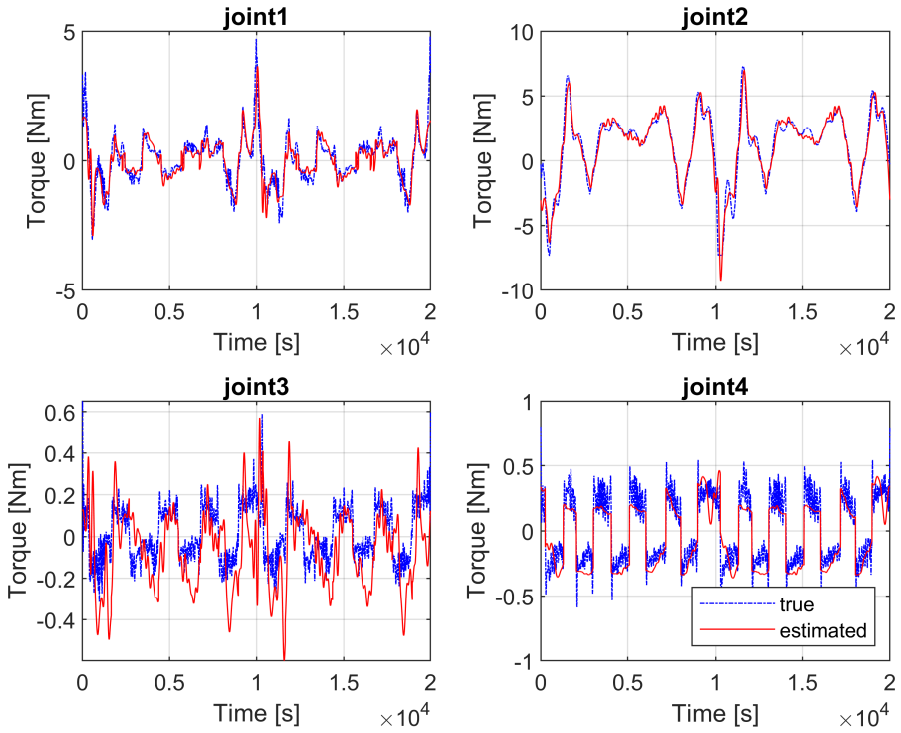


Figure 5.4: *Identification* trajectory. Measured torque vs predicted torque of the SA solution: $\hat{\theta}_{SA}$.

flexibilities in the connections between links, resulting in deformations that are not accounted for by the dynamic model.

Furthermore, due to the substantial weight of the links, the motors at joints one and two, during certain segments of the trajectory, are required to generate torque near their current limit. This challenges the assumptions made in the current-to-torque approximation, leading to non-linearities in the current-to-torque relationship.

5.4 Dynamic controller simulation

Due to time constraints the dynamic controllers were tested only in the simulation environment of gazebo. In particular after the estimation phase the new link parameter vector was included in the URDF model of the manipulator.

The estimated dynamic parameters were incorporated into the URDF model of the simulation file, while the CAD-based estimation was used as input for the controller model. In other words, the controller utilizes the theoretical parameters, whereas the simulation operates on the estimated model.

To evaluate the system, several test trajectories were generated using the same procedure as for the exciting trajectory. The torque commands, derived from equation 2.21, were then applied to the control boards. The resulting Cartesian tracking error is shown in Figure 5.6, where the motion of the end-

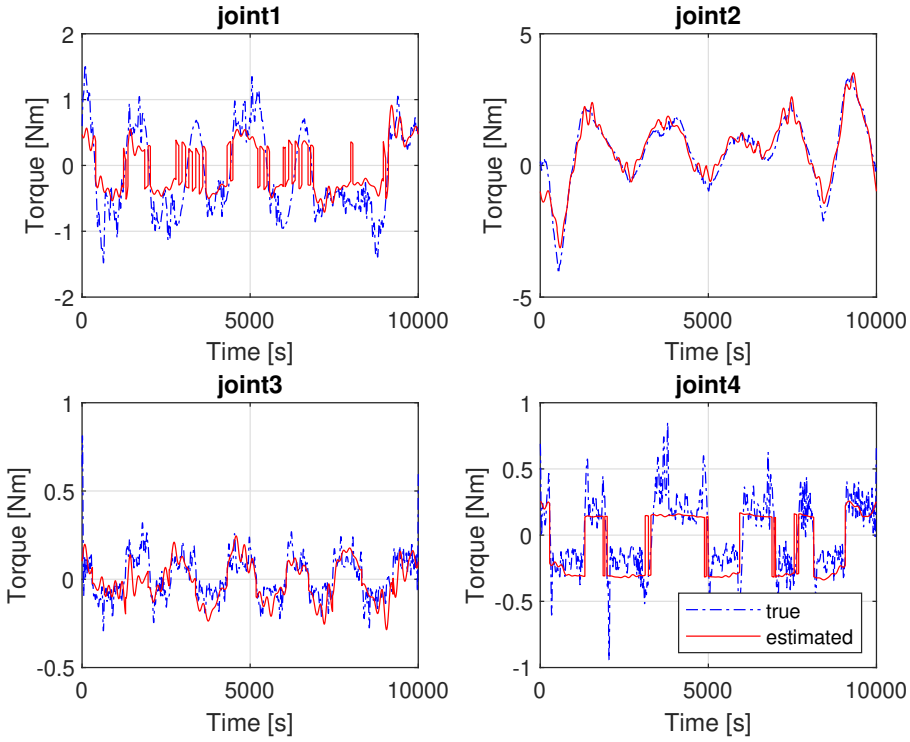


Figure 5.5: *Validation* trajectory. Measured torque and predicted torque with the SA solution: $\hat{\theta}_{SA}$.

effector is illustrated. This phase aimed to assess the performance of the dynamic controllers and validate the system parameters.

It was observed that, for certain trajectories, the error in the simulation does not asymptotically converge to zero. Therefore, for experiments involving the real manipulator, robust control techniques must be implemented to mitigate the effects of unmodeled dynamics.

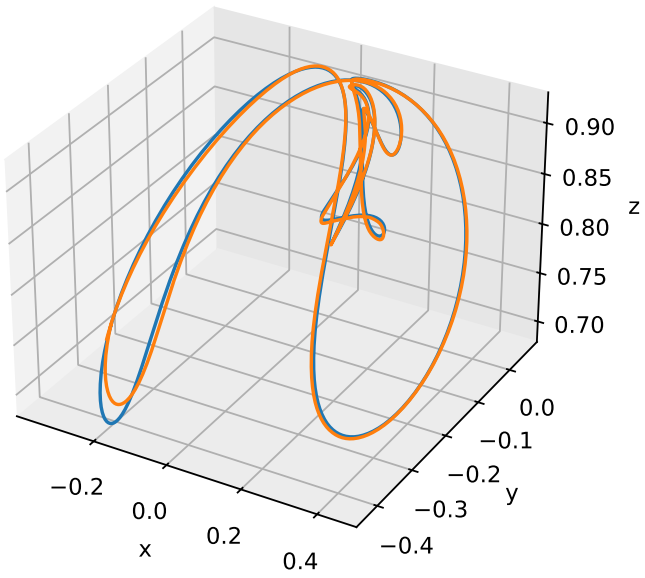


Figure 5.6: Cartesian trajectory controlled via torque commands. The blue trajectory represents the desired end-effector position assuming perfect tracking of the joint trajectory, while the orange trajectory depicts the position executed by the simulation.

6

Conclusions and Future Works

In this thesis, we presented a manipulator designed for aerial manipulation tasks. The arm was developed at LAAS-CNRS, with the specific objective of extending its workspace beyond the area occupied by the drones propellers.

Brushless motors were selected to actuate the joints, equipped with gearboxes at their output. The system includes brushless motors, encoders, and control boards. Since full access to motor information is available, it is possible to focus controller development on optimizing the energy used by the manipulator, a key point for battery-powered applications.

Low-level feedback control loops have been designed for the joints, providing high performance in terms of execution speed and tracking accuracy. Different control structures have been implemented, including decentralized control for parameter estimation and centralized impedance loops leveraging the system dynamics. In the estimation of dynamic parameters feasibility constraints were considered, where the feasible set can be further restricted to accommodate additional constraints. However, the adopted optimization method was computationally slow, suggesting that alternative approaches, such as exploiting positive semidefinite constraints, may enhance the estimation process. Despite the optimization process, the prediction error remained high. To enhance the estimation accuracy, a precise measurement of the motor torque constant at high currents is necessary. Additionally, the use of 3D-printed plastic for the gears proved suboptimal, as it introduced backlash and degraded over time. Unmodeled effects significantly influenced the estimation; however, the performance improved for slower trajectories.

With the models of the arm and the drone independently identified, it becomes possible to analyze the coupling effects to develop a comprehensive full-body model. The interaction forces transmitted along the arm can be introduced into the system and subsequently compensated for or used for control purposes.

A

Appendix

A.1 Brushless motor - torque constant

There exists a fundamental relationship between an electric BLDC motor's velocity constant (K_v), armature current (i_q) and torque (τ). The relation in object is described in equation 4.3 and reported below for convenience.

$$K_i = \frac{3}{2}\frac{1}{\sqrt{3}}\frac{60}{2\pi}\frac{1}{K_v} \qquad \tau = K_i\, i_q$$

Where K_v is in $\frac{rpm}{V}$, $i_q\,[A]$ and $\tau\,[Nm]$. It is worth notice that the voltage in K_v refers to a line-to-line voltage, while in the computation below all the voltages refer to phase voltage. In a three phase system the relation between line-to-line and phase quantities is: $V_{LL} = \sqrt{3}\,V_P$.

Under the following hypothesis it is possible to state that, with the above-mentioned K_i, the torque is linear to the quadrature current.

- The low level controller (FOC) maintains $i_d \approx 0$,
- there is a low voltage drop in the coils resistance,
- given as parameters of the motor L_s, the equivalent or synchronous inductance, and Λ_m the flux linkage due to the permanent magnets, then $L_s\, i_q << \Lambda_m$. In practice it is reasonable to assume since the synchronous inductance is usually very small.

It is worth notice that bigger the current (both in d or q axes) bigger is the voltage drop in the resistance and the linear relation is lacking.

From electric equations of SPM motors, for voltage smaller than the nominal one, it holds:

$$v^2 = (Ri_d - \omega_m L_s i_q)^2 + (Ri_q + \omega_m\,(\Lambda_m + L_s i_d))^2$$

Where ω_m is the **electrical** speed, and v is the norm of the voltage in the synchronous frame. Under the hypothesis above it is possible to re-write this equation as follows:

$$\Lambda_m^2 = \frac{v^2}{\omega_m^2} \quad \Longrightarrow \quad \omega_m = \frac{1}{\Lambda_m} v \tag{A.1}$$

From the motor's power equation 4.1, after accounting for power losses, the remaining mechanical power can be converted into torque. In particular $\tau = \frac{3}{2} p \Lambda_m i_q$ (where p is the number of pole pairs). This relation is linear but it does not depend explicitly on K_v.

Given the relation between mechanical (ω) and electrical (ω_m) speed: $\omega = p\,\omega_m$, then the above equation can be rewritten as $\omega = \frac{1}{\Lambda_m p} v = K_v' v$.

As mentioned K_v refers to line-to-line voltage, so it must be converted to phase voltage. Also the speed measure is not standard and a conversion in $\frac{rad}{s}$ is required.

$$K_v' = \frac{2\pi}{60}\sqrt{3}\,K_v \qquad K_v \left[\frac{rpm}{V_{LL}}\right] \quad K_v' \left[\frac{rad}{s\,V_P}\right]$$

At this point equation 4.3 is straightforward

$$\tau = \frac{3}{2} p \Lambda_m i_q = \frac{3}{2}\frac{1}{K_v'} i_q = \frac{3}{2}\frac{1}{\sqrt{3}}\frac{60}{2\pi}\frac{1}{K_v} i_q = K_i\, i_q \tag{A.2}$$

In other words: the factor $\frac{60}{2\pi}$ is the conversion term $rpm \rightarrow \frac{rad}{s}$, $\frac{1}{\sqrt{3}}$ is for converting the line-to-line voltage to phase voltage and $\frac{3}{2}$ is the scaling factor of three phases systems.

A.2 Identification trajectory

Figure A.1 shows both the commanded and measured position trajectory in joint space used to compute the regressor matrix for the identification process. As discussed, and clearly coherently, with the velocity plot of figure 5.3 also in this case the first two joints present the lowest precision in tracking the desired trajectory.

In Figure A.2, the acceleration is shown. The acceleration is derived from the filtered joint velocity to mitigate the dominance of noise. The numerical derivative is computed offline using a second-order approximation of the continuous-time derivative.

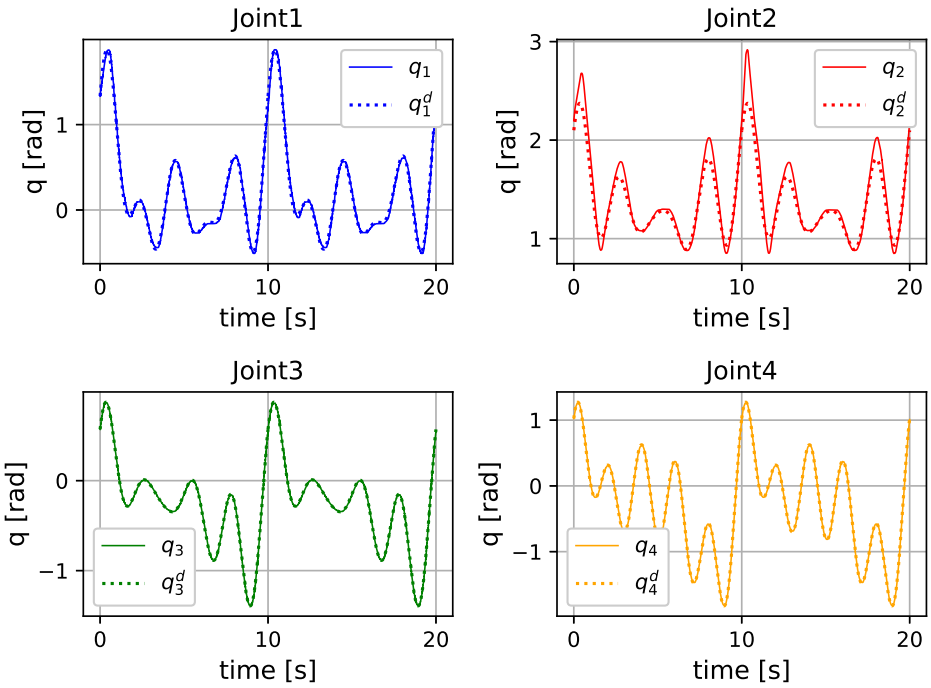


Figure A.1: Actual and desired joint position for the identification trajectory.

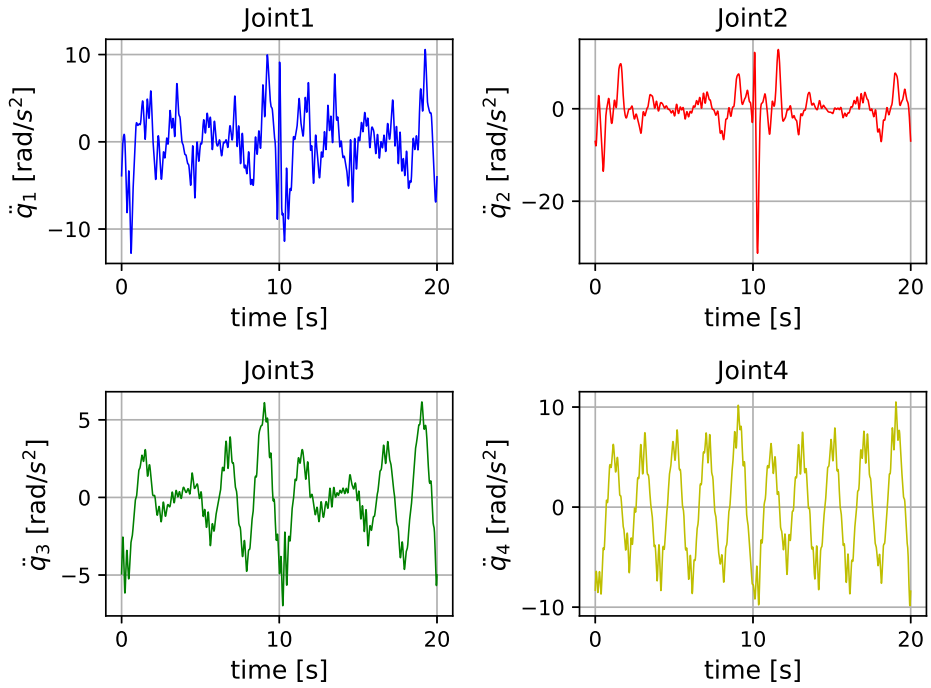


Figure A.2: Differentiation of the filtered version of the joint velocity of figure 5.3, namely acceleration computed offline.

A.3 Link parameter vectors

Table A.1 presents $np = 48$ dynamic parameters of the manipulator. In particular, the parameter vectors from the CAD framework and the estimation process are shown.

	$\hat{\theta}_{CAD}$	$\hat{\theta}_{SLSQP}$	$\hat{\theta}_{SA}$
m1	2.76e-1	2.76e-1	2.27e-1
m*Cx1	0.00e+2	1.50e-4	-3.37e-3
m*Cy1	0.00e+2	-8.11e-5	2.30e-3
m*Cz1	2.87e-2	2.87e-2	5.64e-2
Ixx1	4.33e-3	4.35e-3	2.73e-2
Ixy1	0.00e+2	1.55e-4	-1.84e-5
Iyy1	4.33e-3	4.29e-3	2.73e-2
Ixz1	0.00e+2	-2.86e-5	3.79e-4
Iyz1	0.00e+2	2.34e-4	8.68e-4
Izz1	1.53e-4	2.01e-4	1.00e-4
m2	5.54e-1	5.54e-1	4.00e-1
m*Cx2	1.16e-1	1.16e-1	2.00e-1
m*Cy2	0.00e+2	1.77e-4	-5.00e-2
m*Cz2	1.99e-3	2.23e-3	5.00e-2
Ixx2	3.16e-4	2.75e-4	1.00e-4
Ixy2	0.00e+2	-1.46e-4	-4.59e-3
Iyy2	3.07e-2	3.07e-2	1.41e-1
Ixz2	-3.50e-4	-2.61e-4	-2.43e-2
Iyz2	0.00e+2	-2.21e-4	8.03e-2
Izz2	3.08e-2	3.06e-2	1.00e-4
m3	5.28e-1	5.28e-1	8.00e-1
m*Cx3	2.06e-3	2.08e-3	-5.79e-3
m*Cy3	0.00e+2	9.94e-5	-2.43e-3
m*Cz3	9.98e-2	9.98e-2	2.13e-2
Ixx3	2.54e-2	2.55e-2	1.57e-1
Ixy3	0.00e+2	1.88e-4	5.46e-2
Iyy3	2.54e-2	2.53e-2	1.16e-1
Ixz3	-3.44e-4	-2.45e-4	2.83e-2
Iyz3	0.00e+2	7.70e-5	5.98e-2
Izz3	3.25e-4	3.80e-4	2.18e-2
m4	1.29e-1	1.29e-1	5.00e-2
m*Cx4	6.13e-3	6.09e-3	0.00e+2
m*Cy4	0.00e+2	-6.80e-5	8.81e-3
m*Cz4	0.00e+2	-5.06e-5	1.00e-2
Ixx4	5.10e-5	3.77e-5	1.00e-5
Ixy4	0.00e+2	3.22e-5	-6.06e-3
Iyy4	4.90e-4	5.96e-4	7.08e-3
Ixz4	4.49e-5	6.50e-5	1.04e-3
Iyz4	0.00e+2	1.93e-4	7.00e-3
Izz4	4.83e-4	5.34e-4	1.00e-5
fv1	5.30e-6	1.30e-4	3.57e-1
fv2	4.20e-6	0.00e+2	3.23e-1
fv3	8.00e-6	3.09e-5	1.30e-1
fv4	4.20e-6	0.00e+2	5.02e-3
fs1	2.00e-1	2.00e-1	3.00e-1
fs2	1.10e-1	1.10e-1	4.81e-2
fs3	1.50e-1	1.50e-1	4.64e-2
fs4	1.10e-1	1.10e-1	2.25e-1

Table A.1: Initial parameter estimates and final estimated parameters of the manipulator.

A.4 URDF MODEL OF THE MANIPULATOR

The code snippet below demonstrates an example of defining a link and its subsequent joint.

As described in Section 5.2, a *link* object can be declared with three main fields: inertial, visual, and collision. Each of these fields requires the specification of the object's *pose*, defined in terms of translation (xyz) and rotation using an Euler angle triplet rpy (Roll-Pitch-Yaw), relative to the preceding joint in the chain's reference frame. The collision object should be defined using a simple geometric shape to avoid overloading the simulation. In the example provided, a cylindrical shape has been specified.

A *joint* object, on the other hand, connects two links, referred to as the parent and child links. It includes a *type* attribute and additional parameters characterizing the joint, such as friction, limits, and the axis of rotation. All units of measurement are expressed in standard format, specifically meters and radians.

```
<link name="link1">
    <inertial>
        <mass value="0.276"/>
        <origin rpy="0 0 0" xyz="0.0 0.0 0.104"/>
        <inertia ixx="0.001342508" ixy="0.0" ixz="0.0" iyy="0.001342508" iyz="0.0"
izz="0.000152961"/>
    </inertial>
    <visual name="link1_visual">
        <origin xyz="0 0 0.0" rpy="0 0 0"/>
        <geometry>
            <mesh filename="/meshes/link1.stl" scale="1 1 1"/>
        </geometry>
    </visual>
    <collision name="link1_geom">
        <origin xyz="0 0 0.07" rpy="0 0 0"/>
        <geometry>
            <cylinder length="0.09" radius="0.04"/>
        </geometry>
    </collision>
</link>
<joint name='joint2' type='revolute'>
    <parent link="link1"/>
    <child link="link2"/>
    <origin rpy="1.5707 0.0 0.0" xyz="0.0 0.0 0.167"/>
    <axis xyz="0 0 1"/>
    <limit effort="50.0" lower="-6.283" upper="6.283" velocity="5.0"/>
    <dynamics damping="3.96e-6" friction="0.103"/>
</joint>
```

Code A.1: Example of Link and Joint definition in a URDF File.

References

[1] A. Oliva C. Gaz M. Cognetti, P. Robuffo Giordano, and A. De Luca. "Dynamic Identification of the Franka Emika Panda Robot With Retrieval of Feasible Parameters Using Penalty-Based Optimization". In: *IEEE RA-L* (2019).

[2] J. Carpentier et al. "The Pinocchio C++ library – A fast and flexible implementation of rigid body dynamics algorithms and their analytical derivatives". In: *International Symposium on System Integration (SII)*. 2019.

[3] W. Verdonck J. Swevers and J. De Schutter. "Dynamic model identification for industrial robots". In: *IEEE Control Syst. Mag., vol. 27, no. 5* (Oct. 2007), pp. 58–71.

[4] Bruno Siciliano et al. *Robotics: Modelling, Planning and Control*. 1st. Springer Publishing Company, Incorporated, 2008. ISBN: 1846286417.

[5] C. Sousa and R. Cortesão. "Physical feasibility of robot base inertial parameter identification: A linear matrix inequality approach". In: *Robot. Res., vol. 33, no. 6* (2014), pp. 931–944.

Acknowledgments

This research was supported by LAAS-CNRS and the University of Padova.

I would like to express my deepest gratitude to my supervisors, Dr. Marco Cognetti and Dr. Gianluca Corsini, for their guidance, patience, and support throughout this project.

I would also like to express my gratitude to Professor Angelo Cenedese for providing me with this opportunity and for his support throughout the project.

I would like to extend my thanks to my colleagues at the Lab, for their collaboration and support. The discussions, ideas, and shared experiences made this journey both productive and enjoyable.

I am deeply thankful to my family for their unwavering support and encouragement throughout my academic journey.

To my friends, thank you for providing a constant source of motivation and distraction when needed.