UNIVERSITÀ
DEGLI STUDI
DI BRESCIA

DOTTORATO DI RICERCA IN INGEGNERIA MECCANICA E INDUSTRIALE

ING-IND/13 MECCANICA APPLICATA ALLE MACCHINE

XXXIV CICLO

INNOVATIVE EXPERIMENTAL PROCEDURES AND DYNAMICS MODELS FOR INDUSTRIAL ROBOT MANIPULATORS

ROBERTO PAGANI

PROF. GIOVANNI LEGNANI

PROF. ANTONIO VISIOLI

PROF. RICCARDO ADAMINI

PROF.SSA LAURA ELEONORA DEPERO

RINGRAZIAMENTI

Desidero ringraziare in particolar modo il Professor Giovanni Legnani per avermi affidato questo interessante argomento di ricerca. Ringrazio inoltre i professori Antonio Visioli, Riccardo Adamini ed Alberto Borboni per tutto il supporto fornitomi in questo periodo.

Grazie a Cristina, senza dubbio la persona con la quale ho condiviso la maggior parte dei risultati ottenuti durante questi 3 anni di dottorato.

Un ringraziamento speciale va ad Alberto, il quale è stato uno dei primi a credere in me, senza di lui probabilmente questo percorso non sarebbe nemmeno iniziato. Ringrazio inoltre Luca, Domenico, Manuel, Marco e Simone per essere sempre stati così disponibili ogni volta che mi sono trovato in difficoltà.

Ringrazio tutti i miei compagni di Dottorato: Marco, Stefano, Federica, Enrique, Luca e Davide con cui ho condiviso momenti indimenticabili, affrontando le difficoltà e condividendo insieme meritate soddisfazioni. Un profondo ringraziamento va a tutti i miei amici, compagni di vita, che mi sono sempre stati vicini.

Ringrazio Francesca, la quale mi ha supportato e sopportato per tutti questi anni. Un ultimo ringraziamento va alla mia famiglia, per aver sempre creduto in me.

Ma soprattutto grazie mamma, grazie papà.

CONTENTS

LIST OF FIGURES

LIST OF TABLES

ACRONYMS

BP Base dynamics Parameters set

CAD Computer Aided Design

CNC Computer Numerically Controlled

EP Essential dynamics Parameters set

GA Genetic Algorithm

IR Industrial Robot

IV Instrumental Variable

KF Kalmann Filtering

OLS Ordinary Least Square

RMS Root Mean Square

SVD Singular Value Decomposition

TCP Tool Center Point

WLS Weighted Least Square

ITALIAN ABSTRACT

L'utilizzo di manipolatori industriali viene ormai sempre più considerato una valida alternativa all'impiego di alcune macchine a controllo numerico. Uno dei principali vantaggi dati dall'impiego di un robot rispetto ad una macchina CNC, è la possibilità di sostenere ridotti costi d'acquisto pur mantenendo elevati volumi di produzione e garantendo maggiore flessibilità ed adattamento in futuro. Nonostante l'evidente diffusione di questi manipolatori industriali, le loro prestazioni presentano ancora delle lacune. I problemi da affrontare in questo campo sono infatti molteplici e spaziano dalla compensazione di inaccuratezze cinematiche (calibrazione cinamatica), al miglioramento delle performance dinamiche (calibrazione dinamica), fino a sfociare nella valutazione dell'influenza del processo tecnologico sulla struttura del robot, ad esempio la possibilità di riscontrare danni causati da agenti esterni, come in caso di elevate temperature. All'interno di tale contesto, questa tesi si prefigge l'obiettivo di introdurre e investigare nuove tecniche per la calibrazione di modelli dinamici.

Partendo dalla descrizione dello stato dell'arte in ambito della modellazione dinamica, questo lavoro si è preposto di sviluppare un modello dinamico a corpi rigidi, con particolare attenzione agli effetti termici e la relativa variazione del fenomeno d'attrito. La scelta di attribuire particolare attenzione alla modellizzazione dell'attrito è motivata dal fatto che la compensazione dell'attrito è essenziale al fine di ottenere miglioramenti significativi nelle prestazioni dei sistemi meccatronici anche migliorando gli algotirmi di controllo. Siccome i modelli di attrito più comuni non tengono in considerazione il riscaldamento della macchina durante il funzionamento, per migliorare la stima dell'attrito durante le ore di lavoro delle macchine sono stati sviluppati dei modelli termici di attrito.

Nella tesi viene introdotta un'innovativa procedura per la calibrazione di modelli a corpi rigidi che passa per la definizione di traiettorie di identificazione/eccitazione ottime. Gli algoritmi di calibrazione e i relativi set di parametri dinamici stimati sperimentalmente sono inoltre stati validati attraverso prove pratiche su robot industriali.

Le procedure che sono state sviluppate in questa tesi utilizzano soltanto sensori e controllori che sono già presenti sulla macchina per il suo normale funzionamento. Nessun sensore è stato montato per ottenere migliorie nella stima dei parametri: questa caratteristica potrebbe far sì che tali soluzioni vengano appli-

cate a livello industriale.

Mentre esperimenti di questo tipo sono generalmente effettuati su un singolo robot, in questa tesi sono stati analizzati dati provenienti da due robot industriali (due repliche dello stesso modello) per verificare la ripetibilità dei risultati. Questo è un altro importante aspetto per garantire la robustezza delle procedure di identificazione del modello in caso di implementazione in applicazioni industriali. I risultati del confronto possono essere utilizzati per garantire miglioramenti a livello di produzione e nella manutenzione delle macchine anche implementando meodologie di diagnostica e manutenzione predittiva.

In conclusione, la tesi affronta l'analisi della modellazione e della calibrazione dinamica di manipolatori industriali. Vengono inoltre introdotte nuove ed efficaci strategie per la stima dei parametri del modello dinamico. I modelli calibrati sono quindi in grado di fornire elevate prestazioni in termini di accuratezza nella previsione delle coppie dei motori, tenendo inoltre in considerazione la variazione dell'attrito in base alla temperatura.

L'efficacia di tali tecniche di calibrazione è stata provata su due robot industriali con l'obiettivo di verificarne la ripetibilità e la robustezza.

ENGLISH ABSTRACT

Nowadays, industrial robot manipulators (IRs) are considered an immediate feasible alternative to CNC machines. One of the main advantages is the possibility to reduce costs while maintaining high production volumes. Despite the evident diffusion of these industrial manipulators, their adoption possesses some difficulties reducing their performance. These issues span from the compensation of kinematics inaccuracies (kinematics calibration) to the improvement of dynamics performances (dynamics calibration). The influence of the technological production process on the robot structure may also have an impact; consequently, external agents such as temperature must be considered. Hence, this thesis proposes and investigates new techniques for the calibration of dynamics model.

Starting from the description of the state-of-the-art in dynamics modeling, this work aims to develop a rigid-body dynamic model with particular attention to the variation of friction due to thermal effects. This choice has been motivated by the fact that friction compensation is essential to improve the robot's positioning precision. Thus, by considering that the most common friction models do not take into account the heating of the speed reducers during the motion tasks, to improve high precision positioning performance of mechatronic systems the development of new friction models that are based on temperature estimation has been performed.

In this thesis, a novel procedure for calibrating rigid-body models through the definition of optimal identification/excitation trajectories is presented. The calibration algorithms and their experimentally estimated dynamics parameters set have also been validated through some experiments conducted on industrial robots.

The procedures that have been developed in this thesis only adopt sensors and controllers that are already installed on the machines. No external sensors have been mounted on them to improve the parameter estimation. This feature is fundamental for real industrial applications.

Usually, such experiments are generally performed on a single robot. However, in this thesis work, the data coming from two industrial robots (two replicas of the same model) has been used to verify the repeatability of the results. This is an important aspect to ensure the robustness of the model identification procedures, especially in the case of real industrial applications. The results of the comparison

can be used to ensure improvements in production and machine maintenance.

In summary, the thesis deals with the analysis of Industrial robots' dynamics modeling and calibration. A set of novel and effective strategies for dynamics parameters estimation has been detailed as well. The dynamics models calibrated using the proposed approaches provide high performance in the prediction of motor torques, also taking into account the variation of friction caused by temperature.

The effectiveness of these calibration techniques has been tested on two industrial robots to verify their robustness and repeatability.

LIST OF PUBLICATIONS

The main research topic of this thesis was the robot dynamics with particular focus on the modeling of friction considering temperature effects. Moreover, several collaborations have been made in the field of Human-Robot Interaction (HRI).

The theoretical and practical results achieved in both topics have generated the following publications:

1. Legnani, G., Incerti, G., **Pagani, R.**, & Gheza, M. (2019, August). Modelling and evaluation of the friction in robotic joints considering thermal effects. In International Design Engineering Technical Conferences and Computers and Information in Engineering Conference (Vol. 59230, p. V05AT07A062). American Society of Mechanical Engineers.

2. Nuzzi, C., Pasinetti, S., **Pagani, R.**, Docchio, F., & Sansoni, G. (2019, September). Hand gesture recognition for collaborative workstations: A smart command system prototype. In International Conference on Image Analysis and Processing (pp. 332-342). Springer, Cham.

3. **Pagani, R.**, Padula, F., Legnani, G., Loxton, R., & Visioli, A. (2019, November). A fractional model of the friction-temperature behavior in robot joints. In 2019 7th International Conference on Control, Mechatronics and Automation (ICCMA) (pp. 157-161). IEEE.

4. Nuzzi, C., Ghidini, S., **Pagani, R.**, & Ragni, F. (2020, March). RemindLy: A Personal Note-bot Assistant. In Companion of the 2020 ACM/IEEE International Conference on Human-Robot Interaction (pp. 631-632).

5. **Pagani, R.**, Legnani, G., Incerti, G., & Gheza, M. (2020). Evaluation and modeling of the friction in robotic joints considering thermal effects. Journal of Mechanisms and Robotics, 12(2).

6. Faroni, M., **Pagani, R.**, & Legnani, G. (2020, June). Real-time trajectory scaling for robot manipulators. In 2020 17th International Conference on Ubiquitous Robots (UR) (pp. 533-539). IEEE.

7. Nuzzi, C., Ghidini, S., **Pagani, R.**, Pasinetti, S., Coffetti, G., & Sansoni, G. (2020, June). Hands-Free: a robot augmented reality teleoperation system. In 2020 17th International Conference on Ubiquitous Robots (UR) (pp. 617-624). IEEE.

8. **Pagani, R.**, Legnani, G., Incerti, G., Beschi, M., & Tiboni, M. (2020, August). The Influence of Heat Exchanges on Friction in Robotic Joints: Theoretical Modelling, Identification and Experiments. In International Design Engineering Technical Conferences and Computers and Information in Engineering Conference (Vol. 83990, p. V010T10A053). American Society of Mechanical Engineers.

9. **Pagani, R.**, Nuzzi, C., Ghidelli, M., Borboni, A., Lancini, M., & Legnani, G. (2021). Cobot User Frame Calibration: Evaluation and Comparison between Positioning Repeatability Performances Achieved by Traditional and Vision-Based Methods. Robotics, 10(1), 45.

10. Hao, L., **Pagani, R.**, Beschi, M., & Legnani, G. (2021). Dynamic and Friction Parameters of an Industrial Robot: Identification, Comparison and Repetitiveness Analysis. Robotics, 10(1), 49.

11. Nuzzi, C., Pasinetti, S., **Pagani, R.**, Ghidini, S., Beschi, M., Coffetti, G., & Sansoni, G. (2021). MEGURU: a gesture-based robot program builder for Meta-Collaborative workstations. Robotics and Computer-Integrated Manufacturing, 68, 102085.

12. Nuzzi, C., Pasinetti, S., **Pagani, R.**, Coffetti, G., & Sansoni, G. (2021). HANDS: an RGB-D dataset of static hand-gestures for human-robot interaction. Data in Brief, 35, 106791.

1

ROBOT DYNAMICS MODEL DERIVATION

An industrial robot is defined as a mechanical machine that is programmed to automatically perform production related tasks in industrial environments. Industrial robots are considered to be a form of flexible automation since they are reprogrammable and can be used for many different types of robotic applications. Robots are growing into the preferred automation choice for manufacturers as they are extremely effective for increasing productivity, producing high-quality products, and reducing costs.

In the last decade, the performance of Industrial robots (IRs) has improved thanks to the effort of both robotic researchers and manufacturers. However, their adoption in tasks where interactions with the environment are required is still an open challenge. In particular, their kinematics and dynamics precision are still inadequate to face complex tasks such as robotic machining.

Moreover, to limit the purchase costs and energy consumption of IRs, recently the robotic industry focused more on designing lighter and weaker robots [1]. The obvious drawback of such structures, however, is a general loss of mechanical performance. For example, the robot's lower mechanical stiffness reduces the kinematics and dynamics precision and limits the noise rejection capability when the robot is subjected to external high-frequency forces as in a machining task. A potential solution to this issue involves model-based control strategies adopting accurate robot modeling. These strategies based on the robot dynamics model calibration improve the dynamics performance during interaction tasks [2].

However, the estimation of the model parameters is not an easy task since most of the identification strategies are ad hoc solutions that require comprehensive experience and scientific knowledge. This is why the research community focused on the development of intelligent algorithms to automatically create mathematical models for the parameters estimation task.

Motivations and Problem Statement

Robotic machining is considered an alternative to CNC machining, especially when the material removal is required during the process. The main advantage of using IRs instead of CNC machines is the lower price of IRs and their faster working volume. However, IRs are not commonly adopted in machining applications although this evident benefit for companies. As reported in [3], within the total sales of 2011, the percentage of IRs used in material removal operations is less than 5%. Nonetheless, recent technological advances improve IRs' mechani-

cal behavior, thus reducing the time required to generate reliable working paths. This resulted in the increased adoption of robotic cells for machining applications.

It is worth noting that the efforts of companies who conducted independent research to produce competitive commercial products also helped to further improve IR technology. For example, IRs have been adopted in machining tasks as an alternative to traditional CNC machining. Here, the robot path is controlled using the tool center point (TCP) similar to traditional applications of CNC machining, such as milling, turning, drilling and cutting.

Companies also adopted IRs for surface finish tasks, which are based on controlling the force of the working tool on the workpiece surface. Such applications are more focused on grinding, brushing, polishing, and debarring tasks. Therefore, the robot's main purpose is to execute manufacturing processes that traditionally have been conducted by humans.

As reported in [4] sources of error in robot machining belong to three macro-categories: (i) environment-dependent errors, (ii) robot-dependent errors, and (iii) process-dependent errors. Robot-dependent errors are summarized as:

- Low kinematics precision in terms of absolute positioning error. For a standard robot manipulator, this usually means up to a tenth of a millimeter considering that position precision can be assumed as 10 times the pose repeatability.
- Low stiffness of the mechanical structure, usually a serial kinematic open chain. This error is also intensified in the case of mechanical structures made of lightweight materials.
- High dynamics tracking error during task execution, caused by external forces due to the material removal process acting on the robot TCP. This is particularly evident in hard material machining. The machining forces can easily reach more than 100 N, causing deviations of the TCP position of several millimeters.

Kinematics Precision

In the absence of external forces acting on the TCP, the absolute positioning error depends mostly on geometric inaccuracies. Typically, the robot manufacturer provides a default rough kinematic calibration that partially compensates for geometric errors. Furthermore, scientific literature provides extensive examples of algorithms to perform absolute kinematics calibration [5, 6, 7, 8, 9]. These procedures typically require external sensors, such as cameras or laser trackers. Some robot manufacturers and companies also provide absolute kinematics calibration as an additional service.

Dynamics Precision

Dynamics calibration and model-based control strategies improve the robot's performance reducing the tracking error [10]. The robot dynamics model can be extended considering also joints and links stiffness, defining also so-called "extended flexible joint dynamics model" [11].

Dynamics model calibration procedures exploit identification techniques for estimating inertial parameters of the robot arm and the friction contribution, such as mass, the first momentum, and links' inertia [12].

It is important to discriminate between different applications of the calibrated dynamics model. For robot manufacturers, the model is embedded in the robot controller defining well-known model-based control strategies [13] such as computed torques control algorithm. For end-users, a standard IR controller provides only high-level functionalities preventing them from interfering with the low-level control layer of the robot. Hence, positions, velocities, and current control loops are typically inaccessible. Moreover, external control strategies applied on a standard IR controller have limited performance, mainly because of the reduced bandwidth due to the communication channel's rate.

Advanced control strategies may be designed using special versions of robot controllers, also known as "open" controllers [14] that allow high-speed real-time communication between robot controllers and external entities. In this case, several control options may be adopted, for example, closing high-frequency external control loops exploiting external sensors.

1.1 RIGID BODY DYNAMICS

The dynamics model of a manipulator describes the relationship between joint actuator torques and the motion of the structure. The derivation of the dynamics model of a manipulator plays a fundamental role for (i) motion simulation, (ii) design of control algorithms, and (iii) analysis of manipulator structures.

Several mathematical formulations allow the computation of the dynamics model of the robot. A common requirement is the model calibration, which involves a model reduction determining its own "minimal form" (i. e. a model without parameters that are not observable) to be used for the experimental dynamics parameters estimation. Many researchers have investigated methodologies involving linear reductions of the rigid body model into a base set of lumped dynamics model parameters (BP) to be estimated. It is also experimentally demonstrated that, for a given trajectory, only a smaller subset of essential parameters (EP) is significant.

The dynamics model of a robotic manipulator can be expressed by exploiting two approaches; however, other mathematical formulations such as in [15] can be found in the literature, but their use is limited to specific applications. The

first method is based on the Lagrange formulation and is conceptually simple and systematic. The second one is based on the Newton-Euler formulation and generates the model in a recursive form; it is computationally more efficient since it exploits the typically open structure of the manipulator kinematics chain. Both methods result in the same outcome, hence the choice of the method depends on the user's preference. The mathematical formulation of this chapter is based on [13] and it considers a standard IR with an anthropomorphous structure based on open-chain kinematics.

Considering the Lagrangian formulation, the dynamics of a robot manipulator can be expressed by considering motor torques τ as the sum of the inertial effects τ_i, the friction contribution τ_i and the external forces acting on the robot end-effectorτ_e. Thus, it is possible to write:

$$\boldsymbol{\tau} = \mathbf{M}(\mathbf{q})\ddot{\mathbf{q}} + \mathbf{V}(\mathbf{q}, \dot{\mathbf{q}})\dot{\mathbf{q}} + \mathbf{G}(\mathbf{q}) + \mathbf{F}_v\dot{\mathbf{q}} + \mathbf{F}_s sng(\dot{\mathbf{q}}) - \mathbf{J}^T(\mathbf{q})\mathbf{h}_e \tag{1}$$

where q, $\dot{q}$ and $\ddot{q}$ are the joint position, velocity and acceleration vectors respectively, $M(q)$ the inertial matrix, $V(q, \dot{q})$ the matrix containing the centrifugal and Coriolis terms, $G(q)$ the gravitational term, $J^T(q)$ is the Jacobian matrix and F_e denotes the vector of the forces and the moments exercised by the end-effector on the environment. A simplified model of friction considers only the viscous friction and static friction torques, where F_v denotes the diagonal matrix with viscous friction terms, while for the static one may consider the Coloumb friction torques $F_s sng(\dot{q})$ where F_s is a diagonal matrix of static friction coefficients.

An important property of the dynamics model is the linearity with respect to the dynamics parameters characterizing the manipulator links and rotors. For an open-chain rigid robot, equation (1) can be rewritten as:

$$\boldsymbol{\tau} = \mathbf{Y}(\mathbf{q}, \dot{\mathbf{q}}, \ddot{\mathbf{q}})\boldsymbol{\pi} \tag{2}$$

where $\boldsymbol{\pi}$ is the set of dynamics model parameters and the matrix function K is a generalized acceleration matrix dependent only on the geometry and on the kinematics state of the robotic arm, even known as dynamics regressor.

Each link of the manipulator has 11 inertial parameters plus the terms related to the friction model. The number of required friction parameters is based on the friction model complexity. The inertial parameters of each j-th link are:

$$\boldsymbol{\pi}_j = \begin{bmatrix} m_j & mP_{jx} & mP_{jy} & mP_{jz} & I_{jxx} & I_{jxy} & I_{jxz} & I_{jyy} & I_{jyz} & I_{jzz} & I_{mj} \end{bmatrix}^T \tag{3}$$

where m_j is the link mass, $[mP_{jx}, mP_{jy}, mP_{jz}]$ are the three components of the first moment of inertia, $[I_{jxx}, I_{jxy}, I_{jxz}, I_{jyy}, I_{jyz}, I_{jzz}]$ are the six components of the inertial tensor, and I_{mj} denotes the scalar value of inertia rotor.

It is worth noting that not all the dynamics parameters of the π vector give a contribution to robot dynamics. Only a restricted amount of relative motion between the links is allowed, hence, actuators apply torques along the joint axes

only. However, not all the parameters are observable by measuring the joints' position, velocity, acceleration, and torque. Hence, the dynamics model parameters can be grouped into 3 main categories:

- (i) *absolutely identifiable* parameters
- (ii) *identifiable in linear combination* with others parameters
- (iii) *unidentifiable* parameters

This classification results in (i) independent columns of matrix $\mathbf{Y}$ for absolutely identifiable parameters, (ii) linear combination columns of matrix $\mathbf{Y}$ for parameters identifiable as a linear combination of others, and (iii) unidentifiable parameters that do not give any contribution to the robot joint torques and, as such, can be removed (e. g. the mass of a link fixed to the ground).

However, knowing the values of the dynamics model parameters is necessary to adopt robot dynamics models in model-based control algorithms. This information is not always known and even when they are available their precision is not always adequate. For example, CAD modeling techniques allow the computation of inertial parameters of various components such as links, actuators, and transmissions according to their geometry and material. Nevertheless, the estimates obtained using such techniques are inaccurate because of the simplification typically introduced by geometric modeling. Moreover, complex dynamics effects such as joint friction cannot be taken into account by the procedure.

Identification techniques exploit the linearity property of the dynamics model with respect to a suitable set of dynamics parameters. They allow the computation of the parameter vector π from the measurements of joint torques τ and of relevant quantities for the evaluation of the matrix $\mathbf{Y}$ when suitable motion trajectories are imposed to the manipulator. The minimum representation of the dynamics parameters is fundamental, as highlighted in [12] since the minimal system representation is mandatory to avoid bias in the numerical identification procedures.

It is worth noting that the aim of dynamics parameters estimation is not to obtain exact values of any single parameter, but rather to estimate the values of parameters that are expressed as a linear combination of others. In this regard, model-based control algorithms are mainly adopted to minimize the error between the predicted torques and the measured ones.

1.2 REGRESSION MATRIX, THE NEWTON-EULER (N-E) METHOD

As previously mentioned, there are two approaches for establishing the robot dynamic model. They are based on the Lagrange formulations [16] and Newton–Euler formulation [17]. Authors of [18] proved that both models can be

equivalent. However, the computation cost of the robot dynamics model coming from Lagrange formula is higher than the model built using the Newton–Euler method. Therefore, the dynamics model of the robot is derived from Newton–Euler formulation.

The model contains two sets of recursions. Forward recursions transfer the arms' velocities and accelerations from the base to the end-effector, while backward recursions transfer the forces of the joints from the end-effector to the base. The complete formulations as in [19, 17] are:

- Forward recursions, from Link $0 \rightarrow 5$:

$$\omega_{i+1} = \mathbf{R}_i^{i+1}\omega_i + \dot{\theta}_{i+1}z_{i+1}^{i+1} \tag{4}$$
$$\dot{\omega}_{i+1} = \mathbf{R}_i^{i+1}\dot{\omega}_i + \mathbf{R}_i^{i+1}\omega_i \times \dot{\theta}_{i+1}z_{i+1}^{i+1} + \ddot{\theta}_{i+1}z_{i+1}^{i+1} \tag{5}$$
$$\dot{v}_{i+1} = \mathbf{R}_i^{i+1}\left[\dot{\omega}_i \times \mathbf{p}_{i+1}^i + \omega_i \times \left(\omega_i \times \mathbf{p}_{i+1}^i\right) + \dot{v}_i\right] \tag{6}$$
$$\dot{v}_{c_{i+1}} = \dot{\omega}_{i+1} \times \mathbf{p}_{c_{i+1}}^{i+1} + \omega_{i+1} \times \left(\omega_{i+1} \times \mathbf{p}_{c_{i+1}}^{i+1}\right) + \dot{v}_{i+1} \tag{7}$$
$$\hat{f}_{i+1} = m_{i+1}\dot{v}_{c_{i+1}} \tag{8}$$
$$\hat{n}_{i+1} = \mathbf{I}_{i+1}^{c_{i+1}}\dot{\omega}_{i+1} + \omega_{i+1} \times \mathbf{I}_{i+1}^{c_{i+1}}\omega_{i+1} \tag{9}$$

  with $\omega_0 = \dot{\omega}_0 = 0$.
- Backward recursions, from Link $6 \rightarrow 1$:

$$f_i = \mathbf{R}_{i+1}^i f_{i+1} + \hat{f}_i \tag{10}$$
$$n_i = \hat{n}_i + \mathbf{R}_{i+1}^i n_{i+1} + \mathbf{p}_{c_i}^i \times \hat{f}_i + \mathbf{p}_{i+1}^i \times \mathbf{R}_{i+1}^i f_{i+1} \tag{11}$$
$$\tau_i = (n_i)^T z_i^i \tag{12}$$

Where:

$\mathbf{R}_i^{i+1}$ is the rotation matrix from link i to link $i+1$.
$\dot{\theta}_i$ and $\ddot{\theta}_i$ are the angle speed and acceleration of link i.
ω_i, $\dot{\omega}_i$ and $\ddot{\omega}_i$ are the angular position, speed, and acceleration vector of link i.
$\dot{v}_i$ is the linear acceleration of the origin of the link coordination system.
$\dot{v}_{ci}$ is the linear acceleration of the center of mass of the link i.
$\hat{f}_i$ is the inertial force of the center of mass of the link i.
$\hat{n}_i$ is the inertial torque of the center of mass of the link i.
f_i is the force on link i exerted from the link $i-1$.
n_i is the torque on link i exerted from the link $i-1$.
τ_i is linear actuator force of the link i.
$\mathbf{p}_{i+1}^i$ is the transformation vector of the link i, which is pointed from the origin of link i to link $i-1$.

m_i is the mass of link i.
$\mathbf{I}_{i+1}^{c_{i+1}}$ is the inertia tensor of link $i+1$ based on its own center of mass coordinate.
$\mathbf{p}_{c_i}^{i}$ is the vector of the center of mass of the link i in coordination with link i.
$z_i^i = \begin{bmatrix} 0 & 0 & 1 \end{bmatrix}^T$, with the meaning of selecting the value followed the z-axis of DH coordinates.

It should be pointed out that with $\dot{\mathbf{v}}_i = \mathbf{g}$ the gravity acceleration of each link will be counted into the recursion process.

1.2.1 *(N-E) Recursion Algorithm*

The Newton-Euler linearization started from Equation (11):

$$\mathbf{n}_i = \hat{\mathbf{n}}_i + \mathbf{R}_{i+1}^i \mathbf{n}_{i+1} + \mathbf{p}_{c_i}^i \times \hat{\mathbf{f}}_i + \mathbf{p}_{i+1}^i \times \mathbf{R}_{i+1}^i \mathbf{f}_{i+1} \tag{13}$$

with

$$\hat{\mathbf{f}}_i = m_i \dot{\mathbf{v}}_{c_i} \tag{14}$$

$$\dot{\mathbf{v}}_{c_{i+1}} = \dot{\boldsymbol{\omega}}_{i+1} \times \mathbf{p}_{c_{i+1}}^{i+1} + \boldsymbol{\omega}_{i+1} \times \left(\boldsymbol{\omega}_{i+1} \times \mathbf{p}_{c_{i+1}}^{i+1} \right) + \dot{\mathbf{v}}_{i+1} \tag{15}$$

$$\hat{\mathbf{n}}_{i+1} = \mathbf{I}_{i+1}^{c_{i+1}} \dot{\boldsymbol{\omega}}_{i+1} + \boldsymbol{\omega}_{i+1} \times \mathbf{I}_{i+1}^{c_{i+1}} \boldsymbol{\omega}_{i+1} \tag{16}$$

Then, taking into account Equation (15) and Equation (14), and rewriting Equation (16):

$$\hat{\mathbf{f}}_i = m_i \left[\dot{\boldsymbol{\omega}}_i \times \mathbf{p}_{c_i}^i + \boldsymbol{\omega}_i \times \left(\boldsymbol{\omega}_i \times \mathbf{p}_{c_i}^i \right) + \dot{\mathbf{v}}_i \right] \tag{17}$$

$$\hat{\mathbf{n}}_i = \mathbf{I}_i^{c_i} \dot{\boldsymbol{\omega}}_i + \boldsymbol{\omega}_i \times \mathbf{I}_i^{c_i} \boldsymbol{\omega}_i \tag{18}$$

Then Equation (13) can be written with Equation (17) and Equation (18):

$$\begin{aligned} \mathbf{n}_i =& \mathbf{I}_i^{c_i} \dot{\boldsymbol{\omega}}_i + \boldsymbol{\omega}_i \times \mathbf{I}_i^{c_i} \boldsymbol{\omega}_i + \mathbf{R}_{i+1}^i \mathbf{n}_{i+1} + \mathbf{p}_{i+1}^i \times \mathbf{R}_{i+1}^i \mathbf{f}_{i+1} \\ &+ m_i \left[\mathbf{p}_{c_i}^i \times \left(\dot{\boldsymbol{\omega}}_i \times \mathbf{p}_{c_i}^i \right) \right] + m_i \left\{ \mathbf{p}_{c_i}^i \times \left[\boldsymbol{\omega}_i \times \left(\boldsymbol{\omega}_i \times \mathbf{p}_{c_i}^i \right) \right] \right\} + m_i \left(\mathbf{p}_{c_i}^i \times \dot{\mathbf{v}}_i \right) \end{aligned} \tag{19}$$

Remembering the cross product and the vector triple product properties:

$$\begin{aligned} & a \times [b \times (b \times a)] = b \times (a^T a \mathbf{I}_{3*3} - a a^T) b \\ \Rightarrow \quad & a \times (b \times a) = (a^T a \mathbf{I}_{3*3} - a a^T) b \end{aligned}$$

Where $\mathbf{I}_{3*3}$ is the 3 by 3 identical matrix, it is possible to modify the last part of Equation (19):

$$m_i \left[\mathbf{p}_{c_i}^i \times \left(\dot{\boldsymbol{\omega}}_i \times \mathbf{p}_{c_i}^i \right) \right] = m_i \left(\mathbf{p}_{c_i}^{i\,T} \mathbf{p}_{c_i}^i \mathbf{I}_{3*3} - \mathbf{p}_{c_i}^i \mathbf{p}_{c_i}^{i\,T} \right) \dot{\boldsymbol{\omega}}_i$$

$$m_i \left\{ \mathbf{p}_{c_i}^i \times \left[\boldsymbol{\omega}_i \times \left(\boldsymbol{\omega}_i \times \mathbf{p}_{c_i}^i \right) \right] \right\} = m_i \left[\boldsymbol{\omega}_i \times \left(\mathbf{p}_{c_i}^{i\,T} \mathbf{p}_{c_i}^i \mathbf{I}_{3*3} - \mathbf{p}_{c_i}^i \mathbf{p}_{c_i}^{i\,T} \right) \boldsymbol{\omega}_i \right]$$

Using the parallel axis theorem [20], the two equations above became:

$$\mathbf{I}_i^i = \mathbf{I}_i^{c_i} + m_i \left(\mathbf{p}_{c_i}^{i\,T} \mathbf{p}_{c_i}^i \mathbf{I}_{3*3} - \mathbf{p}_{c_i}^i \mathbf{p}_{c_i}^{i\,T} \right) \Rightarrow m_i \left(\mathbf{p}_{c_i}^{i\,T} \mathbf{p}_{c_i}^i \mathbf{I}_{3*3} - \mathbf{p}_{c_i}^i \mathbf{p}_{c_i}^{i\,T} \right) = \mathbf{I}_i^i - \mathbf{I}_i^{c_i}$$

Then it is trivial to obtain:

$$m_i \left[\mathbf{p}_{c_i}^i \times \left(\dot{\boldsymbol{\omega}}_i \times \mathbf{p}_{c_i}^i \right) \right] = \left(\mathbf{I}_i^i - \mathbf{I}_i^{c_i} \right) \dot{\boldsymbol{\omega}}_i$$
$$m_i \left\{ \mathbf{p}_{c_i}^i \times \left[\boldsymbol{\omega}_i \times \left(\boldsymbol{\omega}_i \times \mathbf{p}_{c_i}^i \right) \right] \right\} = \boldsymbol{\omega}_i \times \left(\mathbf{I}_i^i - \mathbf{I}_i^{c_i} \right) \boldsymbol{\omega}_i$$

Now Equation (19) can be written as follows:

$$\mathbf{n}_i = \mathbf{R}_{i+1}^i \mathbf{n}_{i+1} + \mathbf{p}_{i+1}^i \times \mathbf{R}_{i+1}^i \mathbf{f}_{i+1} + \mathbf{I}_i^i \dot{\boldsymbol{\omega}}_i + \boldsymbol{\omega}_i \times \mathbf{I}_i^i \boldsymbol{\omega}_i + m_i \mathbf{p}_{c_i}^i \times \dot{\mathbf{v}}_i \tag{20}$$

Starting from Joint 6, Equation (10), Equation (17) and (20) can be written as:

$$\mathbf{f}_6 = \hat{\mathbf{f}}_6 = m_6 \dot{\mathbf{v}}_6 + m_6 \dot{\boldsymbol{\omega}}_6 \times \mathbf{p}_{c_6}^6 + m_6 \boldsymbol{\omega}_6 \times \left(\boldsymbol{\omega}_6 \times \mathbf{p}_{c_6}^6 \right) \tag{21}$$
$$\mathbf{n}_6^6 = -\dot{\mathbf{v}}_6 \times m_6 \mathbf{p}_{c_6}^6 + \mathbf{I}_6^6 \dot{\boldsymbol{\omega}}_6 + \boldsymbol{\omega}_6 \times \mathbf{I}_6^6 \boldsymbol{\omega}_6 \tag{22}$$

It is now possible to transfer Equation (21) and (22) to matrix form:

$$\begin{bmatrix} \mathbf{f}_6 \\ \mathbf{n}_6^6 \end{bmatrix} = \begin{bmatrix} \dot{\mathbf{v}}_6 & [\dot{\boldsymbol{\omega}}_6\times] + [\boldsymbol{\omega}_6\times][\boldsymbol{\omega}_6\times] & \mathbf{0} \\ \mathbf{0} & -[\dot{\mathbf{v}}_6\times] & [\dot{\boldsymbol{\omega}}_6\cdot] + [\boldsymbol{\omega}_6\times][\boldsymbol{\omega}_6\cdot] \end{bmatrix} \begin{bmatrix} m_6 \\ m_6 p_{c_6}^6|_x \\ m_6 p_{c_6}^6|_y \\ m_6 p_{c_6}^6|_z \\ I_{xx} \\ I_{xy} \\ I_{xz} \\ I_{yy} \\ I_{yz} \\ I_{zz} \end{bmatrix} \triangleq \mathbf{A}_6 \boldsymbol{\pi}_6 \tag{23}$$

It should be pointed that $[\mathbf{R}_{i+1}^i \mathbf{f}_{i+1}]$ in Equation (10) and $[\mathbf{R}_{i+1}^i \mathbf{n}_{i+1} + \mathbf{p}_{i+1}^i \mathbf{R}_{i+1}^i \mathbf{f}_{i+1}]$ in Equation (11) for Joint 6 is 0 because there is no forward axis.

The same procedure can be applied also on Joint 5:

$$\mathbf{f}_5 = \hat{\mathbf{f}}_5 + \mathbf{R}_5^6 \mathbf{f}_6 \tag{24}$$
$$\mathbf{n}_5^5 = -\dot{\mathbf{v}}_5 \times m_5 \mathbf{p}_{c_5}^5 + \mathbf{I}_5^5 \dot{\boldsymbol{\omega}}_5 + \boldsymbol{\omega}_5 \times \mathbf{I}_5^5 \boldsymbol{\omega}_5 + \mathbf{R}_6^5 \mathbf{n}_6 + \mathbf{p}_6^5 \times \mathbf{R}_6^5 \mathbf{f}_6 \tag{25}$$

and again the matrix form is:

$$\begin{bmatrix} f_5 \\ n_5^5 \end{bmatrix} = A_5\pi_5 + \begin{bmatrix} R_6^5 & 0 \\ [p_6^5\times]R_6^5 & R_6^5 \end{bmatrix} \begin{bmatrix} f_6 \\ n_6^6 \end{bmatrix} \triangleq A_5\pi_5 + T_{56}A_6\pi_6 \tag{26}$$

Applying the same operation for the other joints it is possible to obtain:

$$\begin{bmatrix} f_4 \\ n_4^4 \end{bmatrix} = A_4\pi_4 + T_{45}A_5\pi_5 + T_{45}T_{56}A_6\pi_6 \tag{27}$$

$$\begin{bmatrix} f_3 \\ n_3^3 \end{bmatrix} = A_3\pi_3 + T_{34}A_4\pi_4 + T_{34}T_{45}A_5\pi_5 + T_{34}T_{45}T_{56}A_6\pi_6 \tag{28}$$

$$\begin{bmatrix} f_2 \\ n_2^2 \end{bmatrix} = A_2\pi_2 + T_{23}A_3\pi_3 + T_{23}T_{34}A_4\pi_4 + T_{23}T_{34}T_{45}A_5\pi_5 + T_{23}T_{34}T_{45}T_{56}A_6\pi_6 \tag{29}$$

$$\begin{bmatrix} f_1 \\ n_1^1 \end{bmatrix} = A_1\pi_1 + T_{12}A_2\pi_2 + T_{12}T_{23}A_3\pi_3 + T_{12}T_{23}T_{34}A_4\pi_4 + T_{12}T_{23}T_{34}T_{45}A_5\pi_5 + T_{12}T_{23}T_{34}T_{45}T_{56}A_6\pi_6 \tag{30}$$

Finally, regrouping from Equation (23) to Equation (30) and then transfer to matrix form:

$$\begin{bmatrix} f_1 \\ n_1^1 \\ f_2 \\ n_2^2 \\ f_3 \\ n_3^3 \\ f_4 \\ n_4^4 \\ f_5 \\ n_5^5 \\ f_6 \\ n_6^6 \end{bmatrix} = \begin{bmatrix} \bar{U}_{11} & \bar{U}_{12} & \bar{U}_{13} & \bar{U}_{14} & \bar{U}_{15} & \bar{U}_{16} \\ 0_{6*10} & \bar{U}_{22} & \bar{U}_{23} & \bar{U}_{24} & \bar{U}_{25} & \bar{U}_{26} \\ 0_{6*10} & 0_{6*10} & \bar{U}_{33} & \bar{U}_{34} & \bar{U}_{35} & \bar{U}_{36} \\ 0_{6*10} & 0_{6*10} & 0_{6*10} & \bar{U}_{44} & \bar{U}_{45} & \bar{U}_{46} \\ 0_{6*10} & 0_{6*10} & 0_{6*10} & 0_{6*10} & \bar{U}_{55} & \bar{U}_{56} \\ 0_{6*10} & 0_{6*10} & 0_{6*10} & 0_{6*10} & 0_{6*10} & \bar{U}_{66} \end{bmatrix} \begin{bmatrix} \pi_1 \\ \pi_2 \\ \pi_3 \\ \pi_4 \\ \pi_5 \\ \pi_6 \end{bmatrix} = \bar{Y}_{36*60}\Pi_{60*1} \tag{31}$$

Where:

$$\bar{U}_{ij} = \begin{cases} A_i, & i = j \\ T_{i,i+1}T_{i+1,i+2}\cdots T_{j-1,j}A_j, & i \neq j \end{cases}$$

$$\Pi = \begin{bmatrix} \pi_1 & \pi_2 & \pi_3 & \pi_4 & \pi_5 & \pi_6 \end{bmatrix}^T$$

Because only the torque in the z-axis direction is considered, it is possible to write:

$$\tau_i = \begin{bmatrix} 0 & 0 & 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} f_i \\ n_i^i \end{bmatrix} = \begin{bmatrix} 0 & 0 & 0 & 0 & 0 & 1 \end{bmatrix} \times \begin{bmatrix} f_{ix} & f_{iy} & f_{iz} & n_{ix}^i & n_{iy}^i & n_{iz}^i \end{bmatrix}^T = n_{iz}^i \tag{32}$$

Thus the concluded matrix can be written as:

$$\tau_d = \begin{bmatrix} \tau_{Joint1} \\ \tau_{Joint2} \\ \tau_{Joint3} \\ \tau_{Joint4} \\ \tau_{Joint5} \\ \tau_{Joint6} \end{bmatrix}_{6*1} = Y_{6*60}\Pi_{60*1} = \begin{bmatrix} U_{11} & U_{12} & U_{13} & U_{14} & U_{15} & U_{16} \\ 0_{6*10} & U_{22} & U_{23} & U_{24} & U_{25} & U_{26} \\ 0_{6*10} & 0_{6*10} & U_{33} & U_{34} & U_{35} & U_{36} \\ 0_{6*10} & 0_{6*10} & 0_{6*10} & U_{44} & U_{45} & U_{46} \\ 0_{6*10} & 0_{6*10} & 0_{6*10} & 0_{6*10} & U_{55} & U_{56} \\ 0_{6*10} & 0_{6*10} & 0_{6*10} & 0_{6*10} & 0_{6*10} & U_{66} \end{bmatrix} \begin{bmatrix} \pi_1 \\ \pi_2 \\ \pi_3 \\ \pi_4 \\ \pi_5 \\ \pi_6 \end{bmatrix} \tag{33}$$

Where: Y is the matrix of $\bar{Y}$ with the values of the z-axis direction only and the subscript d stands for *dynamic inertia*.

1.2.2 *Friction and Motor Inertia*

Friction and motor inertia must be considered to obtain a correct model of the robot dynamics. Hence, the regressor must be modified as follows, where the total torque output of each joint is expressed as:

$$\tau = \tau_d + \tau_f + \tau_m \tag{34}$$

Where τ_f is the friction torque and τ_m is the torque that depends on motor inertia.

Friction behavior will be explained in more detail in Chapter 4. In the context of this chapter, it is enough to state that it can be modeled with good approximation by a polynomial function as in [21]:

$$\tau_f = f_1 sign(\dot{\theta}) + f_2\dot{\theta} + f_3\dot{\theta}^2 sign(\dot{\theta}) + f_4\dot{\theta}^3 \tag{35}$$

where $\dot{\theta}$ is the velocity of the joints, and f_1, f_2, f_3 and f_4 are the coefficients of the friction model. Equation (35) can be rewritten in matrix form as:

$$\tau_f = \begin{bmatrix} sign(\dot{\theta}) & \dot{\theta} & \dot{\theta}^2 sign(\dot{\theta}) & \dot{\theta}^3 \end{bmatrix} \begin{bmatrix} f_1 \\ f_2 \\ f_3 \\ f_4 \end{bmatrix} = Y_f \pi_f \tag{36}$$

Where Y_f is the friction information matrix that can be used in the robot dynamic model, and π_f is the friction coefficient. The motor inertia equation can be written in matrix form as:

$$\tau_m = I_m \ddot{\theta} \tag{37}$$

$$\Rightarrow \quad \tau_m = \left[I_m\right]\left[\ddot{\theta}\right] = Y_m \pi_m \tag{38}$$

where I_m is the inertia of the motor, and $\ddot{\theta}$ is the joint acceleration.

Therefore, based on the Newton-Euler equation discussed in Section 1.2.1, and on the friction model and motor inertia previously explained, the parameters of each joint can be computed as:

$$\pi_d = \begin{bmatrix} m & mP_x & mP_y & mP_z & I_{xx} & I_{xy} & I_{xz} & I_{yy} & I_{yz} & I_{zz} \end{bmatrix}^T \tag{39}$$

$$\pi_f = \begin{bmatrix} f_1 & f_2 & f_3 & f_4 \end{bmatrix}^T \tag{40}$$

$$\pi_m = \begin{bmatrix} I_m \end{bmatrix} \tag{41}$$

and the matrix equation, based on (34) can be written as:

$$\tau = Y_d \pi_d + Y_f \pi_f + Y_m \pi_m = \begin{bmatrix} Y_d & Y_f & Y_m \end{bmatrix} \begin{bmatrix} \pi_d \\ \pi_f \\ \pi_m \end{bmatrix} \tag{42}$$

Where:

$$Y_d = \begin{bmatrix} u_{11} & u_{12} & u_{13} & u_{14} & u_{15} & u_{16} \\ 0_{1*10} & u_{22} & u_{23} & u_{24} & u_{25} & u_{26} \\ 0_{1*10} & 0_{1*10} & u_{33} & u_{34} & u_{35} & u_{36} \\ 0_{1*10} & 0_{1*10} & 0_{1*10} & u_{44} & u_{45} & u_{46} \\ 0_{1*10} & 0_{1*10} & 0_{1*10} & 0_{1*10} & u_{55} & u_{56} \\ 0_{1*10} & 0_{1*10} & 0_{1*10} & 0_{1*10} & 0_{1*10} & u_{66} \end{bmatrix} \tag{(42)a}$$

$$Y_f = \begin{bmatrix} y_{f,\,J1} & 0_{1*4} & 0_{1*4} & 0_{1*4} & 0_{1*4} & 0_{1*4} \\ 0_{1*4} & y_{f,\,J2} & 0_{1*4} & 0_{1*4} & 0_{1*4} & 0_{1*4} \\ 0_{1*4} & 0_{1*4} & y_{f,\,J3} & 0_{1*4} & 0_{1*4} & 0_{1*4} \\ 0_{1*4} & 0_{1*4} & 0_{1*4} & y_{f,\,J4} & 0_{1*4} & 0_{1*4} \\ 0_{1*4} & 0_{1*4} & 0_{1*4} & 0_{1*4} & y_{f,\,J5} & 0_{1*4} \\ 0_{1*4} & 0_{1*4} & 0_{1*4} & 0_{1*4} & 0_{1*4} & y_{f,\,J6} \end{bmatrix} \tag{(42)b}$$

$$\mathbf{Y}_m = \begin{bmatrix} y_{m,\,J1} & 0 & 0 & 0 & 0 & 0 \\ 0 & y_{m,\,J2} & 0 & 0 & 0 & 0 \\ 0 & 0 & y_{m,\,J3} & 0 & 0 & 0 \\ 0 & 0 & 0 & y_{m,\,J4} & 0 & 0 \\ 0 & 0 & 0 & 0 & y_{m,\,J5} & 0 \\ 0 & 0 & 0 & 0 & 0 & y_{m,\,J6} \end{bmatrix} \tag{(42)c}$$

$$\boldsymbol{\pi}_d = \begin{bmatrix} \pi_{d,\,J1} & \pi_{d,\,J2} & \pi_{d,\,J3} & \pi_{d,\,J4} & \pi_{d,\,J5} & \pi_{d,\,J6} \end{bmatrix}^T \tag{(42)d}$$

$$\boldsymbol{\pi}_f = \begin{bmatrix} \pi_{f,\,J1} & \pi_{f,\,J2} & \pi_{f,\,J3} & \pi_{f,\,J4} & \pi_{f,\,J5} & \pi_{f,\,J6} \end{bmatrix}^T \tag{(42)e}$$

$$\boldsymbol{\pi}_m = \begin{bmatrix} \pi_{m,\,J1} & \pi_{m,\,J2} & \pi_{m,\,J3} & \pi_{m,\,J4} & \pi_{m,\,J5} & \pi_{m,\,J6} \end{bmatrix}^T \tag{(42)f}$$

1.3 DYNAMIC MODEL REDUCTION

Researchers focused their attention on the minimal description of rigid-dynamics linear dynamics system [22, 23]. In [23, 24] an analytical method to reduce the full set of dynamics parameters of an open-chain mechanism into the base dynamics parameters (BP) set is detailed, and it is considered a milestone of the field. However, it is shown in [25] that this approach requires a case-by-case analysis to handle special cases where the algorithms are too complex to be automated. Another analytical reduction method is presented in [26] and is based on the definition of multi-poles. This formulation allows the definition of a simple and general geometrical procedure to obtain the base parameters set. A different approach is given in [25, 27, 28], where the identification of the minimum representation is addressed mathematically. The methods impose some suitable numerical thresholds for each joint torque, compute the parameters and their linear combinations, and map torques on the corresponding joints averaged over the threshold. Such identified parameters are named essential dynamics parameters set (EP). The set of EP includes only a combination of inertial parameters observable along any exciting trajectory that generates Y; hence, EP can only be computed mathematically. They are, however, less than BP, since the numerical analysis regroup and eliminates the parameters that show a minimum contribution to the joint torques. This is acceptable for serial IRs with 6-degrees of freedom and a high motor reduction ratio, but it is unsatisfactory for generic mechanisms. These numerical methods are based on a random mapping of the joints' workspace in terms of positions, velocities, and accelerations; moreover, the convergence to a univocal result is not demonstrated by literature. However, it is experimentally demonstrable that selecting a proper threshold modifies the minimum number of required EP. The main drawback of this procedure is that the tuning of the algorithm's parameters depends heavily on the user's expertise.

Thanks to the advances in electrical computing science, the derivation of BP can be performed by using QR factorization and SVD decomposition [29, 30, 31, 32], which results are presented as the relationships among links. The purpose of both decompositions is to factorize a matrix into a product of matrices. QR decomposition [33] is used to decompose a real matrix $\mathbf{A}$ into a product as:

$$\mathbf{A} = \mathbf{QR} \tag{43}$$

Where:

$\mathbf{Q}$ is an orthogonal matrix.
$\mathbf{R}$ is an upper triangular matrix.

This decomposition is often introduced to solve linear least square problems. Moreover, with QR decomposition, it is easy to solve the system with the equation $\mathbf{Ax} = \mathbf{b}$:

$$\mathbf{Ax} = \mathbf{b} \xrightarrow{A=QR} \mathbf{QRx} = \mathbf{b} \xrightarrow{Q^TQ=I} \mathbf{Q^TQRx} = \mathbf{Q^Tb} \Rightarrow \mathbf{Rx} = \mathbf{Q^Tb} \tag{44}$$

Meanwhile, SVD factorization [34] (singular value decomposition) is a general eigenvalue decomposition of a matrix into any $m * n$ matrix through an extension of the polar decomposition. It can be explained with an $m * n$ real matrix $\mathbf{A}$ as:

$$\mathbf{A} = \mathbf{U\Sigma V^T} \tag{45}$$

Where:

$\mathbf{U}$ and $\mathbf{V}$ are $m * m$ and $n * n$ complex unitary matrix, respectively.
$\mathbf{\Sigma}$ is an $m * n$ rectangular diagonal matrix with non-negative real numbers on the diagonal.

The diagonal elements of $\mathbf{\Sigma}$ are the singular values of $\mathbf{A}$, and they are uniquely determined. Whereas $\mathbf{U}$ and $\mathbf{V}$ are not necessary to be unique.

1.3.1 *QR Factorization*

QR factorization is used to search the zero-elements-columns of matrix $\mathbf{R}$ in Equation (47). The corresponding parameter can be eliminated during the identification of the dynamic model because they are unrecognizable, or because they have a linear relationship with other parameters. The torque of each joint can be explained as follows:

$$\boldsymbol{\tau} = \mathbf{Y}\boldsymbol{\pi} \tag{46}$$

where $\mathbf{Y}$ is the information matrix and $\boldsymbol{\pi}$ is a vector containing the set of the dynamic parameters of the links.

Matrix $\mathbf{Y}$ can be factorized by QR decomposition in the following form:

$$\mathbf{Y} = \mathbf{QR} = \begin{bmatrix} \mathbf{Q}_1 & \mathbf{Q}_2 \end{bmatrix} \begin{bmatrix} \mathbf{R}_1 \\ \begin{bmatrix} 0 \end{bmatrix} \end{bmatrix} = \mathbf{Q}_1 \mathbf{R}_1 \tag{47}$$

Where $\mathbf{R}_1$ is an upper triangular matrix:

$$\mathbf{R}_1 = \begin{bmatrix} r_{1,1} & r_{1,2} & r_{1,3} & \cdots & r_{1,60} \\ 0 & r_{2,2} & r_{2,3} & \cdots & r_{2,60} \\ 0 & 0 & r_{3,3} & \cdots & r_{3,60} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & 0 & \cdots & r_{60,60} \end{bmatrix} \tag{48}$$

The vector of main diagonal of $\mathbf{R}_1$ is:

$$\mathbf{r}_{1,diag} = \begin{bmatrix} r_{1,1} & r_{2,2} & r_{3,3} & \cdots & r_{60,60} \end{bmatrix} \tag{49}$$

The identifiable parameters of matrix $\boldsymbol{\pi}$ are selected by the position of the no-zeros elements on the diagonal of the matrix $\mathbf{R}_1$. It is worth noting that also the elements below a certain threshold are considered equal to zero. The results of the QR factorization are shown in Table 2. In this table, the parameters that are not highlighted in yellow are identifiable with the corresponding link. For example, the parameter $I_{5,xx}$ of Joint 1 is identifiable due to its value of -168.5869. However, parameter m_3 of Joint 2 is unidentifiable due to its value of 5.08E-14, which is less than 1/10000.

1.3.2 *SVD Decomposition*

SVD decomposition is applied to solve the rank deficiency problem of $\mathbf{Y}$ and it is used to verify the results of QR factorization. The main purpose of SVD decomposition is to obtain different values in matrix $\mathbf{V}$ with large condition number of matrix $\mathbf{S}$. The computing routine starts with Equation (50):

$$\mathbf{Y} = \mathbf{U}\boldsymbol{\Sigma}\mathbf{V}^T = \mathbf{U} \begin{bmatrix} \mathbf{S} \\ \begin{bmatrix} 0 \end{bmatrix} \end{bmatrix} \mathbf{V}^T \tag{50}$$

Where:

1, $\mathbf{Y}$ is the regression matrix, and has a dimension of $m * n$.

2, $\mathbf{S}$ is a diagonal matrix with non-negative real numbers, and its form is:

$$\mathbf{S} = \begin{bmatrix} r_{1,1} & 0 & 0 & \cdots & 0 \\ 0 & r_{2,2} & 0 & \cdots & 0 \\ 0 & 0 & r_{3,3} & \cdots & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & 0 & \cdots & r_{n,n} \end{bmatrix} \tag{51}$$

or

$$\mathbf{s}_{diag} = \begin{bmatrix} r_{1,1} & r_{2,2} & r_{3,3} & \cdots & r_{n,n} \end{bmatrix} \tag{52}$$

with:

a, The size of matrix $\mathbf{S}$ has been decided by the dimension of matrix $\mathbf{Y}$.
b, All values of matrix $\mathbf{S}$ are non-negative real numbers.
c, The values on the diagonal are in decreasing order, such as $r_{1,1} > r_{2,2} > r_{3,3} > \cdots > r_{n,n}$.
d, $\mathbf{s}_{diag}$ is the vector of diagonal elements of $\mathbf{S}$.

A larger condition number of matrix $\mathbf{S}$ indicates that there are some parameters needed to be removed. They can be found by the location of some particularly different values in the column of matrix $\mathbf{V}$, which is the same as the minimum value in matrix $\mathbf{s}_{diag}$. In this work, the threshold to determine whether the matrix has a high condition number is set to 100. Singular parameters with smaller values are associated with those that cannot be estimated, hence, they have been deleted.

An example of matrix $\mathbf{S}$ and $\mathbf{V}$ of Link 2 is given in Table 1. The condition number of matrix $\mathbf{S}$ is 1.5742E+16, which is more than 100. This leads to the minimum value of 1.15E-16 in the column of I_m in matrix $\mathbf{S}$, which is highlighted in yellow. In this case, there are some minimum values in the column of I_m in matrix $\mathbf{V}$, which are significantly different from others, located in row I_{zz} and I_m. This means that parameter I_{zz} is unidentifiable. And it is clear that I_{zz} and I_m are parallel in this link.

Two things should be pointed before the SVD decomposition process, which are:

1, The friction elements should be introduced, which means π_f will be used in the identification.
2, The results of QR decomposition should be applied. Thus, according to the results of QR factorization, the parameters of m, mP_x, mP_y, mP_z, I_{xx}, I_{xy}, I_{xz}, I_{yy}, I_{yz} and I_{zz} of Link 1 will be eliminated in the decomposition process due to the value of zero.

The surviving parameters resulting from SVD are marked in red in Table 2.

After applying QR factorization and SVD, the merged results of the base parameters selection are highlighted in Table 2. Here, the values highlighted in yellow and red indicate that they are a linear combination of other parameters. Hence, the information matrix created by these parameter pairs is not full rank, and results in unrecognizable parameters. This problem can be solved by selecting parameters that define a full rank information matrix. Finally, Table 3 summarizes the selected dynamic parameters involved in the identification process.

Table 1: The values of matrix of **S** and **V** form SVD results of Link 2

Matrix	Items		12	13	15	16	17	19	20	***61***	***62***	***63***	***64***	***65***
			mP_x	mP_y	I_{xx}	I_{xy}	I_{xz}	$I_{y}z$	I_{zz}	f_1	f_2	f_3	f_4	I_m
S	12	mP_x	1.8091	0	0	0	0	0	0	0	0	0	0	0
	13	mP_y	0	1.6202	0	0	0	0	0	0	0	0	0	0
	15	I_{xx}	0	0	1.3755	0	0	0	0	0	0	0	0	0
	16	I_{xy}	0	0	0	1.0627	0	0	0	0	0	0	0	0
	17	I_{xz}	0	0	0	0	0.9735	0	0	0	0	0	0	0
	19	$I_{y}z$	0	0	0	0	0	0.8861	0	0	0	0	0	0
	20	I_{zz}	0	0	0	0	0	0	0.7574	0	0	0	0	0
	61	f_1	0	0	0	0	0	0	0	0.6295	0	0	0	0
	62	f_2	0	0	0	0	0	0	0	0	0.4606	0	0	0
	63	f_3	0	0	0	0	0	0	0	0	0	0.3673	0	0
	64	f_4	0	0	0	0	0	0	0	0	0	0	0.1751	0
	65	I_m	0	0	0	0	0	0	0	0	0	0	0	1.15E-16
V	12	mP_x	0.0662	0.4319	-0.077	0.4367	-0.1928	0.0309	-0.4601	0.4576	-0.3493	-0.1423	-0.1043	0
	13	mP_y	0.528	0.0052	0.0411	0.013	0.0136	0.002	-0.2685	0.0935	0.2735	0.2362	0.7125	7.22E-16
	15	I_{xx}	-0.0254	-0.3267	0.1676	-0.5606	0.3815	0.1256	-0.3665	0.3981	-0.2928	-0.0967	-0.0319	9.96E-17
	16	I_{xy}	-0.4229	-0.0017	-0.0172	-0.0448	-0.1181	0.1145	-0.7078	-0.4626	0.2751	-0.0308	0.0157	1.98E-16
	17	I_{xz}	-0.0355	0.2506	0.238	-0.3055	-0.1477	-0.8693	-0.0864	-0.0116	-0.0303	0.0057	0.0003	-8.37E-16
	19	$I_{y}z$	-0.0924	-0.0885	-0.0035	0.4957	0.7856	-0.3221	-0.0924	-0.0708	0.055	0.0189	-0.0067	-2.44E-16
	20	I_{zz}	-0.0153	0.5584	-0.0663	-0.2435	0.2835	0.1729	0.0728	-0.0719	0.0531	0.0182	0.0201	-0.7071
	61	f_1	-0.3428	-0.0481	-0.5192	-0.0547	-0.0207	-0.1039	0.0719	-0.1208	-0.5559	0.2106	0.4725	-1.95E-16
	62	f_2	-0.3016	-0.0415	-0.5067	-0.0902	-0.0246	-0.1343	0.0454	0.5468	0.5665	-0.0422	-0.0026	3.23E-16
	63	f_3	0.405	-0.0612	-0.4304	-0.1158	0.0245	-0.0883	-0.2091	-0.1258	-0.0349	0.5553	-0.506	-1.64E-15
	64	f_4	0.3932	-0.0681	-0.4325	-0.1027	0.0428	-0.1199	-0.0578	-0.2539	-0.0313	-0.747	0.0066	7.77E-16
	65	I_m	-0.0153	0.5584	-0.0663	-0.2435	0.2835	0.1729	0.0728	-0.0719	0.0531	0.0182	0.0201	0.7071

Table 2: The values of $\mathbf{r}_{1,diag}$ from QR results

	Items ↓↘	Link →	1	2	3	4	5	6
Link 1	1	m	0	0	0	0	0	0
	2	mP_x	0	0	0	0	0	0
	3	mP_y	0	0	0	0	0	0
	4	mP_z	0	0	0	0	0	0
	5	I_{xx}	0	0	0	0	0	0
	6	I_{xy}	0	0	0	0	0	0
	7	I_{xz}	0	0	0	0	0	0
	8	I_{yy}	0	0	0	0	0	0
	9	I_{yz}	0	0	0	0	0	0
	10	I_{zz}	-228.6174	0	0	0	0	0
Link 2	11	m	-4.52E-16	0	0	0	0	0
	12	mP_x	9.8627	253.4396	0	0	0	0
	13	mP_y	2.4353	593.2077	0	0	0	0
	14	mP_z	2.52E-14	0	0	0	0	0
	15	I_{xx}	2.7684	-56.3891	0	0	0	0
	16	I_{xy}	23.0138	-97.8936	0	0	0	0
	17	I_{xz}	130.5884	-211.0123	0	0	0	0
	18	I_{yy}	3.12E-14	5.69E-14	0	0	0	0
	19	I_{yz}	62.9039	81.4132	0	0	0	0
	20	I_{zz}	-1.98E-15	-122.6476	0	0	0	0
Link 3	21	m	-2.07E-15	5.08E-14	0	0	0	0
	22	mP_x	37.9513	-334.7127	467.3072	0	0	0
	23	mP_y	-33.2534	354.3488	455.3911	0	0	0
	24	mP_z	-4.90E-14	5.30E-14	0	0	0	0
	25	I_{xx}	-100.5171	-47.7176	-66.8431	0	0	0
	26	I_{xy}	174.542	103.1597	112.4557	0	0	0
	27	I_{xz}	253.2117	-99.1836	-164.7055	0	0	0
	28	I_{yy}	6.53E-14	-5.46E-15	-3.16E-14	0	0	0
	29	I_{yz}	-248.5508	101.679	152.7053	0	0	0
	30	I_{zz}	-1.28E-14	-206.4943	-281.2688	0	0	0
Link 4	31	m	2.70E-14	-8.97E-14	-7.50E-14	0	0	0
	32	mP_x	-108.4773	-386.0355	-286.5489	-398.1636	0	0
	33	mP_y	-123.4907	-472.5241	374.8688	-335.879	0	0
	34	mP_z	-7.44E-14	-2.67E-13	-2.09E-13	0	0	0
	35	I_{xx}	-144.5376	-187.3428	-197.3844	-107.9234	0	0
	36	I_{xy}	280.2405	-365.3538	-383.3898	190.8098	0	0
	37	I_{xz}	108.9514	-162.0417	-247.1844	239.729	0	0
	38	I_{yy}	1.22E-13	2.17E-13	-1.50E-13	6.28E-14	0	0
	39	I_{yz}	94.7208	224.0334	274.2623	-246.0824	0	0
	40	I_{zz}	-228.3691	78.9371	-108.739	-431.3956	0	0
Link 5	41	m	-1.62E-14	5.48E-14	3.45E-14	0	0	0
	42	mP_x	110.3607	407.323	315.8836	65.6913	396.6561	0
	43	mP_y	126.9043	-404.1635	-281.1052	208.6213	394.49	0
	44	mP_z	6.67E-14	-3.01E-13	9.77E-14	1.77E-13	0	0
	45	I_{xx}	-168.5834	-171.4806	-186.6335	212.6078	201.7761	0
	46	I_{xy}	389.4686	-306.1781	-318.4133	513.4559	-399.6391	0
	47	I_{xz}	289.3772	-315.7677	-322.641	363.8805	-411.4982	0
	48	I_{yy}	1.91E-13	-1.86E-13	1.53E-13	1.59E-13	-1.91E-14	0
	49	I_{yz}	319.6101	334.8672	341.2031	368.3976	468.6476	0
	50	I_{zz}	161.8988	-205.6505	-215.2295	142.2798	-463.0538	0
Link 6	51	m	2.08E-14	-6.87E-14	-4.33E-14	2.36E-14	7.80E-15	0
	52	mP_x	426.2574	1068.3319	699.9122	295.3725	292.5512	417.3724
	53	mP_y	-434.786	-1079.0873	-704.9352	303.1425	-347.0608	407.0301
	54	mP_z	-1.11E-13	-2.31E-13	-2.41E-13	-6.23E-14	-1.29E-13	0
	55	I_{xx}	339.3361	-408.1584	-413.338	-238.2624	-476.7573	190.3695
	56	I_{xy}	679.9189	824.2461	829.0289	-506.5916	913.3395	-362.6816
	57	I_{xz}	452.786	-637.3979	-635.4993	617.0631	-1038.9653	445.405
	58	I_{yy}	2.44E-13	-3.99E-13	3.25E-13	2.28E-13	3.06E-13	-1.21E-13
	59	I_{yz}	507.4218	593.0612	551.2006	-655.9884	1109.9515	-479.4588
	60	I_{zz}	320.3927	293.4021	310.0073	558.9444	279.1904	-875.2979

Table 3: Selected Dynamic Parameters

		The Dynamic Parameters														
		m	mP_x	mP_y	mP_z	I_{xx}	I_{xy}	I_{xz}	I_{yy}	I_{yz}	I_{zz}	f_1	f_2	f_3	f_4	I_m
Joints	1										✓	✓	✓	✓	✓	
	2		✓	✓		✓	✓	✓		✓	✓	✓	✓	✓	✓	
	3		✓	✓		✓	✓	✓		✓	✓	✓	✓	✓	✓	✓
	4		✓	✓		✓	✓	✓		✓	✓	✓	✓	✓	✓	✓
	5		✓	✓		✓	✓	✓		✓	✓	✓	✓	✓	✓	✓
	6		✓	✓		✓	✓	✓		✓	✓	✓	✓	✓	✓	✓

2

ROBOT DYNAMICS MODEL CALIBRATION

Robot dynamics models require accurate knowledge of the target robot's dynamics parameters to be properly adopted. However, they are not always known or measurable.

Recent techniques such as CAD modeling allow the computation of inertial parameters of the robot's parts (links, actuators, and transmissions) based on their geometry and material mechanical properties. Unfortunately, the estimation is often inaccurate due to the simplification adopted by geometric modeling. Moreover, complex dynamics effects such as joint friction, also depend on unpredictable factors and effects, such as (i) the environment temperature, (ii) the internal joint temperature, and (iii) the joints transmission wearing. As a result, to improve the quality of the estimation experimental campaigns on the real robot must be performed to actually measure the parameters and the external effects affecting them. To do so, exciting trajectories are employed to provide optimal excitation of all dynamics phenomena.

Estimation techniques involve rigid body models linearized on a set of minimal lumped parameters estimated thanks to the adoption of exciting trajectories. These are designed to excite certain parameters and allow their precise estimation.

Furthermore, once the dynamics model is obtained and reduced as explained in Chapter 1, dynamics calibration procedures should be performed. Dynamics calibration exploits identification techniques for the estimation of inertial parameters of the arm (mass, first momentum, and links' inertia), and the joints friction contribution [22, 23]. It is worth noting that the goal of dynamics calibration is not to estimate the full set of parameters (masses, inertias, friction coefficients) but to minimize the prediction error of the model with respect to the real behavior of the robot.

The robot needs to be sufficiently excited by external signals to reach optimal estimation performance. This is done in two steps:

- joints trajectories are designed considering specific optimization criteria
- the trajectories are evaluated considering their exciting effect on the manipulator mechanical structure.

Several algorithms and procedures focus on the optimization of trajectories that homogeneously excite the model's parameters to obtain a robust, over-constrained, and well-conditioned linear system [25, 35, 36, 37, 38]. Accurate estimation of

torques is based on the conditioning properties of the resulting kinematics function (regressor) that maps the model parameters into torques.

A plethora of examples about the design of exciting trajectories can be found in the literature. In [2] a method to excite the dynamics parameters by means of 5-th order polynomials is presented, while the method in [39] adopts splines instead. A combination of cosine and ramp is adopted in the method shown in [40], and a method adopting a finite sum of harmonic sine and cosine is used in [37, 38, 41].

The choice of the exciting trajectories influences the dynamics calibration because the resulting parameters are polarized on the specific class of exciting trajectories used. However, the abovementioned trajectories are different from those commonly used by IRs motion planners. Hence, estimation algorithms should take into account the adoption of the robot motion planner trajectories instead of other methods.

The well-conditioning of exciting trajectories has been often expressed through the condition number and the singular values of robot dynamics regressors (see Equation (2)) [25, 36, 35, 42]. In publications [43, 44] these properties have been soft-coded into cost functions used in evolutionary-based techniques.

The experimental estimation of the dynamics parameters requires the pseudo-inversion of the dynamics regressor (i. e. a matrix that maps the dynamics parameters into the joint torques). Different techniques have been proposed by the literature, such as the Ordinary Least Square method (OLS), or its extension called Weighted Least Square algorithm (WLS) as in [45]. There, the error covariance matrix is used to weight the regression matrix and measured torques. Authors of [37] obtain the parameters using a method based on the maximum-likelihood estimation. Another solution, which is commonly adopted by the automatic control research community, is the use of the Kalmann filtering algorithm (KF) as in [43]. The work in [46] shows a comparison between WLS and KF methods, suggesting that with the adoption of the KF method the identification process complexity increases without producing useful improvements. Furthermore, KF is very sensitive to initial conditions.

An alternative to these methods is the instrumental variable (IV) method described in [47], which is robust to data filtering and is statistically optimal. The approach gets remarkable results when well-conditioned trajectories are applied; however, their drawback is that such trajectories have to span the whole robot's workspace, resulting in an averagely fitting model.

2.1 DYNAMIC PARAMETER IDENTIFICATION

Making use of rigid multi-bodies dynamics, as already described in Chapter 1, the robot dynamics can be reduced to:

$$\tau = \begin{bmatrix} Y_d & Y_f & Y_m \end{bmatrix} \begin{bmatrix} \pi_d \\ \pi_f \\ \pi_m \end{bmatrix} = Y(q, \dot{q}, \ddot{q})\pi \tag{53}$$

where π is the compound set of dynamics model parameters and Y is the compound regressor. Several techniques are known [48, 12] for the direct solution of the system in (53). Among others, OLS and WLS techniques [45] appear to be the most common. OLS is a method of regression analysis used to obtain the optimal solution based on given data and equations. This method adopts a linear system described as:

$$y(t) = \varphi(t)\vartheta \tag{54}$$

where $y(t)$ and $\varphi(t)$ are the vectors of system output and input at time t respectively, and ϑ is the vector of the system's parameters to identify. The OLS procedure starts with the collected data of y and φ with the time series $\bar{t}$ (from 1 to k), then it uses the estimated parameters ϑ^*. The difference between the estimated results and desired results is the following error function:

$$\varepsilon = y - \varphi\vartheta^* \tag{55}$$

By minimizing j, which is the sum of squared residuals as shown in Equation (56), it is possible to find the optimal parameters ϑ^*.

$$j = \sum_{i=1}^{k} [\varepsilon(i)]^2 \tag{56}$$

In this process, the minimized j can be obtained by setting the partial derivative to zero as follows:

$$\left.\frac{\partial j}{\partial \vartheta}\right|_{\vartheta=\vartheta^*} = 0 \tag{57}$$

Then:

$$\begin{aligned} j &= \sum_{i=1}^{k} [\varepsilon(i)]^2 = \varepsilon^T\varepsilon \\ &= [y - \varphi\vartheta^*]^T [y - \varphi\vartheta^*] \\ &= y^T y - 2(\vartheta^*)^T\varphi^T y + (\vartheta^*)^T\varphi^T\vartheta^* \end{aligned} \tag{58}$$

$$\begin{aligned}\left.\frac{\partial j}{\partial \vartheta}\right|_{\vartheta=\vartheta^*} = 0 \Rightarrow \left.\frac{\partial j}{\partial \vartheta}\right|_{\vartheta=\vartheta^*} &= -2\boldsymbol{\varphi}^T\mathbf{y} + 2\boldsymbol{\varphi}^T\boldsymbol{\varphi}\vartheta^* = 0 \\ \Rightarrow \vartheta^* &= \left[\boldsymbol{\varphi}^T\boldsymbol{\varphi}\right]^{-1}\boldsymbol{\varphi}^T\mathbf{y}\end{aligned} \tag{59}$$

ϑ^* is obtained finally under the condition of $\boldsymbol{\varphi}^T\boldsymbol{\varphi} \neq 0$.

For a given trajectory $(\mathbf{q}, \dot{\mathbf{q}}, \ddot{\mathbf{q}})$ composed by n-samples, the estimation of the dynamic parameters $\hat{\boldsymbol{\pi}}$ can be obtained from the given data $\hat{\mathbf{Y}}$ using the OLS method. The data can be reorganized as a matrix with series points, which are:

$$\hat{\mathbf{Y}}_{\text{pt 1}}, \hat{\mathbf{Y}}_{\text{pt 2}}, \cdots, \hat{\mathbf{Y}}_{\text{pt n}}$$

For example, the information matrix $\mathbf{Y}$ and the estimated parameters $\hat{\boldsymbol{\pi}}$ of Joint 1, based on Equation (42), can be determined as follows:

$$\mathbf{Y}_{\text{Joint 1}} = \begin{bmatrix} \mathbf{u}_{11,\text{pt 1}} & \mathbf{u}_{12,\text{pt 1}} & \mathbf{u}_{13,\text{pt 1}} & \mathbf{u}_{14,\text{pt 1}} & \mathbf{u}_{15,\text{pt 1}} & \mathbf{u}_{16,\text{pt 1}} \\ \mathbf{u}_{11,\text{pt 2}} & \mathbf{u}_{12,\text{pt 2}} & \mathbf{u}_{13,\text{pt 2}} & \mathbf{u}_{14,\text{pt 2}} & \mathbf{u}_{15,\text{pt 2}} & \mathbf{u}_{16,\text{pt 2}} \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ \mathbf{u}_{11,\text{pt n}} & \mathbf{u}_{12,\text{pt n}} & \mathbf{u}_{13,\text{pt n}} & \mathbf{u}_{14,\text{pt n}} & \mathbf{u}_{15,\text{pt n}} & \mathbf{u}_{16,\text{pt n}} \end{bmatrix}_{n*60}$$

$$\mathbf{Y}_{\text{f, Joint 1}} = \begin{bmatrix} \mathbf{y}_{\text{f,1,pt 1}} & 0_{1*4} & 0_{1*4} & 0_{1*4} & 0_{1*4} & 0_{1*4} \\ \mathbf{y}_{\text{f,1,pt 2}} & 0_{1*4} & 0_{1*4} & 0_{1*4} & 0_{1*4} & 0_{1*4} \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ \mathbf{y}_{\text{f,1,pt n}} & 0_{1*4} & 0_{1*4} & 0_{1*4} & 0_{1*4} & 0_{1*4} \end{bmatrix}_{n*24}$$

$$\mathbf{Y}_{\text{m, Joint 1}} = \begin{bmatrix} y_{\text{m,1,pt 1}} & 0 & 0 & 0 & 0 & 0 \\ y_{\text{m,1,pt 2}} & 0 & 0 & 0 & 0 & 0 \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ y_{\text{m,1,pt n}} & 0 & 0 & 0 & 0 & 0 \end{bmatrix}_{n*6}$$

$$\hat{\boldsymbol{\pi}}_{\text{Joint 1}} = \begin{bmatrix} \boldsymbol{\pi}_1 & \boldsymbol{\pi}_2 & \boldsymbol{\pi}_3 & \boldsymbol{\pi}_4 & \boldsymbol{\pi}_5 & \boldsymbol{\pi}_6 \end{bmatrix}^T_{1*60}$$

$$\hat{\boldsymbol{\pi}}_{\text{f, Joint 1}} = \begin{bmatrix} \boldsymbol{\pi}_{\text{f,1}} & \boldsymbol{\pi}_{\text{f,2}} & \boldsymbol{\pi}_{\text{f,3}} & \boldsymbol{\pi}_{\text{f,4}} & \boldsymbol{\pi}_{\text{f,5}} & \boldsymbol{\pi}_{\text{f,6}} \end{bmatrix}^T_{1*24}$$

$$\hat{\boldsymbol{\pi}}_{\text{m, Joint 1}} = \begin{bmatrix} \pi_{\text{m,1}} & \pi_{\text{m,2}} & \pi_{\text{m,3}} & \pi_{\text{m,4}} & \pi_{\text{m,5}} & \pi_{\text{m,6}} \end{bmatrix}^T_{1*6}$$

And then the equation used in identification can be rewritten as:

$$[\boldsymbol{\tau}_{\text{Joint 1}}]_{n*1} = \begin{bmatrix} \tau_{\text{Joint 1, pt 1}} \\ \tau_{\text{Joint 1, pt 2}} \\ \vdots \\ \tau_{\text{Joint 1, pt n}} \end{bmatrix} = \mathbf{Y}_{\text{Joint 1, n*90}}\hat{\boldsymbol{\Pi}}_{\text{Joint 1, 90*1}} \tag{60}$$

$$\Rightarrow \quad \hat{\Pi}_{\text{Joint 1}} = (Y^T_{\text{Joint 1}} Y_{\text{Joint 1}})^{-1} Y^T_{\text{Joint 1}} [\tau_{\text{Joint 1}}] \tag{61}$$

It is possible to apply the same procedure on the other joints to provide the best regression conditions for the system in (53), thus finding the optimum estimate $\hat{\pi}$.

2.2 DESIGN OF OPTIMAL EXCITING TRAJECTORIES

To improve the numerical efficiency of the OLS algorithm, the regression matrix should have a small condition number and a large minimum singular value. Therefore, a small sensitivity of the estimation algorithm to (i) measurement noise and (ii) unmodelled dynamics can be obtained if a suitable cost function is minimized. The aim is to find the n measurement points able to minimize the condition number and maximize the minimum singular value [25]. Another frequently used index is the determinant of the quadratic form of dynamics regressor $D = \log_{10}[\det(Y^T Y)]$. Maximization of index D with respect to the input excitation is referred to as D-optimality and it is desirable in order to get estimates with minimal uncertainty bounds [49]. The minimization or the maximization of such indexes is obtained by designing trajectories following a trial-and-error approach or through optimization algorithms such as genetic algorithms or non-linear programming algorithms.

Test trajectories have been performed on the robots described in 3 to obtain the data needed for the dynamic parameter identification and to observe the related influences in the long-duration operation. Since friction elements are the only factors that may be affected by temperature, the test trajectory has to be appropriately short to avoid a steep temperature rise. In this thesis work, a warm-up trajectory is performed in-between each motion to better understand the influence of friction on the joints. Therefore, the complete test trajectory includes three sub-trajectories with different aims:

1. the excitation trajectory for the parameters identification (stage 1)
2. the friction trajectory to understand the friction behavior (stage 2)
3. the warming trajectory to warm up the robot (stage 3).

An example of one cycle of the full test trajectory is shown in Figure 1.

The excitation trajectory used in the first stage is specifically designed to excite the model parameters with a high range of velocities between zero and the maximum robot speed. The second stage's trajectory is designed for friction measurement and estimation. Finally, the warm-up trajectory in the third stage is designed to prepare the robot for the next cycle. The complete test trajectory adopted in this thesis work has 24 cycles and lasts about 2 hours.

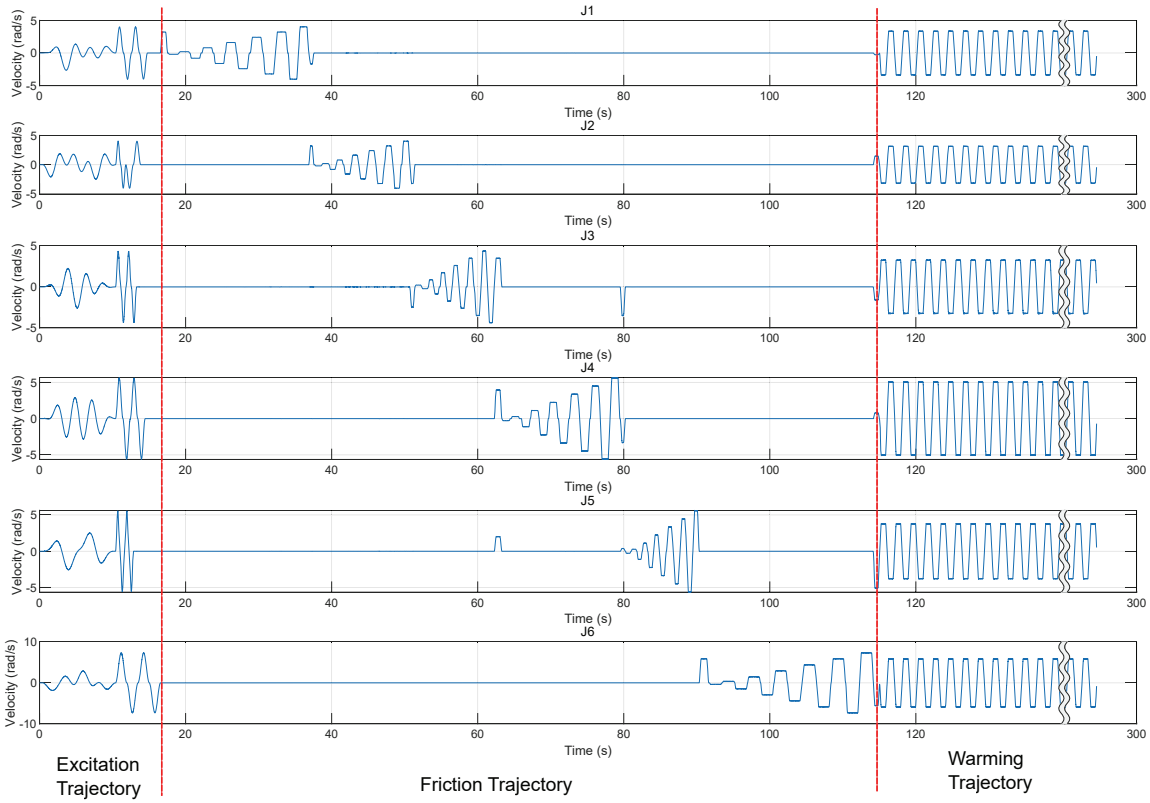


Figure 1: The velocity of one cycle in full test trajectory

2.3 EXCITATION TRAJECTORIES GENERATION

There are two methods to create an appropriate excitation trajectory. The first one is a finite sum of N harmonic sine function. This method creates a high and sudden acceleration at the start of the trajectory, which results in a non-smooth movement. The second one is a finite Fourier series formula [32, 50, 51, 52, 53]. To obtain the optimized excitation trajectory, it is important to select an appropriate cost function. According to the literature, there are several options which are (i) the condition number of the information matrix [50, 51, 54], (ii) the determinant of the information matrix [51, 52, 55], and (iii) a specific formula that adopts both the condition number and the determinant with different weights [56, 57, 58].

In this thesis, the excitation trajectory has been obtained and optimized by minimizing the condition number and the reciprocal of the determinant of the information matrix. Moreover, two different algorithms have been considered to optimize the excitation trajectory: (i) the genetic algorithm "ga" [59] and (ii) the constrained nonlinear multivariable function "fmincon" [60]. Both are functions provided by the optimization toolbox in MATLAB.

2.3.1 *First Trajectory*

The first excitation trajectory used is created by the finite sum of N harmonic sine function:

$$q(t) = \sum_{k=1}^{N} a_k \sin(k\omega_f t) \tag{62}$$

where:

- t is the time sequence
- $q(t)$ is the joint angle at time t
- ω_f is the base frequency
- N is the number of elements introduced in the calculation.

The boundaries applied in Equation (62) are:

$$\begin{aligned} q_{min} \leqslant & q(t) \leqslant q_{max} \\ \dot{q}_{min} \leqslant & \dot{q}(t) \leqslant \dot{q}_{max} \\ \ddot{q}_{min} \leqslant & \ddot{q}(t) \leqslant \ddot{q}_{max} \end{aligned} \tag{63}$$

The upper limit of the Equation (62) is:

$$|q(t)| = \left| \sum_{k=1}^{N} a_k \sin(k\omega_f t) \right| \leqslant \sum_{k=1}^{N} |a_k \sin(k\omega_f t)| \leqslant \sum_{k=1}^{N} |a_k| \leqslant N a_{k,\,max} \tag{64}$$

Thanks to the limitation of Equation (63), it is possible to write:

$$a_{k,\,max} \leqslant \frac{q_{max}}{N} \tag{65}$$

Similarly, for the angle velocity and acceleration, the resulting equations are:

$$\dot{q}(t) = \sum_{k=1}^{N} a_k k\omega_f \cos(k\omega_f t) \tag{66}$$

$$\begin{aligned} |\dot{q}(t)| = \left| \sum_{k=1}^{N} a_k k\omega_f \cos(k\omega_f t) \right| & \leqslant \sum_{k=1}^{N} |a_k k\omega_f \cos(k\omega_f t)| \leqslant \sum_{k=1}^{N} |a_k k\omega_f| \leqslant N a_{k,\,max} k\omega_f \\ = N \frac{q_{max}}{N} k\omega_f & = q_{max} k\omega_f \leqslant \dot{q}_{max} \\ \Rightarrow k\omega_f & \leqslant \frac{\dot{q}_{max}}{q_{max}} \end{aligned} \tag{67}$$

$$\ddot{q}(t) = -\sum_{k=1}^{N} a_k (k\omega_f)^2 \sin(k\omega_f t) \tag{68}$$

$$
\begin{aligned}
|\ddot{q}(t)| &= \left| -\sum_{k=1}^{N} a_k (k\omega_f)^2 \sin(k\omega_f t) \right| \leqslant \sum_{k=1}^{N} \left| a_k (k\omega_f)^2 \sin(k\omega_f t) \right| \leqslant \sum_{k=1}^{N} \left| a_k (k\omega_f)^2 \right| \\
&= N \frac{q_{max}}{N} (k\omega_f)^2 = q_{max} (k\omega_f)^2 \leqslant \ddot{q}_{max} \\
\Rightarrow k\omega_f &\leqslant \sqrt{\frac{\ddot{q}_{max}}{q_{max}}}
\end{aligned}
\tag{69}
$$

In conclusion, the limitation of a_k and $k\omega_f$ of Equation (62) are:

$$a_k \leqslant \frac{q_{max}}{N} \tag{70}$$

$$k\omega_f \leqslant \min\left(\frac{\dot{q}_{max}}{q_{max}}, \frac{\ddot{q}_{max}}{q_{max}} \right) \tag{71}$$

Then, by knowing the value of $q(t)$, $\dot{q}(t)$ and $\ddot{q}(t)$, the value of a_k and $k\omega_f$ can be computed.

Using $N = 3$, Equation (62) can be rewritten as follows:

$$q(t) = a_1 \sin(\omega t) + a_2 \sin(2\omega t) + a_3 \sin(4\omega t) \tag{72}$$

$$\dot{q}(t) = a_1 \omega \cos(\omega t) + 2a_2 \omega \cos(2\omega t) + 4a_3 \omega \cos(4\omega t) \tag{73}$$

$$\ddot{q}(t) = -\left[a_1 \omega^2 \cos(\omega t) + 4a_2 \omega^2 \cos(2\omega t) + 16a_3 \omega^2 \cos(4\omega t) \right] \tag{74}$$

where $a_1 = a_2 = a_3 = a$, $N = 3$, and ω is the abbreviation of ω_f;

$$|q(t)| \leqslant |a_1 \sin(\omega t)| + |a_2 \sin(2\omega t)| + |a_3 \sin(4\omega t)| \leqslant a_1 + a_2 + a_3 = Na \Rightarrow |q(t)| \leqslant Na \tag{75}$$

Applying the same reasoning for $\dot{q}(t)$ and $\ddot{q}(t)$, it is possible to write:

$$|\dot{q}(t)| \leqslant Na4\omega \tag{76}$$

$$|\ddot{q}(t)| \leqslant Na(4\omega)^2 \tag{77}$$

Using $k = 4$, a smaller value of ω is obtained. It is clear that increasing the value of ω, the related joint velocity, acceleration and output torque will rise too during the movement. Hence, it is important to find the optimized ω with the given limitation of $q_{min} \leqslant q(t) \leqslant q_{max}$, $\dot{q}_{min} \leqslant \dot{q}(t) \leqslant \dot{q}_{max}$ and $\ddot{q}_{min} \leqslant \ddot{q}(t) \leqslant \ddot{q}_{max}$. The values of a_1, a_2 and a_3 follow the same reasoning. In MATLAB, the functions "fmincon" and "ga" can easily provide the optimized values of ω, a_1, a_2, and a_3. The boundaries applied for the optimization are discussed and demonstrated in Equation (63).

There was an acceleration problem at the start of the experiments using this excitation trajectory, which led to an inaccurate parameters estimation. The reason was that the excitation trajectory was not smooth enough. Therefore, a second excitation trajectory has been developed.

2.3.2 *Second Excitation Trajectory*

To solve the issue experienced by using the first excitation trajectory, Equation (62) has been modified using a finite sum of harmonic sine and cosine functions, i.e., a finite Fourier series [37]:

$$\theta_i(t) = q_{i,0} + \sum_{l=1}^{N} \left[\frac{a_{l,i}}{w_f l} \sin(w_f l t) - \frac{b_{l,i}}{w_f l} \cos(w_f l t) \right] \tag{78}$$

where $\dot{\theta}_i(t) and \ddot{\theta}_i(t)$ are:

$$\dot{\theta}_i(t) = \sum_{l=1}^{N} [a_{l,i} \cos(w_f l t) + b_{l,i} \sin(w_f l t)] \tag{79}$$

$$\ddot{\theta}_i(t) = w_f \sum_{l=1}^{N} [-a_{l,i} \sin(w_f l t) + b_{l,i} \cos(w_f l t)] \tag{80}$$

where:

- $\theta_i(t), \dot{\theta}_i(t), \ddot{\theta}_i(t)$ are the joint position, velocity, and acceleration respectively
- the subscript i is the joint index
- t is the time sequence
- q_0 is the initial position of joint
- a and b are the elements of the excitation trajectory obtained from the optimization algorithms
- N is the total number of elements a or b
- w_f is the base frequency used to create the test trajectory in the optimization algorithm.

Based on [50, 51, 52, 54], N and w_f have been set to 5 and $2\pi * 0.1$Hz respectively. The boundaries applied on the MATLAB optimization functions are the same as in Equation (63).

2.3.3 *Excitation Trajectory Optimization*

The goal of the optimization procedure is to find the best trajectory that excites the robot parameters for identification. In this work, the optimization procedure has been performed using the functions "fmincon" and "ga" provided by MATLAB. As mentioned at the beginning of Section 2.3, the optimization has been applied using the condition number and the determinant of the regression matrix. Subsequently, the results of the two optimization functions are compared to choose the one that provides the highest velocities in each joint without exceeding the boundaries. In the following discussion, one excitation trajectory optimized

with the genetic algorithm minimizing the condition number of the regression matrix is implemented. A low value of the condition number indicates (i) that the maximum speed and acceleration are reached without exceeding the joint position limitations, and (ii) it also provides large torque output values. It is worth noting that the robot spans the whole workspace during the movement to provide valuable information on each area of its working space.

The genetic algorithm approach results in the best estimation of the dynamics parameters. The Cartesian path performed during the execution of the trajectory is shown in Figure 2. However, some errors occurred during the tests resulting

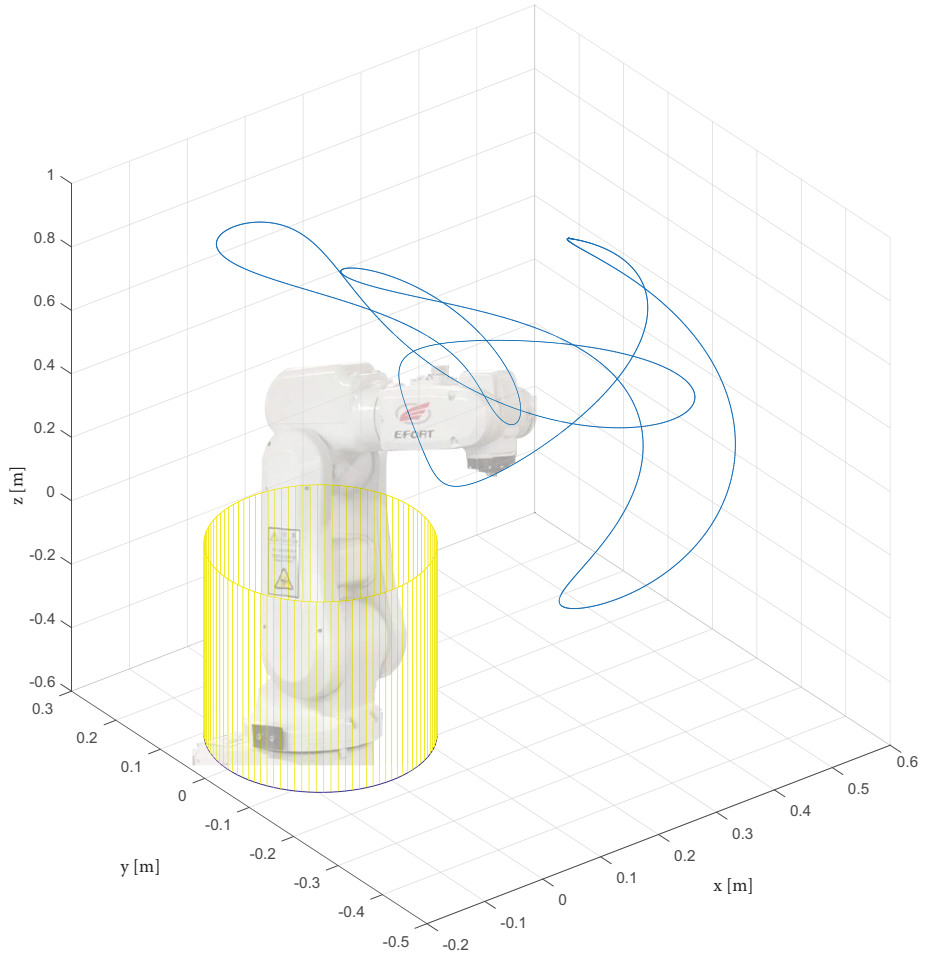


Figure 2: Robot movement performing the excitation trajectory.

in incorrect values of the estimated torque, which are marked in Figure 3. It has been experimentally found that these errors occurred in the path when the joint velocity exceeds the maximum velocity of the test excitation trajectory. Due to this issue, the resulting estimation in said portion of the test trajectory is significantly abnormal compared to the measurement data.

Another possible reason depends on the robot characteristics. The maximum speed of each joint set in the algorithm has been based on the robot's mechanical characteristics and specifications according to the vendor's datasheet provided. However, during the tests adopting the computed optimal trajectory, it has been found that the joints moved at a speed slower than the maximum. Therefore, the

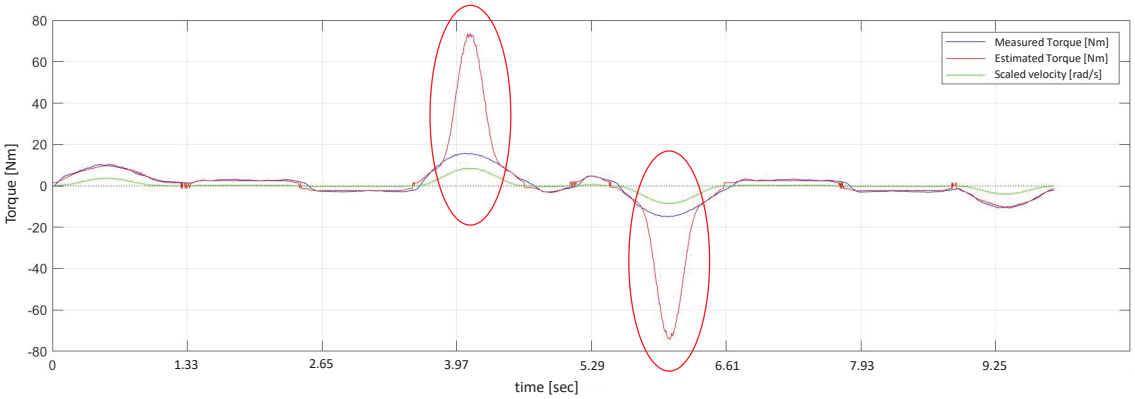


Figure 3: The comparison between the measured torque and the estimated torque with the second excitation trajectory. Experimental data, Joint 5.

excitation trajectory are the following [61]:

$$\theta = \theta_0 + \Delta\theta \left[\frac{t}{T} - \frac{1}{2\pi} \sin\left(\frac{2\pi t}{T}\right) \right] \tag{81}$$

$$\dot{\theta} = \frac{\Delta\theta}{T} \left[1 - \cos\left(\frac{2\pi t}{T}\right) \right] \tag{82}$$

$$\ddot{\theta} = \frac{2\pi\Delta\theta}{T^2} \sin\left(\frac{2\pi t}{T}\right) \tag{83}$$

where:

θ, $\dot{\theta}$ and $\ddot{\theta}$ are the position, velocity, and acceleration of the joints during the additional excitation trajectory

θ_0 is the joint initial position (set at 0 in this case)

t and T are the time variable and the overall duration of the test, respectively

$\Delta\theta$ is the maximum position range of the moving joint. The robot starts and ends at the same position with a speed set at zero for all joints. The trajectory is executed in both positive and negative directions. Furthermore, this additional trajectory is performed after the second excitation trajectory. The complete merged excitation trajectory is shown in Figure 5. It should be noted that some small movements have been added for some joints due to the different duration of the generated trajectories.

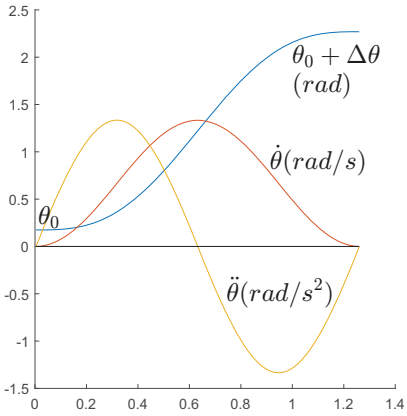


Figure 4: The position, velocity and acceleration of the additional trajectory

2.4 FRICTION TRAJECTORY

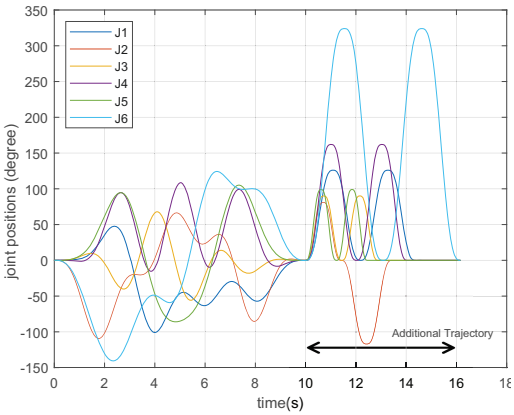
As already told in Section 2.2, the second stage of the complete test trajectory is designed to highlight the friction behavior. This phenomenon will be explained in detail in Section 4.1.1, however, for the sake of clarity, the basic concept behind its design is explained in this section as well.

To measure the friction torque, which depends on speed and temperature, a special trajectory with 6 trapezoidal velocities has been designed, as shown in the central section of Figure 1). There, each segment includes periods with a constant velocity at 5%, 20%, 40%, 60%, 80%, 100% of the maximum joint velocity. The acceleration parts of the trajectory are ignored to avoid dynamics effects. The influence of weight in the constant velocity periods has been removed using the technique explained in detail in Chapter 4. To obtain the same number of data for each speed measurement, all periods have been set equal and, consequently, the amplitude of the motion is bigger at higher velocities. Both positive and negative velocities are included.

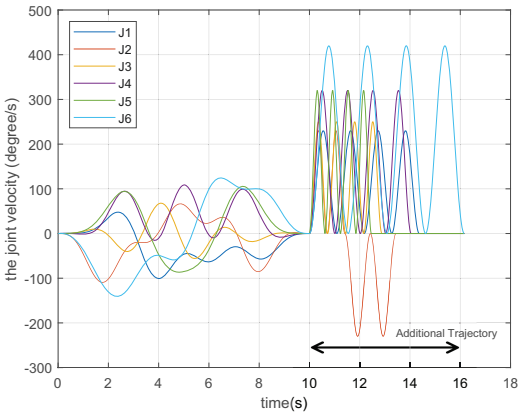
It is worth noting that the acquired torque values contain the gravitational component for some joints. To eliminate its effect, the robot (see Chapter 3) was put into specific configurations so that the rotation axes of joints 1, 4, 5, and 6 are vertical. For the remaining joints, which are Joint 2 and 3, the gravity effect has been removed using the OLS procedure. During the motion at a constant velocity of an individual joint (see Figure 6), the motor torque is:

$$\boldsymbol{\tau}_{no-gravity} = \boldsymbol{\tau}_{motor} - \overrightarrow{\mathbf{P}}\mathbf{y}\cos(\theta) - \overrightarrow{\mathbf{P}}\mathbf{x}\sin(\theta) \tag{84}$$

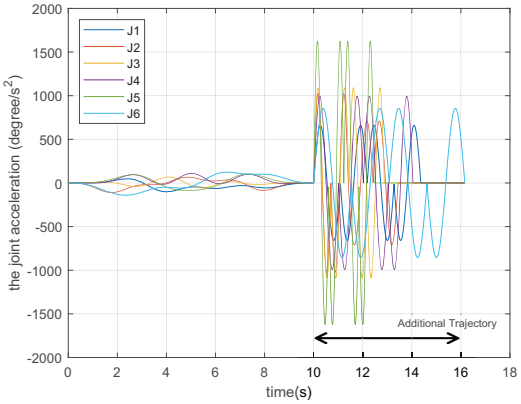
where $\overrightarrow{\mathbf{P}}\mathbf{x}$ and $\overrightarrow{\mathbf{P}}\mathbf{y}$ are the parameters to identify. Once these parameters have been computed, the gravity effects can be subtracted to obtain only the friction component. Examples of the original data for Joints 2 and 3 are shown in Figure 7a). Figure 7 shows the same data after removing the gravity effect. The different



(a) Position



(b) Velocity



(c) Acceleration

Figure 5: Excitation trajectory including the additional section

curves represent the torque at different joint's temperature. An example of the resulting friction measurements is showin in Figure 8.

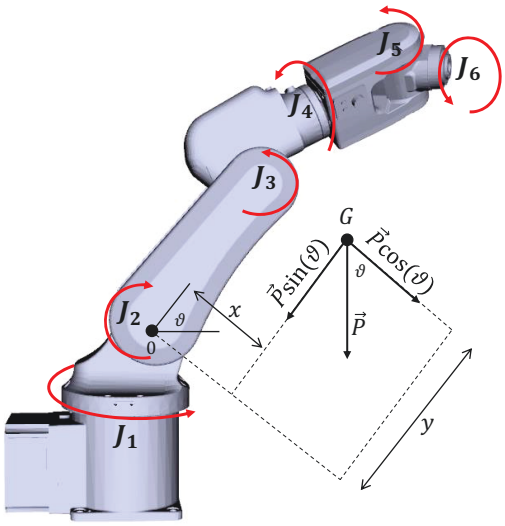


Figure 6: Principle used for the simplified model of (84) when just one joint is moved at each time and the others assume a predefined value (example of Joint 2). The robot in the figure is the 3D model of the manipulator used in this work.

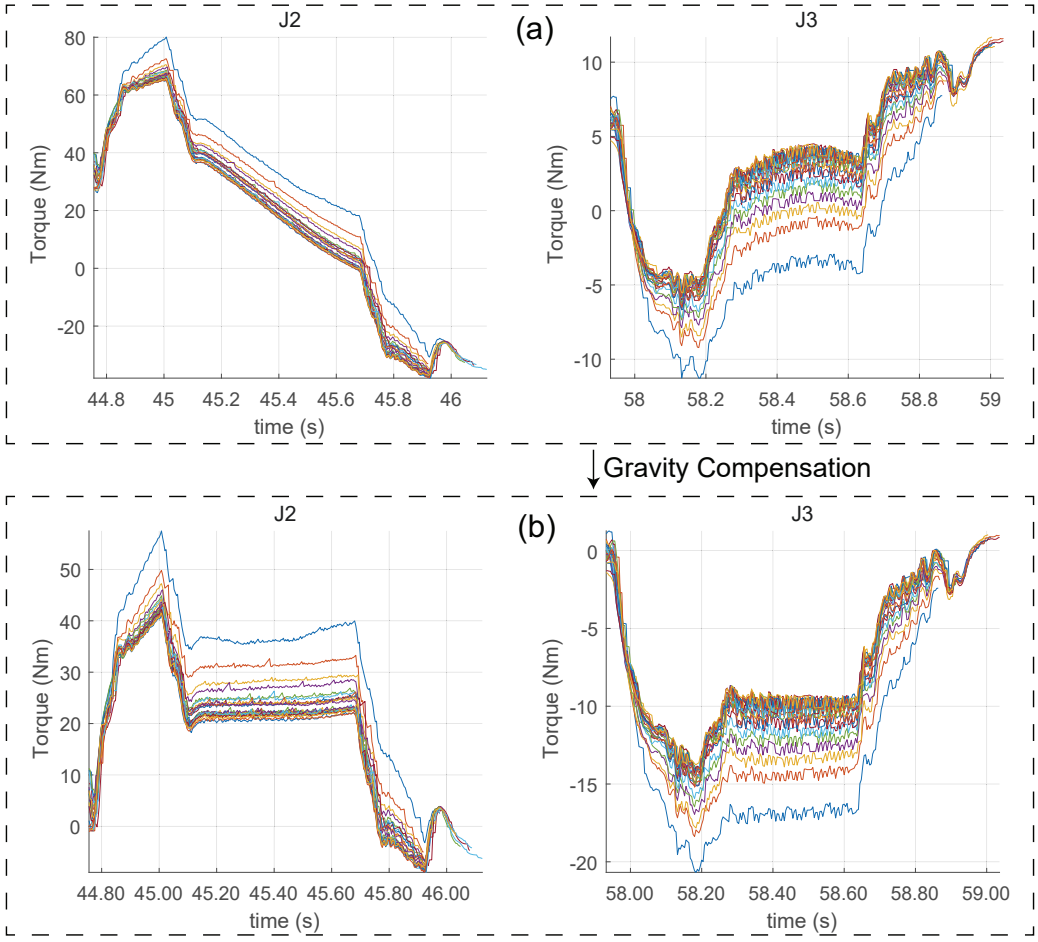


Figure 7: Experimental data. (**a**) The torque motor output of Joints 2 and 3 during the friction measure cycle. Data collected from one test on "Robot 1" (60% of the velocity). (**b**) The same data without the gravity effect.

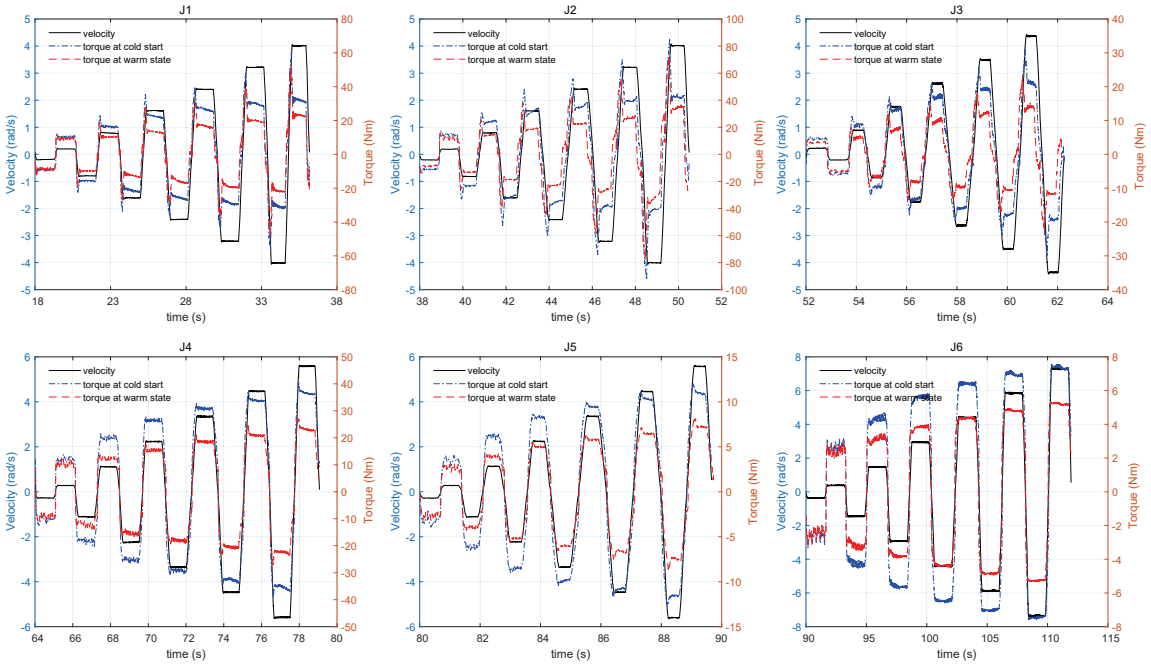


Figure 8: Velocity and torque versus time of the experimental data in the friction measure stage considering both cold and warm conditions, from Joint 1 to 6

2.4.1 *Warming Trajectory*

The warming stage of the trajectory has been designed to increase the internal temperature of the robot joints. Temperature is one of the major external causes that produce observable fluctuations of friction values. During the warming stage, all the joints move at their maximum speed simultaneously, as shown in the trajectory in Figure 1 in the "Warming Trajectory" section. In the experiment, all joints are configured using 80% of their angular limitations at 100% velocity. To capture the friction changes and avoid excessive temperature rising, the duration of this stage has been carefully chosen. Based on the studies in [62] [63], and on the results of the experimental tests, the optimal duration of the warming stage has been set to 3 min.

Therefore, the overall duration of the test trajectory is set to 5 min per cycle. The cycle has been repeated 24 times obtaining an experiment lasting 2 hours. To ensure the same condition at the start of each experiment, it is important to let the robot cool down after each experiment.

3

EXPERIMENTAL SETUP

3.1 ROBOT CHARACTERISTICS

Two EFORT industrial robots have been adopted for the experiments. EFORT (Efort Intelligent Equipment [64]) is one of the largest industrial robots manufacturers of China, providing customers with a full range of products and cross-industry intelligent manufacturing solutions.

The model adopted is an ER3A-C60 6-axis anthropomorphic robot with a payload of 3 kg. There are a total of six servo-motors actuating the six revolute joints of the robot, therefore allowing to fully control both the position and orientation of the end-effector. In Figure 9 is shown the mechanical structure of the robot and the arrangement of the six joints (named J1, J2, J3, J4, J5, and J6), indicating

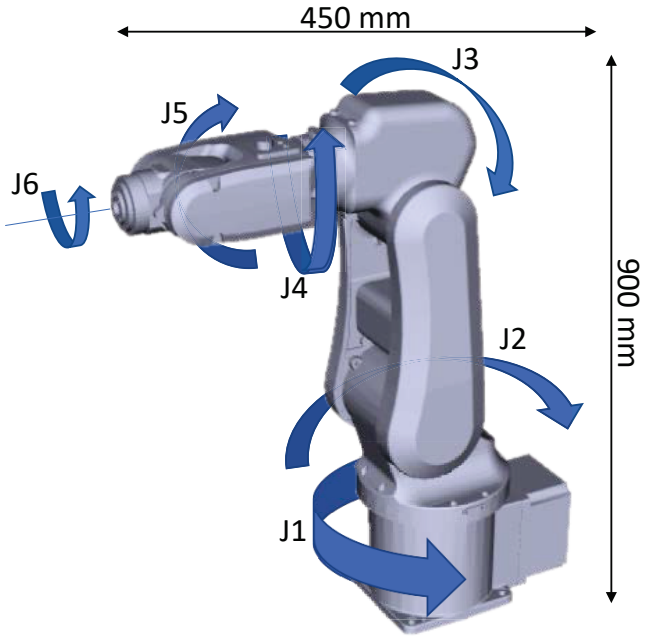


Figure 9: ER3A-C60 mechanical structure

According to the robot maintenance manual, the robot may be mounted at any angle with respect to the ground, even upside-down. The two main links of the robot have a length of 340 mm and 377 mm, allowing for a maximum reach of

Table 4: ER3A-C60 main characteristics and performance parameters

Robot Model		ER3A-C60
Degrees of freedom		6
Drive mode		AC servo drive
Maximum operating speed	J1	230°/s
	J2	230°/s
	J3	250°/s
	J4	320°/s
	J5	320°s
	J6	420°/s
Maximum operating range	J1	±167°
	J2	+90/-130°
	J3	+101°/-71°
	J4	±180°
	J5	±113°
	J6	±360°
Wrist maximum load		3 kg
Maximum operating radius		628 mm
Repeat positioning accuracy		±2 mm
Ambient temperature		0-45°
Robot mass		27 kg

628 mm with respect to the base. Figure 10 outlines the robot parts' dimension and the robot workspace, which is the reachable area by the center of the robot's wrist (intersection of axis J4 and J5). Table 4 summarizes the main characteristics and performance parameters of the robot.

3.2 MOTION CONTROLLER

The motion controller adopted by this model is ROBOX RP-1 (*Robox S.p.A.*) shown in Figure 11, which drives the axes by means of proprietary software. Robox is an Italian company started in 1975, that designs and produces electronic controllers, programming languages, and development environments for robotics and motion control systems. Finally, the robot may also be remotely operated by the *teach pendant* shown in Figure 12.

The main features of the motion controller are:

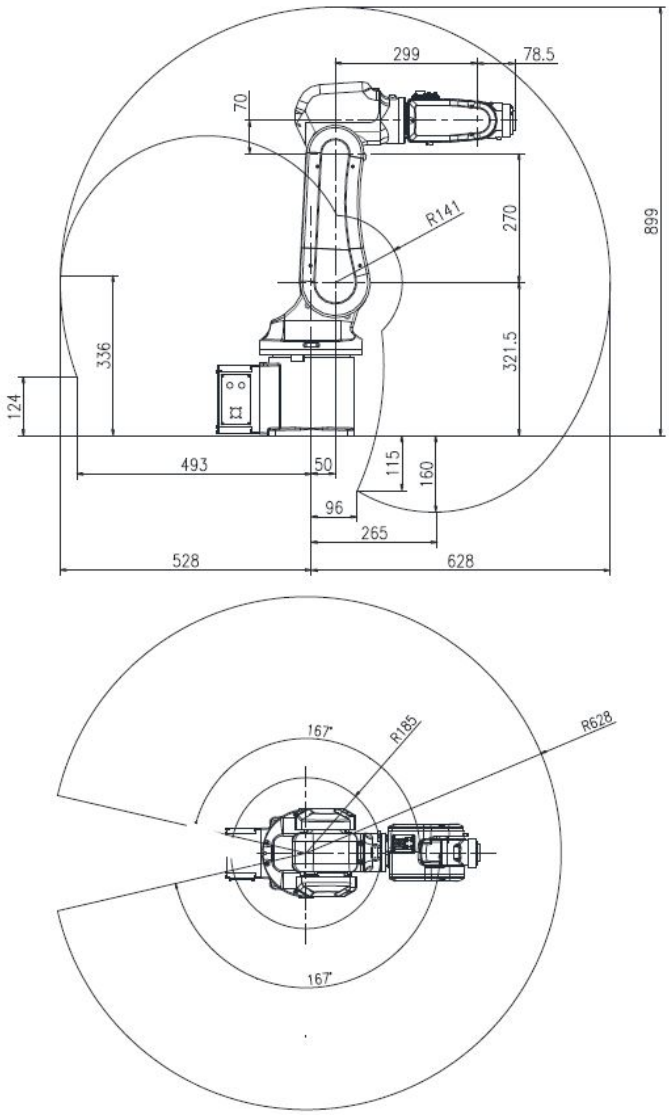


Figure 10: Dimensions and workspace of ER3A-C60

Hardware:

- Freescale Power PC MPC5200 microprocessor (400MHz);
- Up to 32 interpolated axes, driven through Ethercat (Ethernet-based fieldbus system, it is suitable for both hard and soft real-time computing requirements in automation technology) or CANopen fieldbus (communication protocol and device profile specification for embedded systems used in automation).

Memory:

- 60 MB compact flash memory (expandable);

Figure 11: ROBOX RP-1 controller

Figure 12: Teach pendant

- 64 MB DRAM (Dynamic random-access memory is a type of random access semiconductor memory that stores each bit of data in a separate tiny capacitor within an integrated circuit. The capacitor can either be charged or discharged; these two states are taken to represent the two values of a bit, conventionally called 0 and 1);
- 128 KB retentive ram for parameters and alarm history storage.

Communication:

- 1 general purpose RS232 serial channel (Robox BCC/31, DF1);
- 1 general purpose RS422/485 serial channel (Robox BCC/31, DF1);

- 2 Ethernet channels (10/100 Mbit/s) (CoE - Can Over EtherCAT, SoE - Servodrive Over EtherCAT, EoE - Ethernet over EtherCAT, TCP/IP, UDP, TFTP, Modbus/TCP, Ethernet/IP, Robox BCC /31/TCP);
- 2 Canbus channels (DS301, DS401, DS402, Device Net, Robox Cnet protocols) to interface servo drives and remote I/O's;
- 1 Profibus DP slave channel;
- Master bus Axioline for communication with Axioline peripherals (Phoenix Contact);
- O.P.C. Server and ActiveX available for communication in Windows environment.

Interfaces:

- 8 digital, PNP, 24VDC opto-coupled inputs (3 with capture functions);
- 8 digital, PNP, 24VDC, opto-coupled outputs (max 0.5A per channel);
- 1 incremental encoder input, RS422 line driver 5VDC with dedicate homing input.

3.3 SOFTWARE ENVIRONMENT

A hard real-time operating system (Firmware RTE, Real-Time Extended) is installed in the RP-1 controller which ensures that the different tasks are executed with the correct timing and priority. The tasks available to the user can be (i) on event (capture), (ii) at pre-programmed frequency (motion and auxiliary functions), and (iii) backgrounds (logics). A more detailed explanation is provided in Table 5.
The tasks can be written in structured text language (named R3) or ladder through the RDE environment.

RDE is a development environment designed to exploit the Robox R3, RPE, Ladder, and Object Block languages. It allows to (i) write, compile and debug the application software, (ii) evaluate and/or simulate the behavior of the controlled machine, and, as a result, (iii) choose the best solutions to optimize it. Furthermore, with RDE the machine may be described in a graphical form, configuring the axes, the power-sets, Object blocks, and the geometrical structures of the robot. This software operates on personal computers running Windows 2000, XP, Vista, Seven, Linux (X11), and MAC OS/X platforms. The controller is connected to the PC via serial/Ethernet channels (TCP/IP).

There are many languages available with RDE such as *structured text with motion libraries* suitable for motion control application, and *structured text with robotics libraries* suitable to describe robot paths. There is also the *OB (Object Block)* language which extends the concept of function blocks. The programmer

Priority	Task type
1	HIGH PRIORITY LADDER: Tasks that are written in ladder whose execution period is set by the user (1Hz÷2000Hz). Any RTE running operation is interrupted to handle these events. TASK ON EVENT: Particular tasks written in R3 language with maximum system priority, used to solve some particular requirements. Any RTE running operation is interrupted to handle these events
2	SYNCHRONOUS LADDER TASK: Tasks written in ladder language, executed with the RULES. RULES (fixed frequency functions -interrupt-): Tasks written in R3 language, reserved to (i) the path building, (ii) the descriptions of the links among the different axes, and (iii) the execution of the feedback algorithm (loop closure). RTE can execute up to 32 RULES (RC) at a time in the system interrupt. The selection of the RULES to be executed is done with the instruction GROUP, while the execution sequence can be programmed with the instruction ORDER. The rules execution frequency can be programmed with the instruction RULE_FREQ (in the range 25Hz÷2000Hz).
3	NORMAL PRIORITY LADDER TASKS: Task written in Ladder language whose execution frequency is programmed by the user (1Hz÷2000Hz).
4	RULE PERIODIC: RULE executed at a user-defined frequency. Since the priority of this rule is lower than the other RULES, its execution is subject to jitter, whose maximum length will be equal to the time required by RTE to execute the tasks with higher priority.
5	OB Service.
6	LOW PRIORITY LADDER TASKS: Ladder tasks whose execution frequency is defined by the user (1Hz÷2000Hz).
7	TASK in BACKGROUND (time-sharing): They are 8 low-priority tasks written in the R3 language, used for not time-consuming functions (i. e. managing the machine logic). Task 1 ($TASK1) represents the program entry point. This task enables/disables the other tasks. The typical architecture of these tasks consists of an initialization session followed by an endless loop where the different operations/tasks of the controlled machines are performed.

Table 5: Priority Task type

may import it from other languages and compile his own libraries written in C++ programming language. *Ladder IEC1131* is a language suitable for PLC programming which allows real-time monitoring of the program and real-time edits to the running program. At last, *ISO* allows the interpretation of ISO path descriptions generated by external CAD/CAM.

RDE offers the following main features:

Writing the R3, RPE, Ladder, Object block source files using its highlighting integrated editor. These files can also be written with any other pure ASCII editor.

Compiling the source files and obtaining the errors report addressing to the code line containing the error.

Organizing all the files belonging to a particular job into homogeneous groups. The project window is subdivided in:

- Configuration: in this session, the machine is configured in all details. TCP/IP address, Fieldbus networks, axes/connected device, etc.
- Programs: this window displays all the source programs used by the application software.
- Flash files: the file that will be uploaded to the RP-1 compact flash is shown here, also displaying the list of all the files included in the application software.
- RTE debugger: it allows online access to ladder/structured text programs.

Communicating on a Serial/Ethernet line with the Robox motion controller through the Shell window. This window allows to monitor the state of any hardware resource of the connected controller, read the directory of the files in the flashcard of the connected controller, move any file from the PC memory to the flashcard, and vice-versa, delete files, etc.

Displaying in a numerical form the registers/variables of the connected controller through the Monitor window.

Displaying in a graphical form the evolution of the involved variables through the Oscilloscope window. This window works exactly as an oscilloscope, for which it is possible to define the variables associated with each track, the scale and offset, the synchronism, etc.

Setting Debug utilities, such as break-points on the execution of an instruction (stopping the execution or just counting the event occurrences), break-points on a read or write variable operation, and trace on task.

Using the Variable Load/Save window to create customized parameter menus.

3.4 ROS DRIVER IMPLEMENTATION

Despite the numerous advantages offered by the RDE software, it was not suited to efficiently manage all the operations carried out during the experimental procedure of this thesis work. Moreover, the idea behind this project was to create a simple and easy-to-use system that could be adopted regardless of the robot and its motion controller, hence no proprietary software could be used. Moreover, since the procedure may last several hours, it was important to automate it as much as possible. As a result, the project has been developed in ROS (Robot Operation System), which is an open-source flexible platform to develop robot programs.

A typical experiment performed in this work has two steps. First, when the robot is in its cold state, a high-speed point-to-point movement is performed to warm it up. Then, the test trajectory is performed and the related data is collected. This procedure is repeated until the robot reaches its warmed-up state. The overall duration of a procedure is of about 2-4 hours, depending on the type of test performed. Moreover, different types of measurement devices have been mounted on the robot to collect relevant data (i. e. temperature sensors) that were not easy to integrate with the RDE system to automate the data collection procedure but were simple to use with ROS.

However, since ROS is a high-level framework, to communicate with the robot it is necessary to have a ROS-Industrial Robot Driver, which acts as the interface between ROS and the RDE software. Several drivers are available for the most famous IR manufacturers, however, this was not the case for the EFORT robots adopted in this work. Hence, as part of this thesis an EFORT ROS-Industrial driver has been developed (explained in Appendix A).

4

FRICTION MODELING

One of the main problems that must be addressed during the design of an automatic machine (robot, machine tool, etc.) concerns the study of friction phenomena. Friction is often the cause of several issues in mechanical structures, among which we can mention: (i) energy dissipation, (ii) wear, (iii) overheating, (iv) self-excited vibrations, and (v) dynamics problems. Friction occurs when there is relative motion between two touching surfaces and depends on several factors, such as: (i) the surfaces' geometry and roughness, (ii) their material, (iii) the movement speed, (iv) the lubricant's characteristics, (v) the operating temperature (which changes during the working cycle), (vi) the environmental conditions (temperature and humidity).

Friction is a big issue for industrial robots, causing problems related to motion control such as trajectory tracking errors, dynamic instabilities (stick-slip), limit cycles, etc. Furthermore, for some IRs the friction component of the overall actuating torque can reach very high percentages, meaning that most of the motor actuating power is needed to overcome friction alone. Therefore, the integration of mechanical and control design is necessary to develop efficient control strategies.

To implement mathematical models in motion controllers to efficiently predict friction variations as the robot operating conditions change, an intensive experimental activity is required. To provide a correct description of the physical phenomenon and to formulate efficient mathematical models, several theoretical studies have been published in the field of robotic research.

Mathematical models of friction are usually divided into two main categories: static models and dynamic models. Static models are simpler to analyze because the relation between the frictional force/torque and the independent variable (usually speed) is defined by a pre-defined mathematical relation. For example, the simple Coulomb model belongs to this category, in which the friction action is considered constant and depends only on the velocity direction. In the viscous model, the friction resistance varies linearly with the relative velocity of the contact surfaces. In the quadratic model, it is assumed that the resisting action is proportional to the relative velocity (this is the typical case of the aerodynamic drag). Other static models are the Stribeck model [65, 66, 62], in which the transition between the static and dynamic friction conditions is represented by a decreasing exponential function, and the polynomial model, in which a polynomial (usually second or third degree) is used to describe the relationship between velocity and frictional force.

On the contrary, in dynamic models, the friction force/torque depends on state variables that take into account the time history of the system and not only the current situation in which the machine operates as in static models. Among the best known dynamic models, we can mention the Dahl model [67], the LuGre model [68, 69, 70, 71] and the Maxwell-slip model [72], which is one of the most complex available.

One of the most difficult problems of robot control design is the evaluation of friction variations during machine operation. Throughout the working cycle, the robot's axes move at a different speed, hence the temperature of the mechanical transmissions (gears, harmonic drives) changes due to the heating of the mechanical parts (as is known, the energy dissipation due to friction raise temperature [73, 74]). Furthermore, the heating of parts may also occur because of environmental changes in temperature in both summer and winter seasons.

Friction variations during the robot movement affect its positioning precision because friction compensation algorithms implemented in the axis control system typically adopt static models. Therefore, the attention of researchers is increasingly oriented towards the study of dynamic models of friction, particularly focusing on the effect of temperature [74, 75, 76].

The research activity of this thesis mainly investigates the relationship between temperature and friction, showing that friction variations may be modeled and predicted by knowing the mechanical characteristics of the robot and its work cycle. Hence, the goal is to develop a novel friction model that explains the behavior of robot joints for which standard models fail, as presented in several publications resulting from this thesis work [77, 78, 63, 79]. Possible applications of this study concern the fields of predictive maintenance, advanced control strategies, the realization of virtual force sensors, impact detection, and control of human-machine interaction without force sensors. If the friction forces are precisely known, the interaction force can be predicted by measuring the motor torques and using the dynamic model of the robot.

4.1 DYNAMICS ANALYSIS

As already explained in 1.1, the complete dynamic model of a robot is based on the well-known equation:

$$\tau = \mathbf{M}(\mathbf{q})\ddot{\mathbf{q}} + \mathbf{V}(\mathbf{q}, \dot{\mathbf{q}})\dot{\mathbf{q}} + \mathbf{G}(\mathbf{q}) + \boldsymbol{\tau}_f - \mathbf{J}^T(\mathbf{q})\mathbf{h}_e \tag{85}$$

where $\mathbf{q}$, $\dot{\mathbf{q}}$, and $\ddot{\mathbf{q}}$ are the joint position, velocity, and acceleration vectors respectively, τ is the vector of the driving actions applied to the robot joints, $\mathbf{M}(\mathbf{q})$ the inertial matrix, $\mathbf{V}(\mathbf{q}, \dot{\mathbf{q}})$ the matrix containing the centrifugal and Coriolis terms, $\mathbf{G}(\mathbf{q})$ the gravitational term, $\mathbf{J}$ the extended Jacobian matrix, $\mathbf{F}_e$ the vector of the external forces and $\boldsymbol{\tau}_f$ the vector containing the frictional effects on the joints.

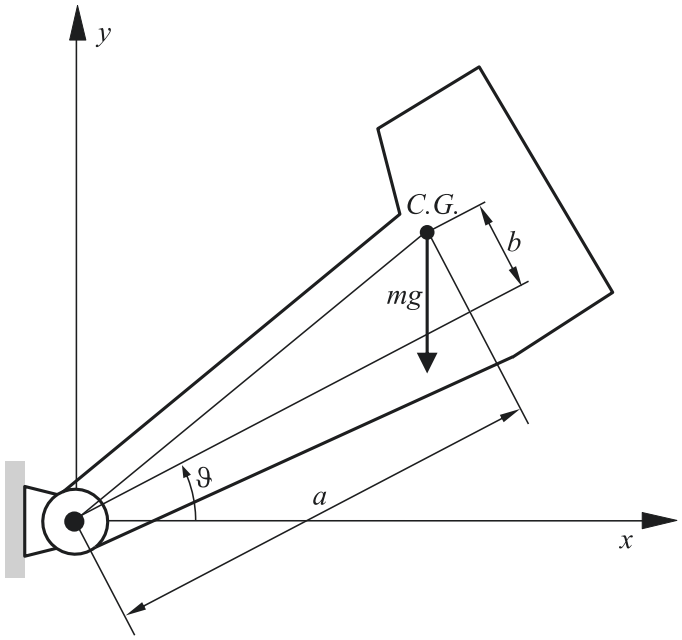


Figure 13: Scheme of a revolute joint with gravitational effect

The estimation of friction carried out by the simultaneous actuation of several robot joints requires the use of the complete dynamic model of the manipulator, described by Equation (85). However, even if this approach is feasible, it is rather complicated to perform. Hence, it is preferable to actuate one joint at a time to simplify the dynamic analysis.

4.1.1 *Dynamic Model of a Single Robot Axis*

To properly analyze friction effects, individual joint motions have been considered to obtain simpler models. Referring to Figure 13, the dynamic equilibrium equation can be expressed by the following equation:

$$\tau = J\ddot{\theta} + \tau_g + \tau_f \tag{86}$$

where τ, τ_g, and τ_f indicate the driving torque, the torque due to gravity, and the friction torque respectively. $\ddot{\theta}$ and J are the angular acceleration of the arm and the moment of inertia reduced to the motor axis (the effect of the gearbox transmission ratio is therefore included). Bearing in mind that the center of gravity, in general, does not belong to the arm axis, it is convenient to introduce parameters a and b (see Figure 13) indicating the position of the center of gravity when $\theta = 0$. Therefore, the gravitational torque is:

$$\tau_g = mg(a\cos\theta - b\sin\theta) = Q_a\cos\theta - Q_b\sin\theta \tag{87}$$

where $Q_a = mga$ and $Q_b = mgb$.

Concerning dissipative effects, it should be noted that temperature and velocity play a fundamental role in determining the friction torque affecting the robot joints. Some studies conducted by the author in a previous research [76, 80] show that this friction torque can be modeled with good approximation by a polynomial of the type:

$$\tau_f = a_0 sgn(\dot{\theta}) + a_1\dot{\theta} + a_2\dot{\theta}^2 sgn(\dot{\theta}) + a_3\dot{\theta}^3 \tag{88}$$

where $sgn(x) = x/|x|$ if $x \neq 0$ and $sgn(x) = 0$ if $x = 0$.

Coefficients a_i $(i = 0, \ldots, 3)$, which must be experimentally identified, can be often considered independent from (i) the direction of rotation and (ii) positive and negative velocities. However, there are cases where two sets of coefficients have to be considered for the two motion directions. From Equations (86), (87), and (88) we obtain:

$$\begin{aligned} \tau &= J\ddot{\theta} + Q_a \cos\theta - Q_b \sin\theta + \\ &+ a_0 sgn(\dot{\theta}) + a_1\dot{\theta} + a_2\dot{\theta}^2 sgn(\dot{\theta}) + a_3\dot{\theta}^3 \end{aligned} \tag{89}$$

Now, if we know the values of the kinematic variables θ_i, $\dot{\theta}_i$, $\ddot{\theta}_i$, and the value of the driving torque τ_i at the generic time instant t_i, Equation (89) can be rewritten n times, where $n = t_a f_c$, being t_a the total duration of the acquisition and f_c the frequency of sampling.

It should be noted that the values of the rotation angle and the angular velocity can be obtained from the sensors mounted on the robot (used for motion control), while the drive torque can be obtained by multiplying the current signal available on the motor drive by the motor torque constant.

Acceleration can be calculated by numerical differentiation of the velocity signal since it is often infeasible to mount accelerometers on the robot.

Using matrix notation, the n equations (89) result in a linear system in the following form:

$$\tau = \mathbf{A}\boldsymbol{x} \tag{90}$$

where

$$\tau = [\tau_1 \quad \tau_2 \quad \ldots \quad \tau_n]^T \tag{91}$$

is the vector of the experimentally detected driving torques,

$$\mathbf{A} = \begin{bmatrix} \ddot{\theta}_1 & \cos\theta_1 & -\sin\theta_1 & sgn(\dot{\theta}_1) & \dot{\theta}_1 & \dot{\theta}_1^2 sgn(\dot{\theta}_1) & \dot{\theta}_1^3 \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ \ddot{\theta}_i & \cos\theta_i & -\sin\theta_i & sgn(\dot{\theta}_i) & \dot{\theta}_i & \dot{\theta}_i^2 sgn(\dot{\theta}_i) & \dot{\theta}_i^3 \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ \ddot{\theta}_n & \cos\theta_n & -\sin\theta_n & sign(\dot{\theta}_n) & \dot{\theta}_n & \dot{\theta}_n^2 sgn(\dot{\theta}_n) & \dot{\theta}_n^3 \end{bmatrix} \tag{92}$$

the matrix containing rotations, velocities and angular accelerations and

$$x = [J \quad Q_a \quad Q_b \quad a_0 \quad a_1 \quad a_2 \quad a_3]^T \tag{93}$$

is the vector of the unknown coefficients, which can be estimated by the Least Squares criterion using the following relation:

$$x = \mathbf{A}^+ \tau \tag{94}$$

where the symbol $\mathbf{A}^+$ indicates the Moore-Penrose pseudo-inverse matrix.

To obtain reliable results, the motor must reach all the allowed velocity values (in both directions) several times to eliminate the influence of measurement noise. It should be noted that, although Equation (88) allows to correctly model the relationship between drive velocity and friction torque, it does not take into account the friction variation due to temperature. As is known, friction produces heat that gradually increases the temperature of the mechanical transmission, making the oil less viscous and progressively reducing the friction torque. The inaccuracy of robot positioning is due to thermal errors in addition to geometric errors, as shown in [81, 82], and this justifies a more in-depth study of thermal phenomena.

Figure 14 shows the variation of friction torque as a function of the speed in the gearbox of an industrial robot. Each curve has been obtained at a different temperature of the gearbox, performing the first test with a cold start (therefore with a temperature equal to the environmental one). The trend of the curves increases with speed and, for a given speed value, the friction torque is lower when the transmission has been heated during the execution of previous working cycles. Therefore, to correctly evaluate the effect of temperature on friction, a thermal balance equation must also be included in the calculations, as will be described in the following section.

4.2 TEMPERATURE EFFECT - FIRST ORDER MODEL

To analyze the influence of temperature on the friction torque, we must first define the thermal balance equation of a joint (see Figure 15), which can be expressed as:

$$W_{acc} = W_{in} - W_{out} \tag{95}$$

where the symbols W_{in}, W_{out}, and W_{acc} are the mechanical power entering the transmission due to friction, the thermal power dissipated in the environment, and the thermal power accumulated inside the gearbox, respectively. Using for convenience the symbol ω to indicate the angular velocity of the joint (i.e. $\omega = \dot{\theta} = \frac{d\theta}{dt}$), the mechanical power due to friction can be written as:

$$W_{in} = \tau_f \omega \tag{96}$$

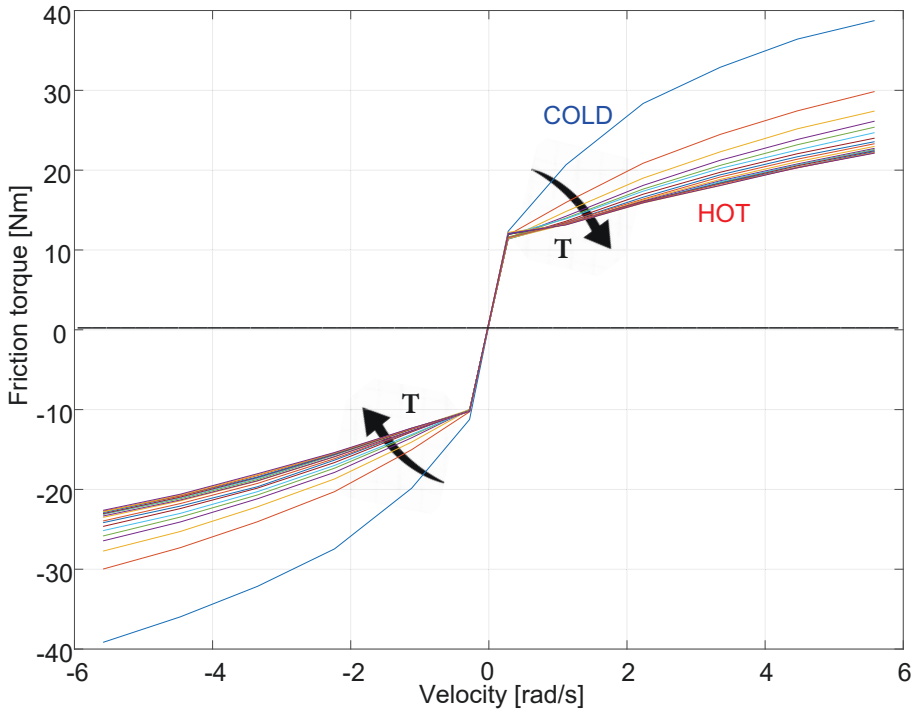


Figure 14: Friction versus speed changes as the robot warm up (joint 1 of the robot in Figure 17)

where the friction torque τ_f for an assigned joint velocity ω changes with the joint temperature T.

The output power W_{out} can be calculated by multiplying the heat exchange coefficient K and the difference between the internal temperature $T(t)$ of the gearbox and the ambient temperature T_e (assumed constant), that is:

$$W_{out} = K(T - T_e) \tag{97}$$

Lastly, the thermal power W_{acc} that accumulates inside the gearbox is obtained as:

$$W_{acc} = C\frac{dT}{dt} \tag{98}$$

where C is the thermal capacity of the gearbox, and dT/dt is the temperature gradient.

Based on Equation (95) we can write:

$$\frac{dT}{dt} = \frac{W_{in} - W_{out}}{C} = \frac{\tau_f \omega - K(T - T_e)}{C} \tag{99}$$

Equation (99) is a linear differential equation that must be solved taking into account the dependence of the torque on the temperature. Therefore, we assume

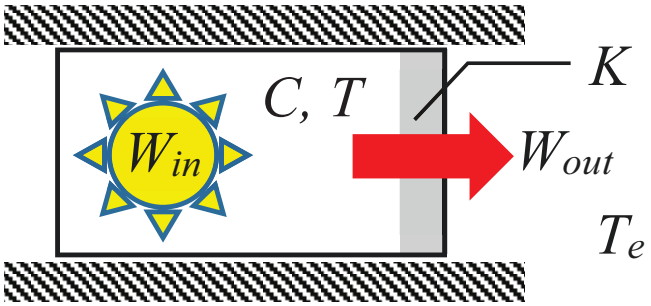


Figure 15: Model of the joint with one thermal capacity and one temperature

that the four coefficients $a_0, \ldots, a_3$ of Equation (88) are linearly variable with the temperature, i.e.:

$$a_i = a_{i0}[\alpha(T - T_0) + \beta] \qquad i = 0, \ldots, 3 \tag{100}$$

where a_{i0} is the value of a_i for $T = T_0$, and α and β are two constants ($\alpha < 0$) of the linear relationship between the friction torque and temperature[1]. The hypothesis of linear dependence of coefficients a_i with temperature is empirical since it has not been obtained by a rigorous analysis of the phenomenon. Nevertheless, this assumption proved to be adequate in the case of some robots [76]. Moreover, temperature T is taken as the average temperature of the joint, thus taking into account the frictional effects of various elements of the gearbox (gears, bearings, etc.).

By substituting Equation (100) in Equation (88) we get:

$$\tau_f = [a_{00}sgn(\dot{\theta}) + a_{01}\dot{\theta} + a_{02}\dot{\theta}^2 sgn(\dot{\theta}) + a_{03}\dot{\theta}^3][\alpha(T - T_0) + \beta] \tag{101}$$

This equation can be rewritten in a more compact form as:

$$\tau_f = \tau_{f0}[\alpha(T - T_0) + \beta] \tag{102}$$

where τ_{f0} clearly indicates the value of the friction torque at the initial temperature T_0, i.e.:

$$\tau_{f0} = [a_{00}sgn(\dot{\theta}) + a_{01}\dot{\theta} + a_{02}\dot{\theta}^2 sgn(\dot{\theta}) + a_{03}\dot{\theta}^3] \tag{103}$$

By considering the environment temperature as the initial temperature ($T_0 = T_e$), Equation (102) becomes:

$$\tau_f = \tau_{fe}[\alpha(T - T_e) + \beta] \tag{104}$$

1 Being a_{i0} the value of a_i for $T = T_0$, one should get $\beta = 1$ based on Equation (100). However, the value may be different due to (i) non-considered friction phenomena, and (ii) the unavoidable presence of noise in the signal captured during the experimental tests.

where τ_{fe} is the friction torque obtained considering the environment temperature T_e. By considering $T_0 = T_e$ it is obvious that the robot has been inactive for a long time and, therefore, all its internal parts are in thermal equilibrium and can be assumed at the same temperature T_e.

By substituting Equation (104) in Equation (134), we obtain:

$$W_{in} = \tau_{fe}\omega[\alpha(T - T_e) + \beta] = \overline{W}[\alpha(T - T_e) + \beta] \tag{105}$$

where $\overline{W} = \tau_{fe}\omega$ is the friction power produced at the environment temperature. Equation (105) can be rewritten compactly as:

$$W_{in} = aT + b \tag{106}$$

$$a = \overline{W}\alpha \qquad b = \overline{W}(\beta - \alpha T_e) \tag{107}$$

From Equation (99) and (105) we obtain:

$$\frac{dT}{dt} = \frac{(aT + b) - K(T - T_e)}{C} = \lambda T + \mu \tag{108}$$

where the coefficients λ and μ are:

$$\lambda = \frac{a - K}{C} \qquad \mu = \frac{KT_e + b}{C} \tag{109}$$

Considering that $\overline{W}$ is constant, the solution of the differential equation (108) is:

$$T(t) = T_\infty + (T_e - T_\infty)e^{-t/t_w} \tag{110}$$

where the asymptotic temperature T_∞ and the time constant t_w can be calculated as follows:

$$T_\infty = -\frac{\mu}{\lambda} = \frac{KT_e + \overline{W}(\beta - \alpha T_e)}{K - \overline{W}\alpha} \tag{111}$$

$$t_w = -\frac{1}{\lambda} = \frac{C}{K - \overline{W}\alpha} \tag{112}$$

Since $\alpha < 0$ and $\overline{W} > 0$, the denominator $K - \overline{W}\alpha$ increases with the increase of $\overline{W}$; hence, the thermal time constant t_w decreases with increasing power $\overline{W}$. This implies that the warming up phase becomes ever more rapid with increasing power $\overline{W}$ produced by friction at the environment temperature.

To analyze the thermal behavior of the transmission during the cooling phase, it is sufficient to set $W_{in} = 0$ in Equation (99) since the robot motors are inactive. The starting temperature will be equal to T_0, corresponding to the temperature

reached by the transmission when the robot has been stopped at the start of the cooling phase. Therefore, the resulting differential equation is:

$$\frac{dT}{dt} = \frac{-K(T - T_e)}{C} \tag{113}$$

Solving Equation (113) we can calculate the time history of the joint temperature during the cooling phase:

$$T(t) = T_e + (T_0 - T_e)e^{-t/t_c} \tag{114}$$

where the time constant t_c is given by:

$$t_c = \frac{C}{K} \tag{115}$$

Since $\alpha < 0$, Equations (112) and (115) clearly show that the cooling time constant t_c is greater than the time constant t_w, meaning that the cooling phase is slower that the warming up phase.

The thermal model described above has been used successfully on an anthropomorphic robot Comau [21, 76].

4.3 TEMPERATURE EFFECT - SECOND ORDER MODEL

The simple model proposed in the previous section worked well for some robots and was useful to clarify the general phenomena, but it was insufficient for the manipulator adopted in this work. Therefore, it was necessary to develop a more complex thermal model (see Figure 16), which requires the introduction of two thermal capacities C_1 and C_2 and two heat exchange coefficients K_1 and K_2.

In this model the state variables are represented by two different temperatures T_1 and T_2 and the input power W_{in} produced by friction assumes the expression given by Equation (106), (i.e. $W_{in} = aT_1 + b$). In this case, however, it only depends on temperature T_1 since the power source is positioned on the left part of the system (see Figure 16).

In our experiments, T_1 represents the temperature of the area where the heat is generated, while T_2 represents the temperature of the surrounding area. Hence, it is necessary to write two different thermal balance equations:

$$\begin{cases} \dfrac{dT_1}{dt} = \dfrac{(aT_1 + b) - K_1(T_1 - T_2)}{C_1} \\ \\ \dfrac{dT_2}{dt} = \dfrac{K_1(T_1 - T_2) - K_2(T_2 - T_e)}{C_2} \end{cases} \tag{116}$$

Using matrix notation and considering the definitions given in Equation (107),

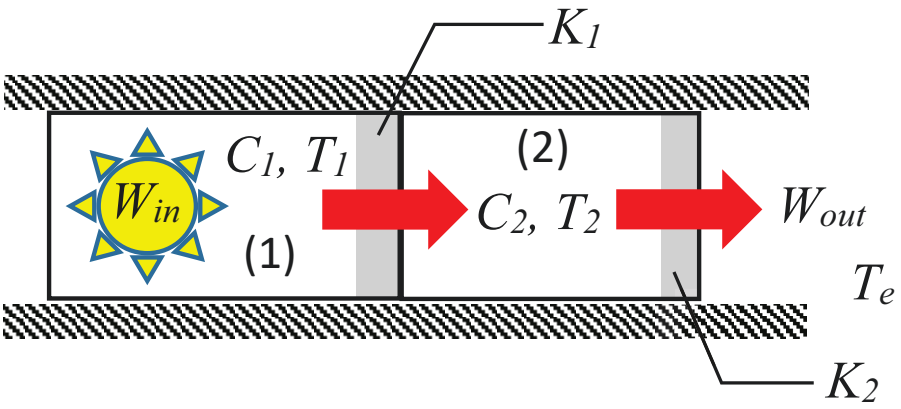


Figure 16: Model of the joint with two thermal capacities and two temperatures: (1) area where heat is produced by friction; (2) surrounding area.

they can be rewritten as:

$$\begin{bmatrix} \frac{dT_1}{dt} \\ \\ \frac{dT_2}{dt} \end{bmatrix} = \begin{bmatrix} \frac{\overline{W}\alpha - K_1}{C_1} & \frac{K_1}{C_1} \\ \\ \frac{K_1}{C_2} & -\frac{K_1 + K_2}{C_2} \end{bmatrix} \begin{bmatrix} T_1 \\ \\ T_2 \end{bmatrix} + \begin{bmatrix} \frac{\overline{W}(\beta - \alpha T_e)}{C_1} \\ \\ \frac{K_2 T_e}{C_2} \end{bmatrix} \tag{117}$$

This is a system of two first-order differential equations, which can be rewritten in compact form as:

$$\dot{\mathbf{y}} = \mathbf{A}\mathbf{y} + \mathbf{b} \tag{118}$$

where

$$\mathbf{y} = [T_1 \quad T_2]^T \tag{119}$$

is the state vector containing the temperatures,

$$\mathbf{A} = \begin{bmatrix} \frac{\overline{W}\alpha - K_1}{C_1} & \frac{K_1}{C_1} \\ \\ \frac{K_1}{C_2} & -\frac{K_1 + K_2}{C_2} \end{bmatrix} \tag{120}$$

is the system matrix, and the vector $\mathbf{b}$ is defined as follows:

$$\mathbf{m} = \begin{bmatrix} \frac{\overline{W}(\beta - \alpha T_e)}{C_1} \\ \\ \frac{K_2 T_e}{C_2} \end{bmatrix} \tag{121}$$

Please note that $\overline{W}$ appears in both matrix $\mathbf{A}$ and vector $\mathbf{b}$. By using the matrix notation in [83], and considering $\overline{W}$ constant, the solution of the system (116) can be written as:

$$\mathbf{y}(t) = \mathbf{y}(\infty) + e^{\mathbf{A}t}\Big(\mathbf{y}(0) - \mathbf{y}(\infty)\Big) \tag{122}$$

where

$$\mathbf{y}(0) = [T_1(0) \quad T_2(0)]^T \tag{123}$$

is the vector of the initial temperatures, and

$$\mathbf{y}(\infty) = [T_1(\infty) \quad T_2(\infty)]^T = -A^{-1}\mathbf{b} \tag{124}$$

are their value at the steady state.

The thermal time constants can be obtained by calculating the eigenvalues of the matrix $\mathbf{A}$, i.e. by solving the equation:

$$|\mathbf{A} - s\mathbf{I}| = 0 \tag{125}$$

where s is the generic eigenvalue of matrix $\mathbf{A}$, and $\mathbf{I}$ is the 2×2 identity matrix. By extending the calculations, we obtain the following 2^{nd} degree polynomial:

$$s^2 + ps + q = 0 \tag{126}$$

where the coefficients p and q are given by:

$$\begin{aligned} p &= \frac{C_1(K_1 + K_2) + C_2(K_1 - \overline{W}\alpha)}{C_1 C_2} \\ q &= -\frac{(K_1 + K_2)\overline{W}\alpha - K_1 K_2}{C_1 C_2} \end{aligned} \tag{127}$$

The solutions s_1 and s_2 of Equation (126) can be used to calculate the two thermal time constants of the system, i.e. $t_{w1} = -1/s_1$ and $t_{w2} = -1/s_2$.

4.3.1 *Second Model Validation*

The robot used in the experimental tests is the Efort Model ER3A-C60 installed at the Industrial Robotics Laboratory of the University of Brescia, as described in Chapter 3. Figure 17 shows the mechanical structure of the robot in its default configuration, with each joint at its zero position.

Several tests have been performed by executing different trajectories on the robot and estimating the friction torque during the working cycles. In the experiments, each joint has been moved individually since there is not cross-influence

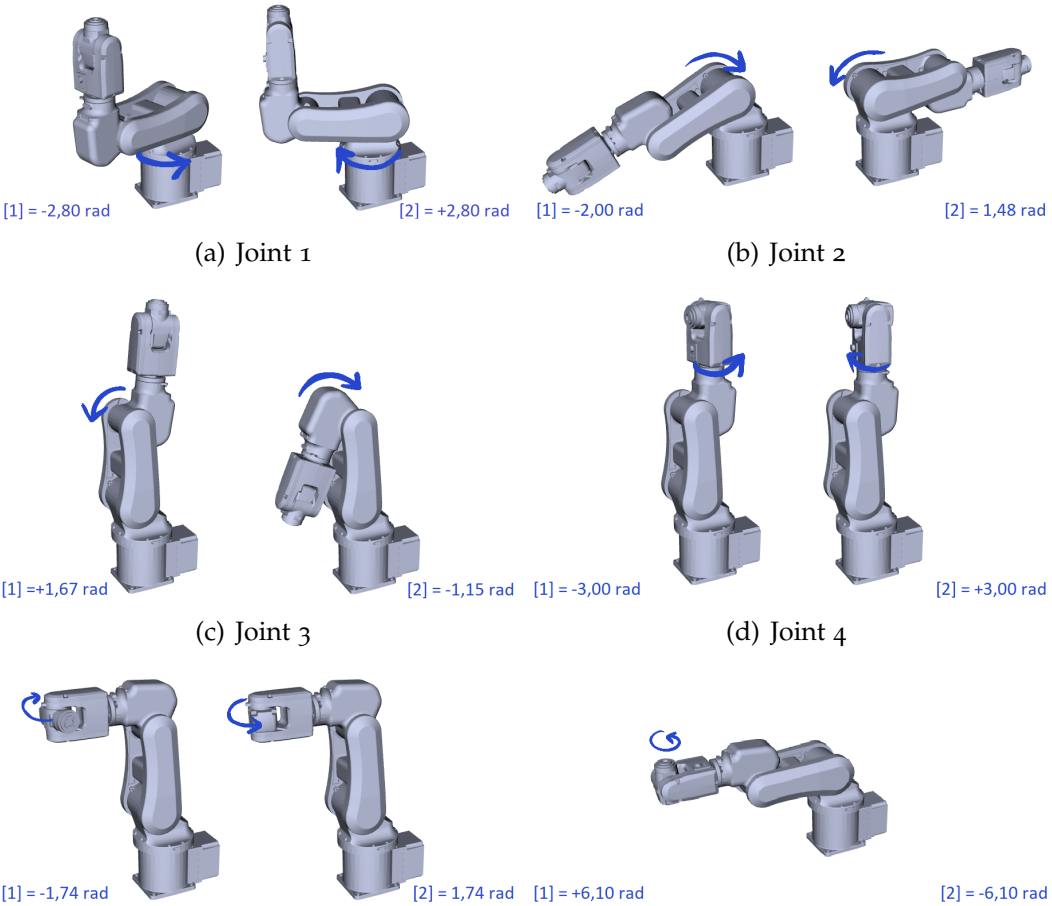
Figure 17: Robot Efort model ER3A-C60

between them affecting the friction values. The component of the torque due to dynamics effects of each joint is influenced by the others. However, the friction component depends on the sliding inside the transmission, therefore, the joints can be considered decoupled. These hypotheses have also been adopted by other papers [73, 74, 84].

The data have been collected using two different tests: the warming test and the cooling test. The warming test has been performed using a trajectory including different velocities equal to 5%, 20%, 40%, 60%, 80%, and 100% of the maximum joint speed, as shown in Figure 19. It has been designed to be short, assuring that during its execution the robot is not heated further. The test trajectory has been executed starting from a cooled-down condition of the robot. Then, each joint has been continuously actuated with a cycle motion at its maximum speed to warm it up. Figure 18 shows the movement of each joint during the experiment.

Subsequently, each 2 minutes the test trajectory has been executed to measure the joint torques at a predefined speed. This procedure has been repeated until the robot reached its warm state. Figure 19 also shows the torque of one joint (i) at the start, and (ii) at the end of the test. It is interesting to note that the torque decreases drastically in the second case.

To confirm the assumption of Equation (1) and (86) that the configuration of the other axis does not affect the friction torque of the joint under examination, a preliminary test has been performed. The first axis of the robot has been moved to two different configurations shown in Figure 20 and the related torque has been measured. The results show a substantial difference in the inertial phenomena for the two cases, but they do not reveal significant changes with respect to friction, confirming the previous thesis. Figure 21 presents a selection of the data



(a) Joint 1 (b) Joint 2

(c) Joint 3 (d) Joint 4

(e) Joint 5 (f) Joint 6

Figure 18: Joint movement during friction tests. Each joints move from position [1] to [2] and vice versa repeatedly. The blue arrows show the rotation directions.

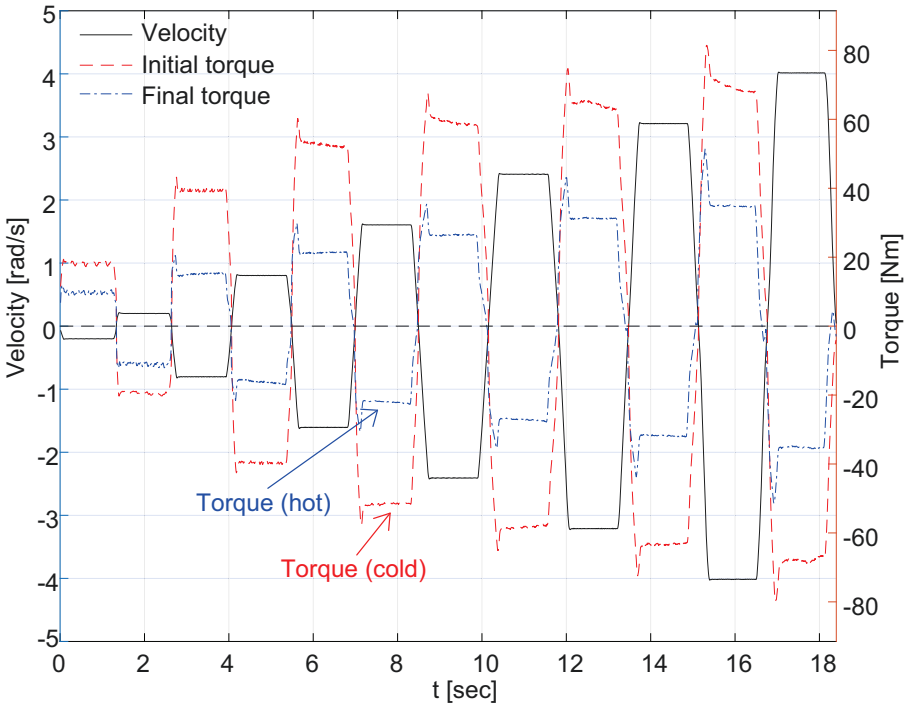


Figure 19: Speed and torque versus time during a working cycle in cold condition and in hot condition: experimental data, axis 1

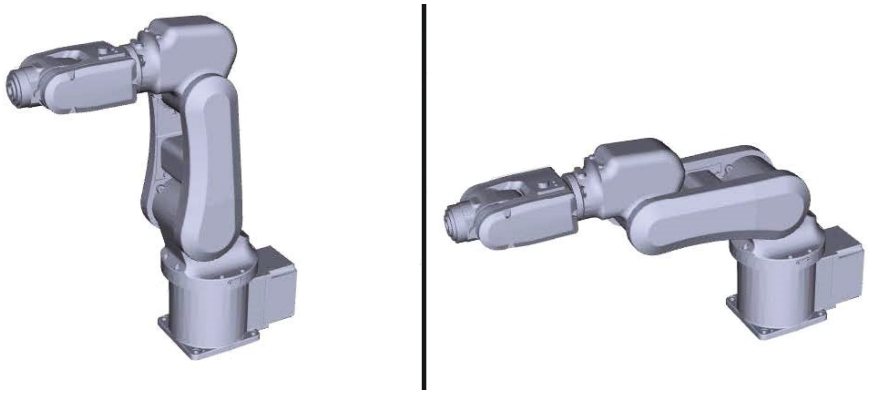


Figure 20: Robot configurations during the preliminary test: configuration 1 (left) and configuration 2 (right)

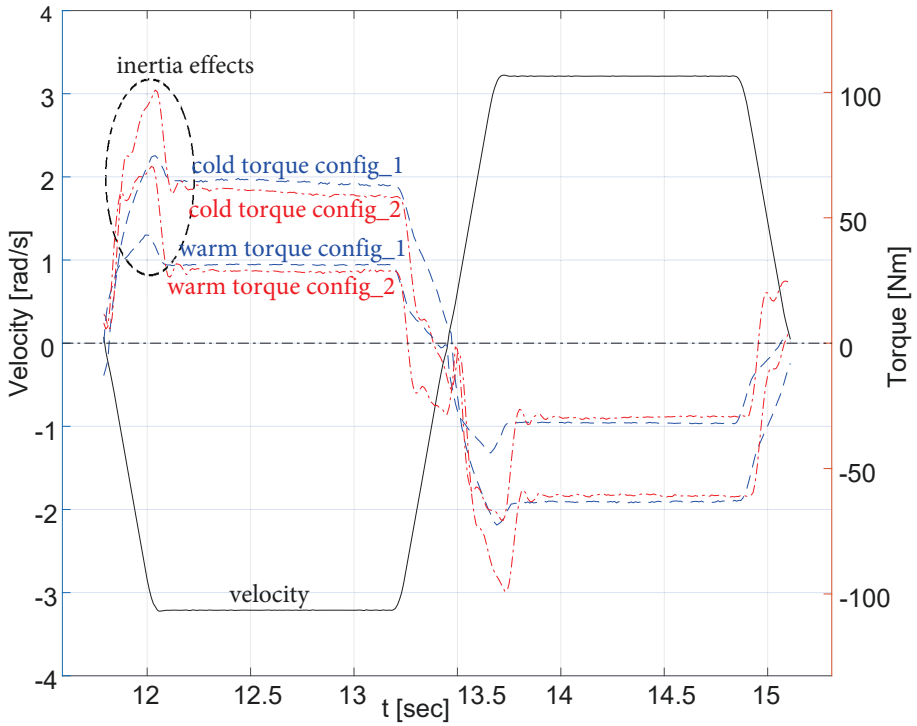


Figure 21: Velocity and torque versus time during a working cycle in cold and warm condition with the robot in configuration 1 and 2: experimental data, axis 1

related to the working cycles taken for both the cold and warm conditions. To estimate the friction torque, the robot has been put in a configuration that avoids the gravitational contribution from Equation (1) whenever possible. When this was not possible, the gravitational contribution has been subtracted from the measured motor torque. Moreover, to ignore the inertial effects, only the data of periods at constant speed have been considered. Since the friction torque of each joint shows limited variation during the cycle of duration t_d, its average value has been considered:

$$\tau_{f_M} = \frac{1}{t_d}\int_{t_d} \tau_f \, dt. \tag{128}$$

According to [21] the graph of τ_{f_M} versus time could be approximated with exponential functions. Figure 22 shows an example of the friction torque decreasing during the heating phase. Each curve represents the reduction of the joint friction at a specific velocity. The torque is represented as the ratio between the actual value and the maximum value (at the start point). Since all the curves decrease with similar behavior, it is possible to state that friction is linearly related to temperature, thus confirming the assumption of Equation (100). Initially, the friction torque versus time has been approximated using a single exponential function. The poor results obtained by such approximation suggested the adoption of a more sophisticated model using two exponentials. A system of this type has two

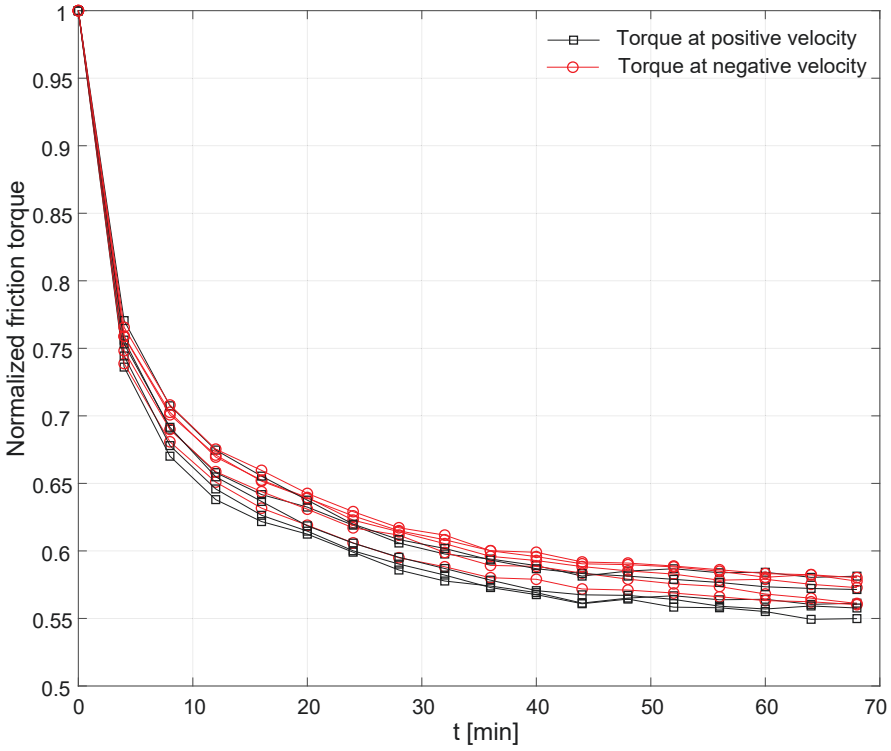


Figure 22: Friction torque at different velocity with respect to the increasing of temperature: experimental data, axis 4

different time constants and it has already been detailed starting from Equation (116).

To estimate the parameters of the model, it was necessary to analyze some experimental data. The tests have been performed by introducing different thermal power values in the system (i.e. the joint has been actuated with different duty cycles during the warming up).

Considering a cycle of duration t_c over a period of duration t_p (Figure 23), the average thermal power in the cycle and in the period is:

$$W_c = \frac{\int_{t_c} \tau_f \dot{q} dt}{t_c} \qquad W_p = D\, W_c = \frac{\int_{t_c} \tau_f \dot{q} dt}{t_p} \qquad D = \frac{t_c}{t_p} \tag{129}$$

where D is the duty cycle. It is trivial to note that the maximum heat is reached for $D = 1$, while $D = 0$ corresponds to the maximum cold condition. Since a period much shorter than the thermal time constant has been chosen for the transmissions ($t_p \ll \bar{t}$), it was possible to consider the average power $\overline{W}_p$ rather than its instantaneous value. The temperature of each joint can be estimated during runtime without any temperature measurement system by integrating the differential equation (117) and remembering Equation (134). The friction torque τ_f is obtained from the experimental measure by subtracting the terms predicted

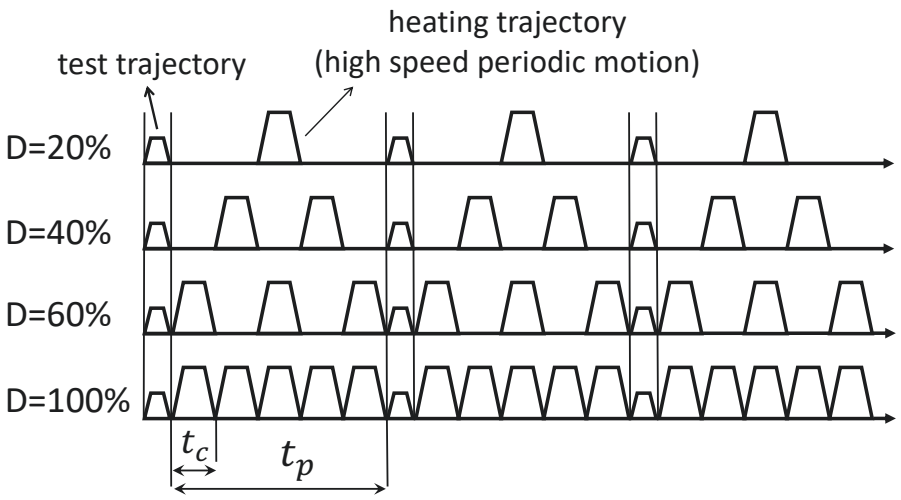


Figure 23: Cycle time t_c over the period t_p for different duty cycles

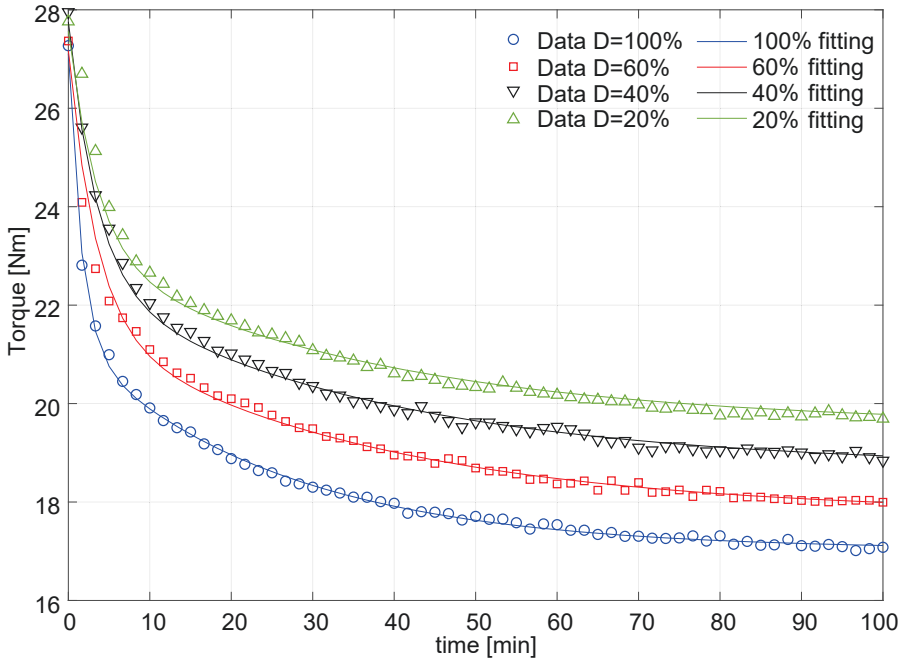


Figure 24: Friction torque versus time at 60% of maximum speed and fitting functions during the execution of the working cycle for different duty cycles (D=20%,40%,60%,100%), axis 4

Table 6: Errors on joint 4, heating test at maximum duty cycle

Errors	First model	Second model
abs mean	0.39 [Nm] → 2.1%	0.06 [Nm] → 0.3%
abs max	2.75 [Nm] → 10.1%	0.25 [Nm] → 1.1%
Steady state	0.61 [Nm] → 3.6%	0.03 [Nm] → 0.2%

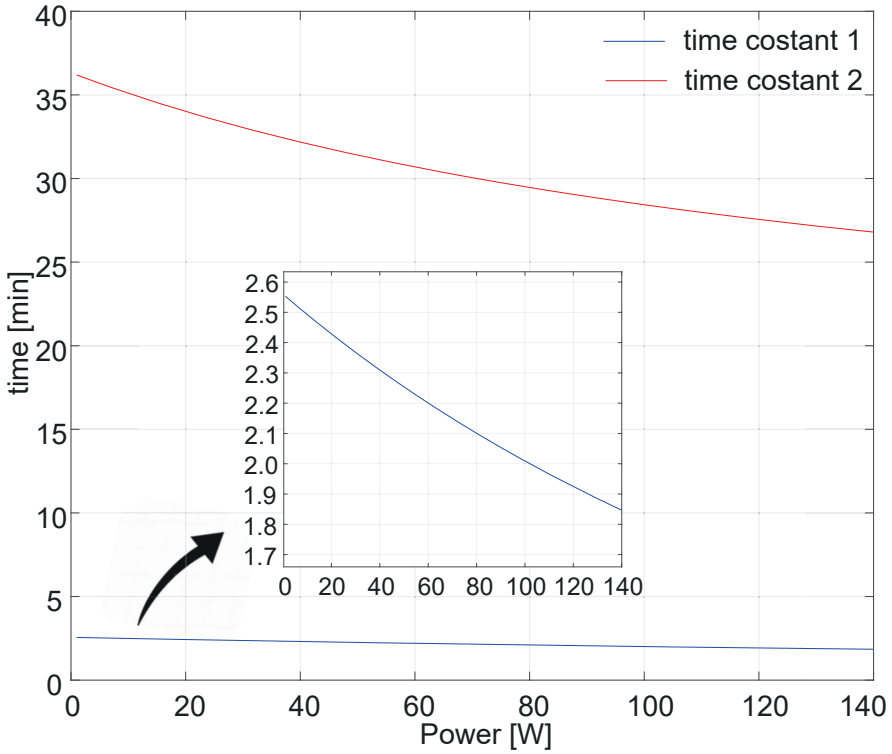


Figure 25: Time constants versus thermal power injected in the system. Time constant 1 has been enlarged for clarification

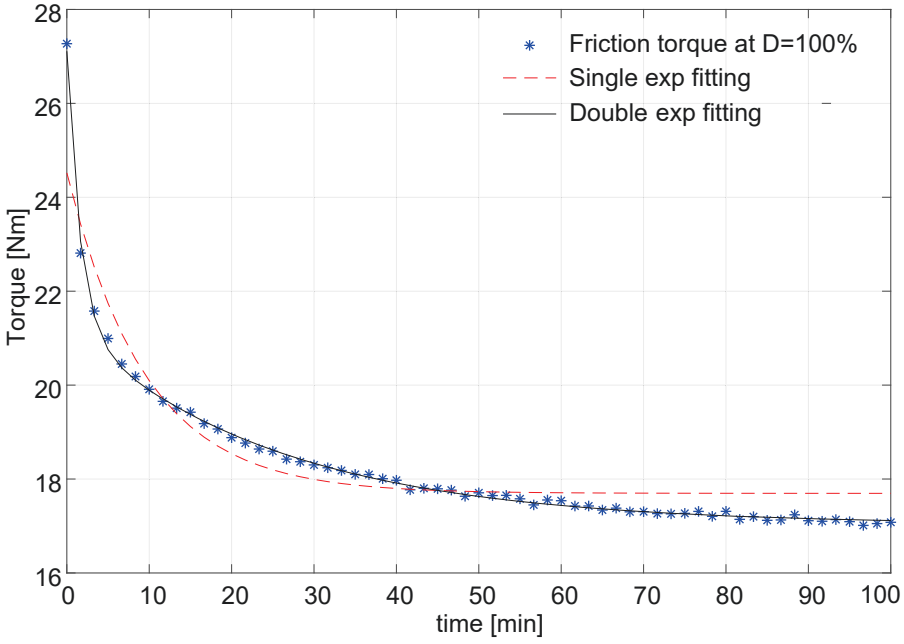


Figure 26: Friction torque versus time at 60% of maximum speed during the heating at D=100%. Experimental data, axis 4. Comparison between 1st and 2nd thermal model, see also Table 6

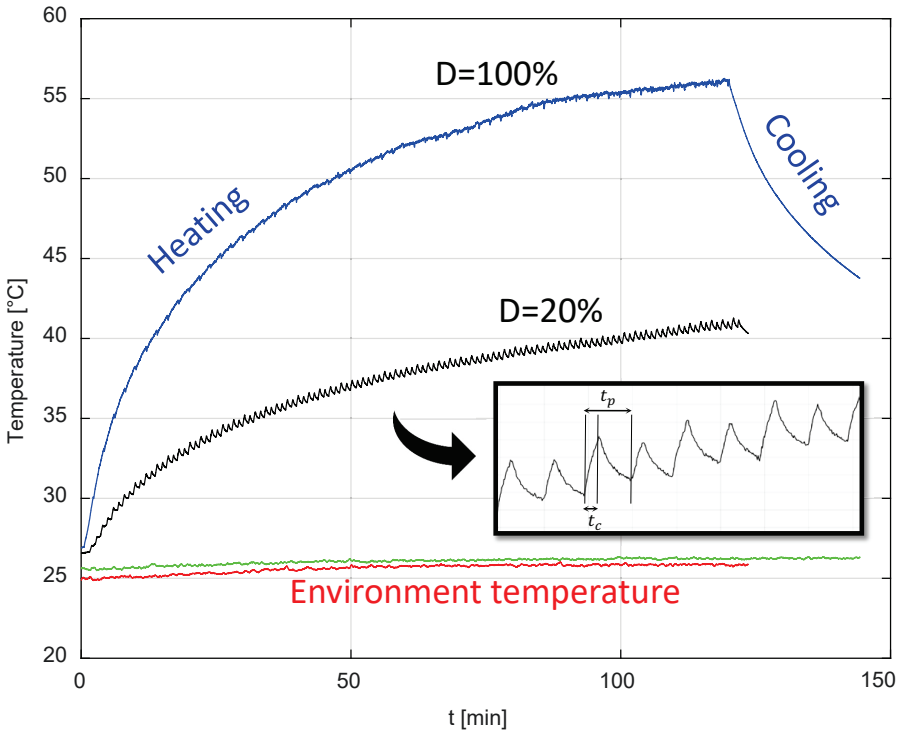


Figure 27: Example of temperature measurement near the speed reducer of joint 4 for different duty cycles

by the dynamic model (1). It is very difficult to insert sensors inside the transmissions, nevertheless, some thermocouples have been mounted near the speed reducer to monitor the temperature trend. The measured temperature has not been used in the model, but it shows that the joints heat up due to friction. As an example, Figure 27 shows the temperature rise during the motion and the cooling down when the robot is stopped, for two different duty cycles. The parameters of Equation (117) have been estimated by using the least square criteria. Some of

Table 7: Errors on joint 4, cooling test

Errors	First model	Second model
abs mean	0.05 [Nm] $\rightarrow$ 0.2%	0.05 [Nm] $\rightarrow$ 0.2%
abs max	0.14 [Nm] $\rightarrow$ 0.8%	0.14 [Nm] $\rightarrow$ 0.8%
Steady state	0.04 [Nm] $\rightarrow$ 0.2%	0.04 [Nm] $\rightarrow$ 0.2%

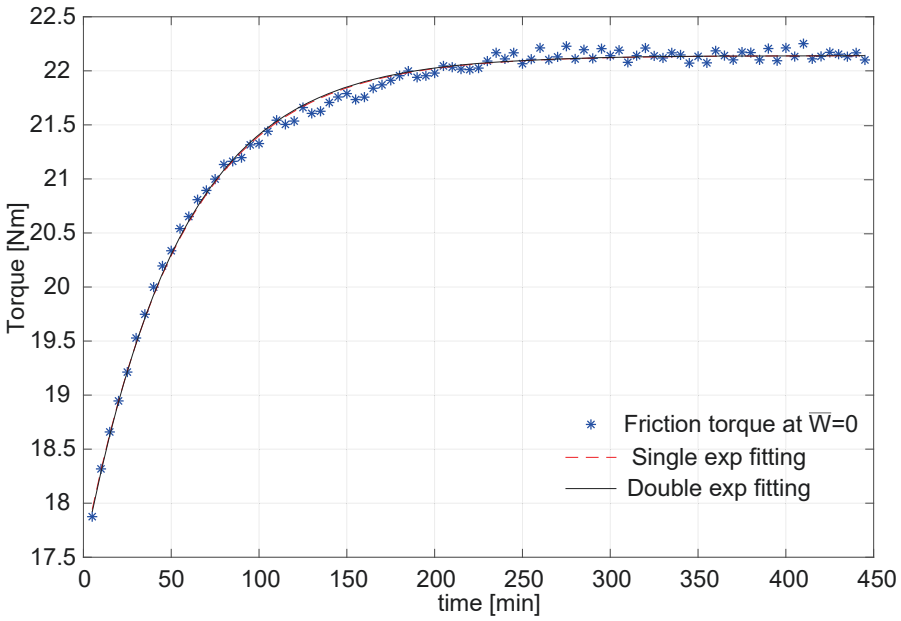


Figure 28: Friction torque versus time at 60% of maximum speed during the cooling. Experimental data, axis 4. Comparison between 1st and 2nd thermal model, see also Table 7

the obtained results are provided below. The identified thermal data for axis 4 are:

$$\begin{aligned} &\alpha = -0.0141\left[\tfrac{1}{K}\right] \quad K_1 = 4.41\left[\tfrac{W}{K}\right] \quad K_2 = 4.57\left[\tfrac{W}{K}\right] \\ &\beta = 0.9971 \qquad\quad C_1 = 737.61\left[\tfrac{J}{K}\right] \quad C_2 = 9163.45\left[\tfrac{J}{K}\right] \end{aligned} \tag{130}$$

Figure 24 shows the comparison between the calculated and measured torque during the heating test for some duty cycles. By calculating the eigenvalues of Equation (120) at different input power, it is possible to know the trend of the time constants represented in Figure 25. Figure 26 shows the τ_{f_M} versus time of axis 4 during the heating test at the maximum duty cycle, which corresponds to an input power equal to $\overline{W} = 97.65[W]$. For the other axes, the trend is similar. It is important to note that it is necessary to use a double exponential function to approximate the friction torque because a single exponential function does not fit the curve properly. The simpler model reaches the steady state much earlier than the second one, causing prediction errors on the torque estimation both during the transient and at the warm state. Table 6 shows an example of the difference between the errors obtained by the two models. The cooling tests started with the robot in its heated state. The test trajectory has been repeated only once every 5 minutes, keeping the robot stopped for the rest of the test time to cool it down. The collected data of this test highlighted a different behavior compared

Table 8: Thermal time constant at maximum duty cycle for each joint

Axis	t_{w1} [min]	t_{w2} [min]
1	22.0	2.5
2	27.3	1.9
3	21.1	2.3
4	27.6	2.3
5	25.8	2.5
6	23.9	2.0

to the previous one. By omitting the input power in the system, the first time constant becomes much shorter than the second one. Therefore, the friction torque could be approximated with a single exponential function, as shown in Figure 28. Table 7 shows another example of the calculated errors during the cooling test. Finally, Table 8 shows the time constants of each axis for the heating.

4.3.2 *Discussion*

At first, this section confirms the importance of friction in the prediction of the necessary torque to actuate a manipulator. Secondly, it is verified that friction may greatly change during the execution of robot tasks due to the increase of temperature throughout the operations. These changes are induced by the energy dissipated by friction. The friction appearing at high operating temperatures is even 50% more than the friction observed when the robot is in its cold condition. The results show that robot joints may reach the steady state condition after 60-80 minutes. Although the numerical results obtained refer to a specific robot, the model is general and can be applied to any industrial manipulator. Furthermore, the proposed methodology does not require the use of temperature sensors.

The main novelty of this study is the new thermal model of the relation between friction and joint temperature in mechanical transmissions. Compared to known models, this one includes two time constants and describes the behavior of robots that cannot be defined by simple models adopting only one time constant. The experimental test shows that this model fits well the theory and opens the route to the development of a forecasting model to predict the joint friction in any working cycle.

This will allow obtaining a full friction model which could be used for several research purposes. For example, (i) to improve dynamic models that ignore friction components in control applications for simplification purposes [85], (ii) predictive maintenance, (iii) to develop virtual force sensors [86], (iv) impact de-

tection and control of human-machine interaction without force sensors, and (v) to evaluate structural deformations occurring due to thermal effects.

4.4 TEMPERATURE EFFECT - FRACTIONAL MODEL

Models of the friction variation based on the knowledge of the robot characteristics and its working cycle have been presented in [73, 74]. The authors of [73] combined a linear relationship between friction coefficients and temperature with a first order differential equation that resulted in an exponential decay of friction during the robot working cycle. However, the experimental results carried out with the industrial manipulator Efort ER3A-C60 in [63] showed that a single exponential term might be insufficient to capture the evolution of friction over time, thus, a more elaborate model is required.

Fractional systems have attracted a great deal of attention in recent years thanks to their ability to describe complex dynamics by using lumped models and a small number of parameters [87, 88, 89, 90, 91, 92, 93]. Fractional systems are obtained by allowing the derivative orders in the differential equations to assume any real value [94, 95]. The modes of linear fractional differential equations are the so-called Mittag-Leffler functions, also referred to as generalized exponential [96]. These functions can capture a variety of different dynamics, ranging from monotonic to non-monotonic behaviors, and are particularly well-suited to describe diffusive phenomena that are normally characterized by a fast response followed by a slow-converging tail (namely, slower than exponential) [97, 94]. As such, fractional models have been successfully exploited in the modeling of thermal phenomena (see [98, 99] and the references therein).

In this section, we propose the use of a so-called fractional "first" order differential equation to model the relationship between the power generated by the friction torque and the temperature of the joint of the robot. Moreover, we describe the evolution of friction over time. Following the work outlined in [73], we adopted a linear model to describe the connection between the friction coefficients and the joint temperature. Combining such linear dependence with the fractional dynamic model, we derived the time evolution of friction when the manipulator executes a repetitive task. Such time evolution is expressed in terms of Mittag-Leffler functions. Finally, we experimentally validate the proposed approach by using the Efort ER3A-C60 industrial robot. The experimental results confirm the effectiveness of the proposed approach and show the advantages brought by the use of a fractional model.

The effect of temperature on the friction torque can be modeled by writing the thermal balance equation for the joint. The increment of temperature is related to the difference between the thermal power generated inside the joint due to friction, and the thermal loss due to dissipation. In [21, 76], the simple first-order linear model is defined as:

$$\frac{dT(t)}{dt} = \frac{w_i(t) - w_o(t)}{C}, \tag{131}$$

where w_i is the power generated by friction, w_o is the power dissipated in the environment, and C is the thermal capacity of the joint. This model was successfully used to model friction on a commercial Comau manipulator [21, 76], however, it fails to adequately capture the dynamics of friction when adopted on the Efort ER3A-C60 robot used in this thesis. This is probably because Equation (131) is based on the assumption that the temperature inside the joint is uniform, which may not be the case according to (i) the geometry of the joint, (ii) the fluid dynamics of the lubricant inside the joint, and (iii) the heat conduction properties of the joint. Experimental results show that the friction evolution over time is characterized by an extremely rapid decrease during the first part of the transient response, followed by a slow-decaying tail. Such behavior is typical of fractional models [94]. As such, we consider the fractional analogue of (131):

$$\frac{d^\nu T(t)}{dt^\nu} = \frac{w_i(t) - w_o(t)}{C}, \tag{132}$$

where the derivative order ν ranges between 0 and 1, and C is an appropriate constant.[2] A detailed discussion about fractional models and fractional differential equations goes beyond the scope of this work, thus we refer the reader to [100, 101]. We limit ourselves here by noting that the response of a fractional differential equation of the form:

$$\frac{d^\nu x(t)}{dt^\nu} = \lambda x(t) + u(t),$$

to a step input, i.e., $u(t) = \mu$ for all $t \geqslant 0$, from zero initial conditions, is:

$$x(t) = -\frac{\mu}{\lambda}\big(1 - E_{\nu,1}(\lambda t^\nu)\big), \tag{133}$$

where $E_{\nu,\beta}(z) = \sum_{k=0}^{\infty} \frac{z^k}{\Gamma(\nu k+\beta)}$ is a Mittag-Leffler function, and Γ is the Euler gamma function. See [100] for more details.

We now exploit the fractional system to model the behavior of friction. The input thermal power depends on the friction according to:

$$w_i(t) = \tau_f\big(T(t)\big)\dot{\theta}(t), \tag{134}$$

2 Note that C is no longer the thermal capacity of the joint, but a different constant with dimension $J\,s^{\nu-1}/K$.

while the dissipation obeys the equation $w_o = K\big(T(t) - T_e\big)$, where T_e is the temperature of the environment (assumed as constant), and K is the heat exchange coefficient between the joint and the external environment. Substituting the previous equations into (132) we obtain:

$$\frac{d^\nu T(t)}{dt^\nu} = \frac{\tau_f\big(T(t)\big)\,\dot\theta(t) - K\big(T(t) - T_e\big)}{C}. \tag{135}$$

We now assume that the coefficients in (88) are linear functions of temperature, hence:

$$a_i\big(T(t)\big) = a_{ie}\big(\alpha(T(t) - T_e) + 1\big), \quad i = 0, \dots, 3, \tag{136}$$

where a_{ie} is the value of the coefficient a_i at ambient temperature T_e and $\alpha < 0$ models the decrease of friction when the temperature increases. Note that the linearity assumption between friction and temperature has been justified by experiential evidence in [76]. Substituting (136) into (88) we easily obtain:

$$\tau_f(t) = \tau_{fe}(t)\big(\alpha(T(t) - T_e) + 1\big), \tag{137}$$

where $\tau_{fe}(t) = a_{0e}\mathrm{sgn}\big(\dot\theta(t)\big) + a_{1e}\,\dot\theta(t) + a_{2e}\,\dot\theta^2(t)\,\mathrm{sgn}\big(\dot\theta(t)\big) + a_{3e}\,\dot\theta^3(t)$ is the friction torque at ambient temperature. We now use (137) in (134) to obtain:

$$w_i\big(T(t)\big) = \tau_{fe}(t)\,\dot\theta(t)\Big(\alpha\big(T(t) - T_e\big) + 1\Big) \tag{138}$$

and we may rewrite (135) as:

$$\frac{d^\nu T(t)}{dt^\nu} = \frac{\tau_{fe}(t)\,\dot\theta(t)\Big(\alpha\big(T(t) - T_e\big) + 1\Big) - K\big(T(t) - T_e\big)}{C}. \tag{139}$$

The term $\tau_{fe}(t)\,\dot\theta(t)$ depends on the joint velocity $\dot\theta(t)$. However, in the vast majority of industrial applications, the robot repeats the same periodic trajectory multiple times. Therefore, the term $\tau_{fe}(t)\,\dot\theta(t)$ is periodic, and the period is normally much smaller than the thermal time constants of the system. Thus, we can neglect the variation over time of $\tau_{fe}(t)\,\dot\theta(t)$, and instead use the average value

$$w_{i0} = \frac{1}{t_c}\int_{t_c} \tau_{fe}(t)\,\dot\theta(t)\,dt,$$

where t_c is the robot cycle time. Substituting $\tau_{fe}(t)\,\dot\theta(t)$ with w_{i0} in (139) we obtain:

$$\frac{d^\nu T(t)}{dt^\nu} = \frac{w_{i0}\Big(\alpha\big(T(t) - T_e\big) + 1\Big) - K\big(T(t) - T_e\big)}{C},$$

which is easily seen to be equivalent to:

$$\frac{d^{\nu}T(t)}{dt^{\nu}} = \lambda\, T(t) + u(t), \tag{140}$$

with $\lambda = \frac{w_{i0}\,\alpha - K}{C}$ and $u(t) = \frac{K\,T_e + w_{i0}\,(1-\alpha\,T_e)}{C}$ constant. By using (133) we compute the temperature response:

$$\begin{aligned} T(t) \;=\; & \frac{K\,T_e + w_{i0}(1-\alpha\,T_e)}{K - w_{i0}\,\alpha}\left(1 - E_{\nu,1}\left(\frac{w_{i0}\,\alpha - K}{C}t^{\nu}\right)\right) \\ & + T_e\, E_{\nu,1}\left(\frac{w_{i0}\,\alpha - K}{C}t^{\nu}\right). \end{aligned} \tag{141}$$

Note that when $\nu = 1$ we have $E_{1,1}(z) = e^{z}$, so that the solution proposed in [76] is a special case of the more general setting presented here. Finally, we compute the evolution of friction by using Equations (141) and (137):

$$\frac{\tau_f(t)}{\tau_{fe}(t)} = \frac{w_{i0}\,\alpha}{w_{i0}\,\alpha - K} E_{\nu,1}\left(\frac{w_{i0}\,\alpha - K}{C}t^{\nu}\right) + \frac{K}{K - w_{i0}\,\alpha}. \tag{142}$$

For a given speed, the proposed model results in a transient response of friction that exhibits a Mittag-Leffler type decay. In the next section, this model has been validated using real-world data.

4.4.1 *Fractional Model Validation*

The experimental setup is exactly the same as section 4.3.1. To reduce measurement uncertainty, for each different speed the average value of the friction torque over the duration t_s of each trait at constant speed has been considered:

$$\tau_{f_M} = \frac{1}{t_s}\int_{t_s} \tau_f \, dt. \tag{143}$$

First, we examine the behavior of friction over time. We focus on a single joint considering only the case when the joint is moving at 60% of its maximum speed. The experimental data are fitted both by using a Mittag-Leffler function and a simple exponential model, obtained by constraining $\nu = 1$ [21]. The results are shown in Figure 29, where the friction torque τ_{f_M} versus time for axis 4 is plotted. It is clear that the Mittag-Leffler function provides a much better fitting of the experimental data, thus confirming that the fractional model can be effectively used to describe the heating transient of the joint, and ultimately the variation of friction over time. In particular, the best fitting is obtained with a fractional order

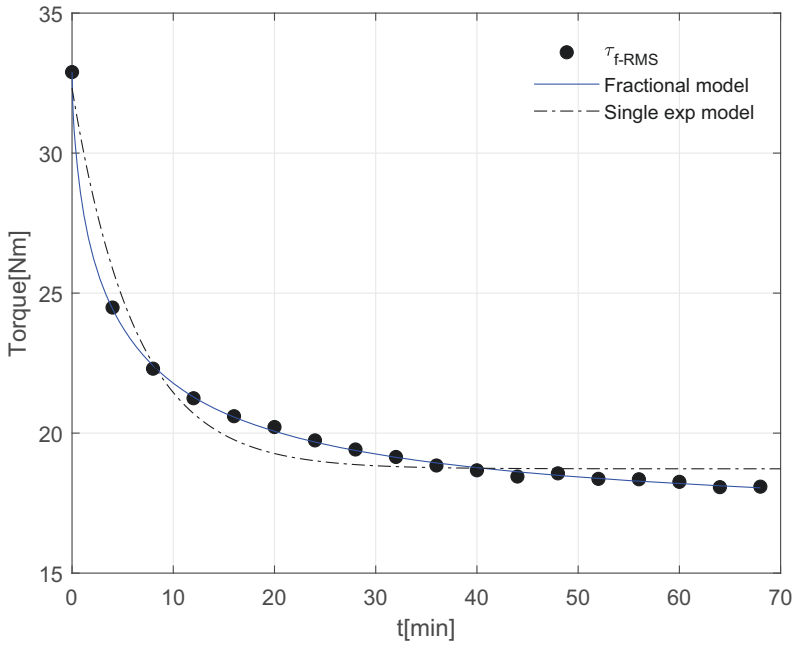


Figure 29: Friction torque at 60% of the maximum velocity for axis 4. Experimental data: dots; fractional model: solid line; integer model: dashed line.

$\nu = 0.6121$, which is different from 1, indicating that a simple first-order model is not sufficient to capture the dynamics in our experimental setup.

Finally, we tested different speeds. Figure 22 shows the friction versus temperature behavior at different velocities for a specific joint. The normalized torque is the ratio between the friction torque at each time instant and the friction torque in cold conditions (the highest value, so that all the curves start at 1). The curves at different speeds are virtually coincident: this result validates the hypothesis of linearity of the relationship between the friction coefficients and temperature, i.e. Equation (136), and shows that the fractional model works well at all the different velocities. The other joints exhibit similar behavior.

4.4.2 *Discussion*

This section presented a technique to model the behavior of friction occurring in the joints of robotic manipulators. The proposed solution hinges on a linear relationship between friction and temperature combined with a fractional thermal model of the joint. This solution is experimentally validated in [78], showing the advantages of the fractional system with respect to a classical integer model. The proposed solution can be conveniently exploited to estimate and compensate for friction in the frequent situation where a direct measure of the temperature of the joint is not available.

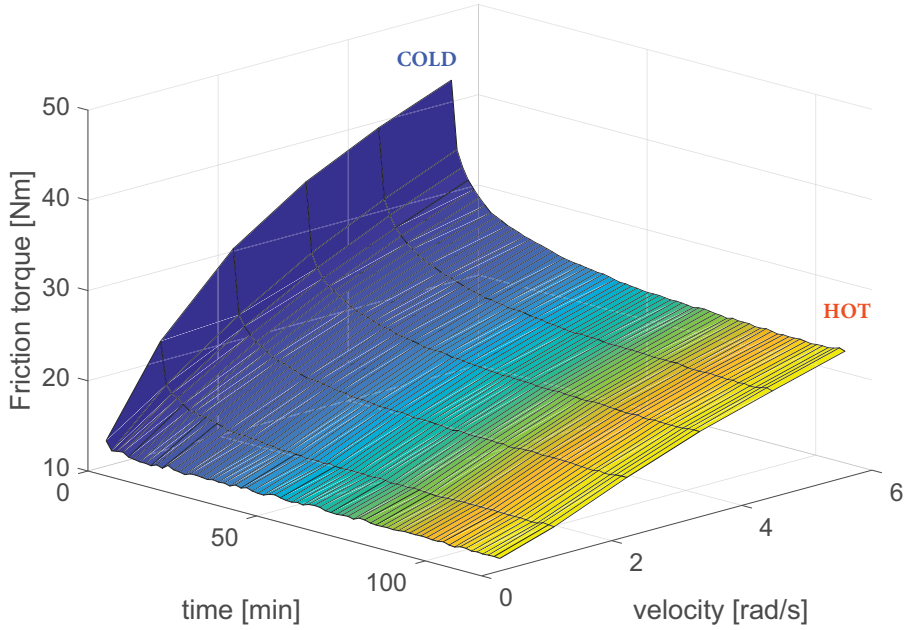


Figure 30: Friction vs positive velocity only during the warming up.

4.5 TEMPERATURE EFFECT - ADVANCED SECOND ORDER MODEL

Several mathematical models can be used to analyze the heating of robotic joints and the consequent variations of friction [102, 103, 75, 77, 63, 73, 74]. However, they typically do not consider all the heat sources (friction, Joule effect, eddy currents), thus, they fail to correctly evaluate thermal dissipation. This section contributes to the topic of friction changes in robotic joints due to thermal effects. Through mathematical modeling based on heat exchange equations, it is demonstrated that it is possible to predict the variations of friction by knowing the characteristics of the robot and its work cycle. As already stated before, friction produces heat that gradually increases the temperature of the mechanical transmission, making the oil less viscous and progressively reducing the friction torque. Figures 14 and 30 show the variation of the friction torque as a function of speed in the gearbox of an industrial robot. Each curve has been obtained at a different temperature of the gearbox, performing the first test starting with a temperature equal to the environmental one (cool condition). The trend of the curves is increasing with the speed and, for a given speed value, the friction torque is lower when the transmission has been heated due to previously performed working cycles. Therefore, to correctly evaluate the effect of temperature on friction, a thermal balance equation must also be included in the calculations, as will be described in the following subsections.

Energy Dissipation in Electric Motors

Electric actuators, like all electric devices, dissipate part of the energy into heat. A portion of this heat is stored as internal energy increasing the temperature of the motor, while the other is discharged to the external environment. During the thermal transient that is typical of operating cycles with variable torque and speed, both these phenomena occur. However, when the motor reaches the thermal regime all the heat produced is released outside the machine. The causes of heat dissipation are mainly two: (i) the moving current in the copper windings (Joule effect), and (ii) the induction of eddy currents of Foucault in the ferromagnetic material in the presence of a variation of the magnetic flux that passes through it. For the brushless AC servomotors which drive the robot joints, the losses in the copper occur in the stator and those in the iron occur in the ferromagnetic material of the rotor. If I_{inst} is the instantaneous value of current intensity in the stator windings and R_f the phase resistance, the instantaneous power dissipated by the Joule effect ($W_{d,Cu}$) is equal to:

$$W_{d,Cu} = 3R_f I_{inst}^2 \tag{144}$$

If we enter the RMS value in expression (144) instead of the instantaneous current value, we obtain the average value over time of the power dissipated. In ferromagnetic materials that are traversed by a magnetic flux that varies over time, eddy currents of Foucault and hysteresis phenomena are induced which cause a power dissipation [104] that increase with the value of the magnetic induction and with the frequency of flux variation, according to:

$$W_{d,Fe} \simeq \lambda_1 B^2 f^2 + \lambda_2 B^\alpha f \tag{145}$$

In this equation, B is the maximum value of the magnetic induction, f is the flux frequency variation, λ_1, λ_2, and α are constants depending on the system materials and geometry. The first term of expression (145) represents an approximation of the losses due to eddy currents, while the second takes into account hysteresis phenomena. A further phenomenon that is important to remember is the influence of temperature on the electrical resistance of the windings. An increase of 100 °C in the copper temperature causes a 39% increase in the electrical resistance. Therefore, the power dissipated in the copper varies as the motor temperature changes. The Joule effect depends on the current, which is proportional to motor torque. Torque is necessary to generate motion, but also to keep the robot still in position, thus, it also depends on static friction. Frequently, to keep a joint still, the controller generates oscillating currents (torque) presenting zero average value and an RMS value different from zero. As a result, heat is generated.

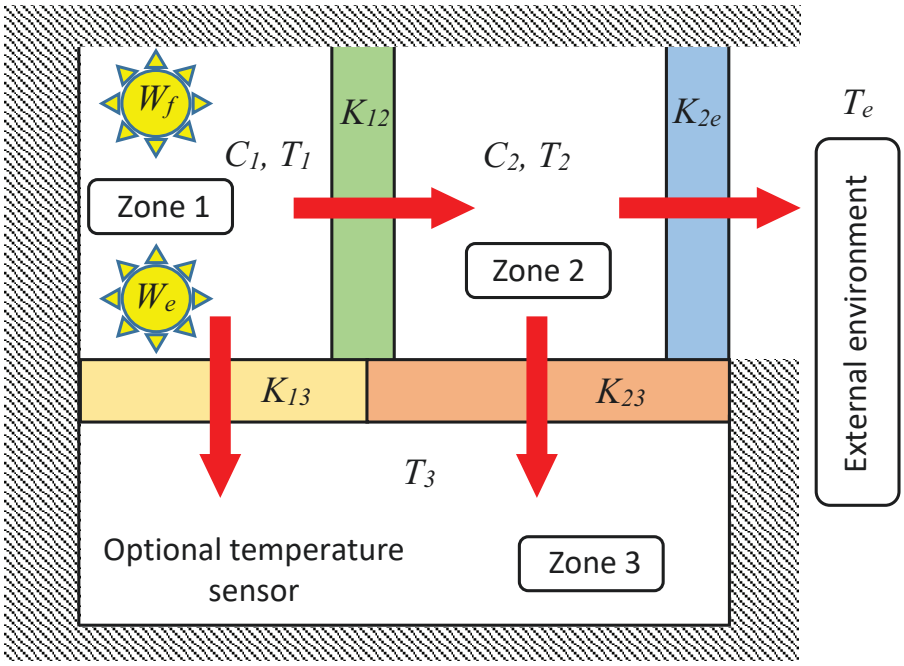


Figure 31: Thermal flows in the robotic joint.

Heat Exchanges Modelling

The variation of the friction torque according to temperature can be studied through suitable equations of thermal balance. This approach has already been used in previous studies carried out in [77, 63], but those models do not take into account the thermal losses. Therefore, a new thermal model has been defined described in detail in this section. Figure 31 shows the thermal power that flows in the robotic joint. The symbols C_i ($i = 1, 2$) and T_i are the thermal capacities and the temperatures of zones 1 and 2, respectively. Temperature T_3 of zone 3 can be optionally measured using a thermocouple. The temperature of the external environment is indicated by T_e, while K_{12}, K_{2e}, K_{31}, and K_{32} indicate the heat exchange coefficients between the walls that delimit the various areas. In zone 1, there is the generation of thermal power deriving from two different sources, indicated with yellow circular symbols. The first source W_f is the friction generated between the moving parts inside the joint (W_{in} in Equation (106)), while the second source W_e indicates the power dissipated in the electric motor due to the Joule effect and to the eddy currents which, as is known, are always present inside an electric motor. Therefore, the total thermal power generated inside the joint is:

$$W_{tot} = W_f + W_e = (aT_1 + b) + W_e \tag{146}$$

The heat balance equations between zones 1 and 2 of the joint can therefore be written in the following form:

$$\begin{cases} \dfrac{dT_1}{dt} = \dfrac{(aT_1 + b) + W_e - K_{12}(T_1 - T_2) - K_{13}(T_1 - T_3)}{C_1} \\ \\ \dfrac{dT_2}{dt} = \dfrac{K_{12}(T_1 - T_2) - K_{23}(T_2 - T_3) - K_{2e}(T_2 - T_e)}{C_2} \end{cases} \tag{147}$$

Using matrix notation, these equations can be rewritten as:

$$\dot{\mathbf{y}} = \mathbf{A}\mathbf{y} + \mathbf{b} \tag{148}$$

where

$$\mathbf{y} = [T_1 \quad T_2]^T \tag{149}$$

is the state vector containing the temperatures,

$$\mathbf{A} = \begin{bmatrix} \dfrac{a - K_{12} - K_{13}}{C_1} & \dfrac{K_{12}}{C_1} \\ \\ \dfrac{K_{12}}{C_2} & -\dfrac{K_{12} + K_{2e} + K_{23}}{C_2} \end{bmatrix} \tag{150}$$

is the system matrix, and

$$\mathbf{b} = \begin{bmatrix} \dfrac{K_{13}T_3 + b + W_e}{C_1} \\ \\ \dfrac{K_{23}T_3 + K_{2e}T_e}{C_2} \end{bmatrix} \tag{151}$$

is the input vector. If $\overline{W}$ and W_e are constant, the differential equation is linear and the thermal time constants are the inverse of the eigenvalues of the matrix $\mathbf{A}$, obtained by the characteristic polynomial

$$s^2 + ps + q = 0 \tag{152}$$

where

$$\begin{aligned} p &= \frac{K_{12} + K_{13} - a}{C_1} + \frac{K_{12} + K_{2e} + K_{23}}{C_2} \\ q &= \frac{(K_{13} - a)(K_{2e} + K_{23}) + K_{12}(K_{2e} + K_{13} + K_{23} - a)}{C_1 C_2} \end{aligned} \tag{153}$$

The solutions s_1 and s_2 of (152) are the two thermal time constants of the system, i.e. $t_{w1} = -1/s_1$ and $t_{w2} = -1/s_2$.

4.5.1 *Advanced Model Validation*

Compared to section 4.3.1 and [63], the experiments have been performed to explicitly highlight the contribution of the electrical power on the friction estimation. In fact, experience showed that if motors are powered on even if the robot is not moved, the temperature in the joint transmissions significantly increases, affecting friction (see Figure 33 in the next sections). Therefore, a genetic algorithm has been developed based on the acquired data to address both the heat generated by (i) friction and (ii) electrical effects. As a result of experimental observations, the term W_e of the model has been considered constant and experimentally estimated.

Genetic Algorithm

The idea behind a genetic algorithm (GA) reflects the process of natural selection, where only a subset of individuals survive and produce a new generation. Ideally, new generations inherit the best traits from their parents that allow them to be better suited for survival. The more this process is completed, the more the latest generation traits suit the "environment" to guarantee them the best survival and reproduction chances. This idea has been applied to optimization problems, where the individuals are a set of solutions for a given problem, and the environment is the real phenomenon to be described. At the start of the optimization procedure using a GA, an initial population of solutions is selected. Each solution is characterized by a set of parameters called genes. At each iteration of the GA, a subset of solutions is selected for reproduction based on their fitness score, which describes how well the solution performs when used to describe the real phenomenon. The selected individuals are combined to produce new individuals with better genes, and this process is repeated until a solution that fits the problem (according to a set of user-defined characteristics) has been found. The idea of using this type of algorithm to solve complex problems has already been exploited in other works [21, 105]. Knowing the values of friction power (W_f), the environmental temperature (T_e), the temperature measured by the thermocouples (T_3) and the real torque values measured during the experiments, it was possible to use a genetic algorithm to find all the unknown parameters of Equation (147) by minimizing the average squared difference between the estimated values and the actual value. Some tests have been performed starting from initial cold conditions (further denoted as power-off tests) and others after warming up the robot by simply switching on the motors (further denoted as power-on tests). So, the test names <power ON/OFF> simply denote the state of the electrical actuators for the hours preceding the warming up tests.

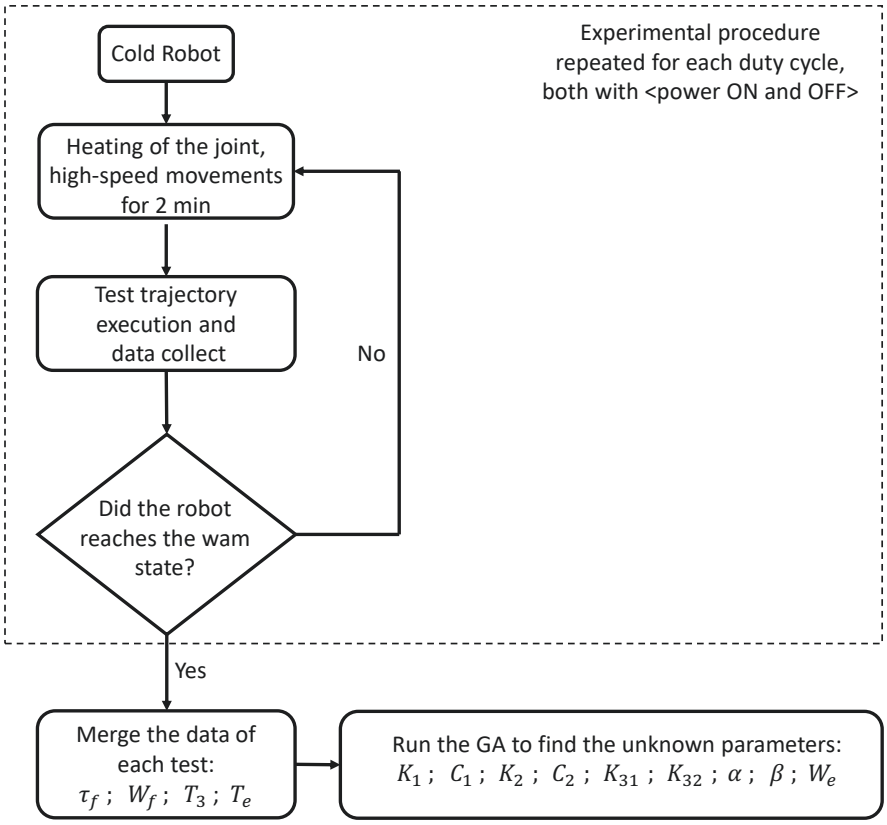


Figure 32: Experimental procedure flowchart.

Experimental data

The robot used in the experiments is the usual Efort manipulator described in chapter 3. The data necessary to run the genetic algorithm has been acquired actuating the robot joints separately using a specific test trajectory. The gravitational contribution in the joint torque measure has been purposely avoided, adopting specific robot configurations or by subtracting the gravitational contribution to the measured torque of joints. During the experiments, data have been collected by actuating the robot joints after more than 12 hours of inactivity with no power, to ensure that the motor's temperature at the start of the experiment was equal to the environmental temperature (cold-motors state). The robot performed first the same test trajectory that has been shown in Figure 19, and the corresponding friction torque data of the joint has been acquired. It is worth noting that the test trajectory presents several portions at a constant velocity, in which only the friction component exists. Then, the robot performed a point-to-point trajectory at the maximum velocity for a total of 2 minutes with different duty cycles (warming-up phase). Afterward, the test trajectory is performed again to collect new data. This step has been repeated until the robot reached the thermal regime.

See Figure 32 for clarifications. Figure 23 in section 4.3.1 shows the different duty cycles adopted to warm-up the robot. In this experiment, only the data coming from D = 20%, 60%, and 100% has been used, while the input average thermal power of a cycle follows Equation 129. It is worth noting that t_p is very low compared to the thermal values of the mechanical system, allowing us to consider the average power value. Time constants of the robot are in the range of a few minutes to dozens, while t_p is in the order of some seconds. To evaluate the contribution of energy dissipation in the electric motors, another set of experiments has been performed. The procedure is the same as the previous one, but in this case, the first set of data has been acquired after a 12h of inactivity with no power (cold-motors state) and a subsequent time of inactivity with power on until the thermal motor regime was reached. This allowed us to highlight how the Joule effect contributes to the heating of the motor. Additional thermal data has been acquired from some thermocouples mounted inside the external cover of the robot, very close to the motor transmission. The thermocouple data T_3 in (116) has been acquired in both experiments but was especially useful to highlight the heating due to electrical losses in the motors. It is worth noting that this effect reduces the robot power consumption during the first robot movements since the motors are not in the cold state but have been warmed up by the Joule effect. Figure 33 shows how T_3 changes over time when only the Joule effect is considered. This finding was the reason to change the model from the one adopted in previous work [63] to the one described in this section.

Figure 34 provides an example of the torque decrease of a specific joint during power ON and power OFF tests, i.e. starting the test from a completely cold state and after reaching the thermal regime of the motor due to the electrical dissipation. It is worth noting that the rise in temperature causes a significant decrement in the friction torque at the initial time, confirming the necessity to add a term dependent on electrical power in the mathematical model. Figure 35 shows the thermal power injected by friction for three different duty cycles during the power-on and power-off test. It is worth noting the difference caused by the warm-up. Figure 36 shows the effectiveness of the model in predicting the collected data.

4.5.2 *Discussion*

The advanced second order thermal model presented in this section has two main differences with respect to other existing models. First, the thermal power generated by the electrical motors is considered in this case. Experiments performed on an industrial robot show that in this case, although this energy is smaller than the one produced by friction, it still influences the temperature of the joints and the friction value. These results are confirmed by simulations and by real temperature measures taken in the inner part of the joint using thermo-

couples. Since it is impossible to measure a very internal part of the joint, we had to place the sensors on a nearby component. The second difference is the adoption of such thermocouples. Measuring internal joint's heat is not strictly mandatory but it may help because (i) it helps to make a more robust identification of the parameters of the models (transmission constants, thermal capacities), usually based only on friction measures, and (ii) the internal temperature of the joint measured is useful to better predict the friction value. This could be useful when external forces are acting on the robot, where the torque measures are not enough to discern between the contributions of external forces and temperature changes [86]. The realization of the observer was outside the scope of this work but will be developed in future advances. In the case of cheap applications, if the observer is not adopted sensors can be temporarily inserted just during the identification phase. The presented model is general and allows understanding the thermal power flow. It can be applied to any robot and it can be easily implemented. Its adoption permits the estimation in real-time of the friction value and its compensation in the control scheme, thus achieving better control performances. Furthermore, by using this model it is possible to easily adapt the robot performances to extreme environmental conditions such as tropical regions or very cold ones. Other applications include the prediction and the optimization of the energy consumption and the valuation and comparison of different lubricants.

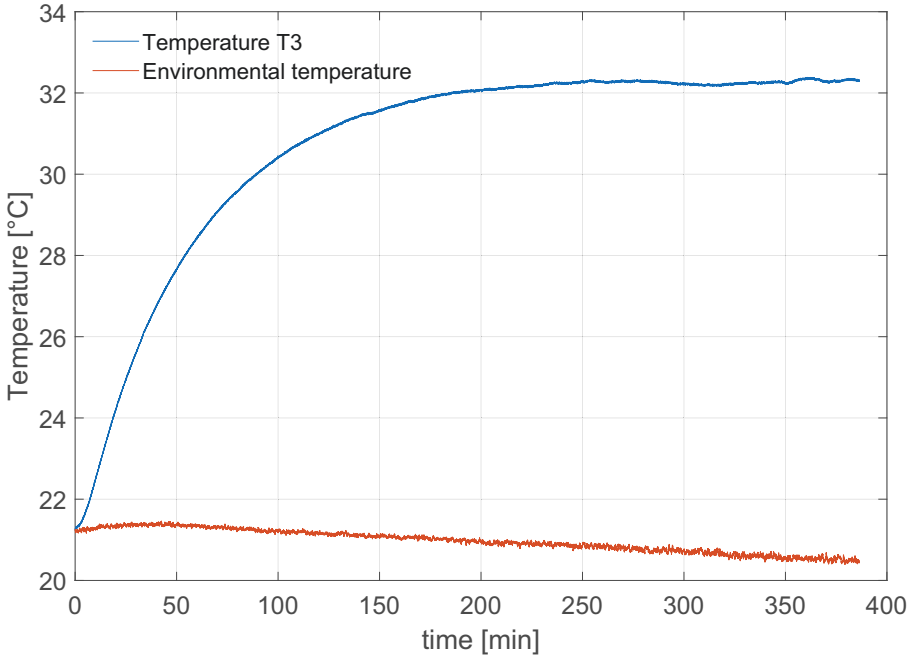


Figure 33: Example of increasing temperature due to electrical effect, experimental data at power on with no joint motion, axis 4.

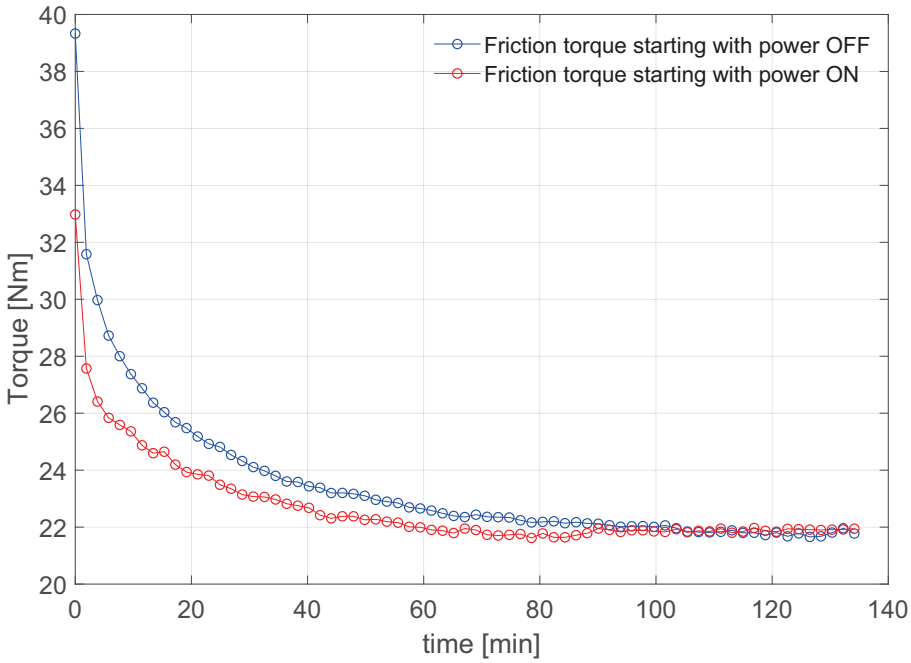


Figure 34: Friction torque versus time at 80% of maximum speed and d=100%, starting at power on and off, axis 4.

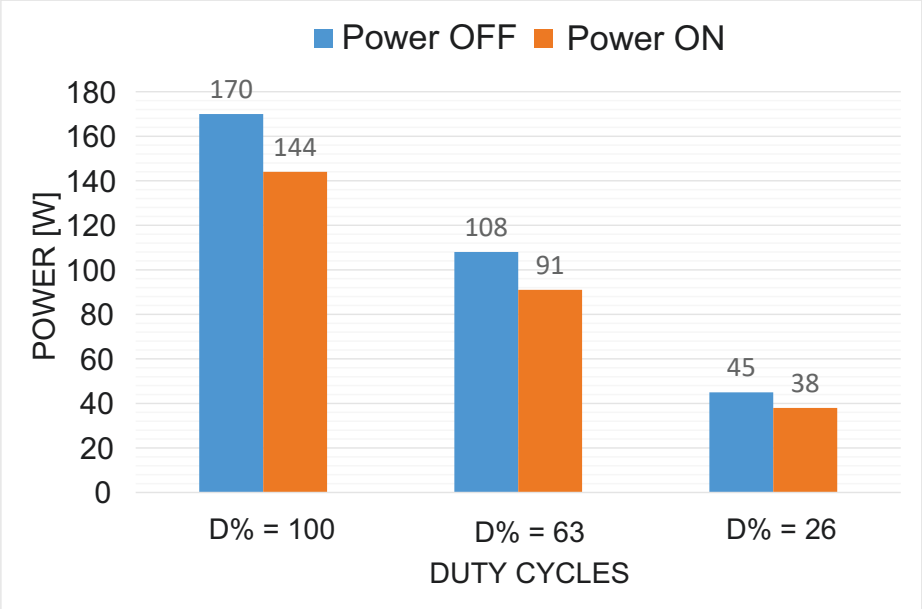


Figure 35: Friction input power at first cycles.

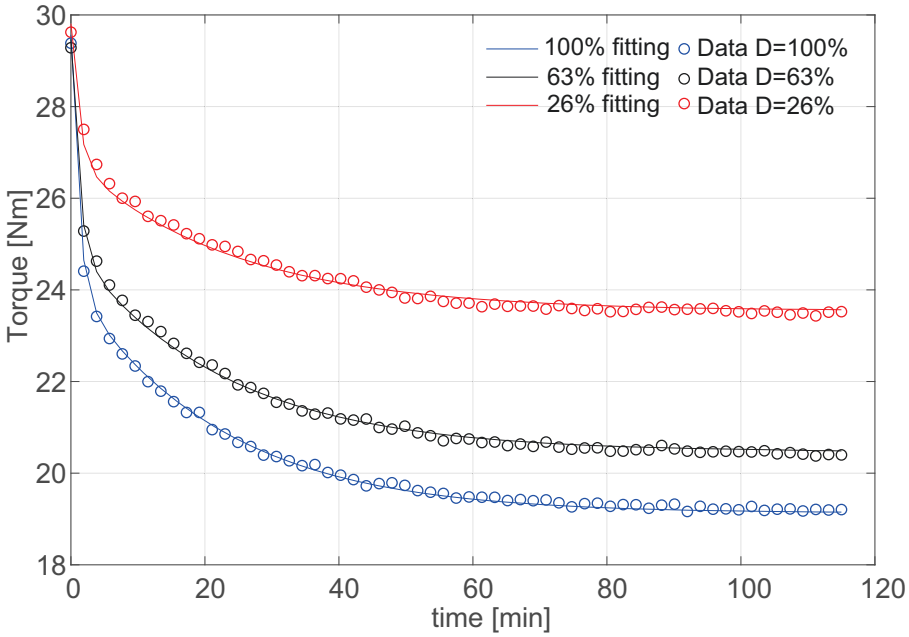


Figure 36: Friction torque fitting example with different duty cycles, experimental data axis 4, power on.

5

IDENTIFICATION, COMPARISON AND REPETITIVENESS ANALYSIS ON 2 ROBOTS

As already stated at the beginning of this thesis, the study of the dynamical performance of robotics machines is one of the most relevant problems in the design of effective robot control strategies. Parameter identification allows obtaining reliable dynamic models of industrial manipulators [106]. It is worth pointing out that a reliable dynamic model can improve position control [107]. Other applications may concern predictive maintenance, trajectory planning for energy efficiency [108], safety systems for robotic cells [109], or management of robot interaction with humans or the environment [110, 111]. Many of these applications may require the knowledge of a good dynamic model of the robot. The robot's dynamical parameters include inertial parameters (masses, centers of mass, and moments of inertia) and friction description. Joints and transmissions are considered rigid, which is acceptable for many industrial applications. The inertial parameters can be obtained by developing a model and by performing experiments on the robot. The authors of [112, 113, 29] showed that it is possible to define a set of parameters that describe the model (torque prediction) for any open-chain robot. The model is linear on these parameters and their regression matrix depends on the position, velocity, and acceleration of the joints. M. Gautier and W. Khalil proposed to drop the parameters that scarcely influence then fitting, obtaining a full-rank matrix. The remaining dynamical parameters involved in the identification are called the base parameters (see Chapter 1).

The identification process uses the Least Square Method, which provides reliable performance when models are linear in the parameters. During the test, the robot is moved on a properly designed excitation trajectory storing the position, velocity, acceleration, and torque of the joints. Then, the parameters that better predict the measured torques are estimated. Several criteria have been proposed to optimize the trajectory [37, 114]. In this work, the excitation trajectory is designed based on the finite Fourier series introduced by J. Swevers [115], with the benefit of data averaging in the time domain and reducing the measurement noise effect. An additional trajectory is appended to the excitation movement to guarantee the reaching of the joint-velocity bounds (see Section 2.3).

The linearity in the parameters does not hold in long time intervals, because the friction might change due to thermal effects. This model mismatch is evident comparing two experiments, one executed at the robot startup and one after the thermal transient. Friction is one of the most relevant undesired phenomena to be taken into account to realize a reliable dynamic model. It arises when there

is a relative motion between two surfaces in contact or when a body moves inside a fluid. Friction depends on the geometry and the roughness of the surfaces, their materials, the type of lubricant, the velocity, the humidity, and so on [116]. Friction causes the loss of energy, heating of the surfaces in contact. This phenomenon introduces nonlinearities that might cause several problems related to motion control, such as trajectory tracking errors, limit cycles, and dynamic instabilities (stick-slip). For this reason, many friction models have been proposed in the technical literature (see [117]) to provide a suitable description of the physical phenomenon and to obtain effective mathematical formulations.

Regarding the dynamic models, on the other hand, the friction force depends on a state variable that takes into account the history of the system and not only its current situation. Relevant examples are the Dahl model [118], the Maxwell-slip model [119] and its extensions [120, 121, 122]. and the LuGre model [123, 124, 125]. In general, friction models are used in the control strategies to compensate for the friction effects (see [126, 67]).

Despite the numerous efforts supported by research, this phenomenon is still a relevant issue for the control of industrial robots, and it depends on other factors than velocity. For this reason, some researchers started to focus on the role of temperature in friction [84, 75, 127]. Temperature is very important for those robot operations that require the manipulator to often stop for a long time, letting the joint temperature cool down and changing the friction characteristics. As explained in [128, 129], friction transforms mechanical energy into heat, which causes a temperature increase [130], and therefore the behavior of the speed reducers changes too [131]. The installation of temperature sensors may be unpractical due to cost and compactness requirements. Thus, a possible solution to observe the relationship between friction and temperature could be based on the output of motor torque, which is easy to collect during the movement. (see Chapter 4)

Opposite to friction parameters, inertial parameters should be independent of temperature, but a weak identification procedure and model mismatches may produce the wrong estimation of inertial parameters in function of the robot temperature. The same identification procedure can be carried out on multiple robots to verify the differences between machines and to verify the robustness of the identification. While experiments generally discuss experimental data validated on one robot only, in this thesis, we analyze multiple tests on two industrial robots (two replicas of the same model) to verify the repeatability of the results. This is important for establishing robust industrial applications. A comparison analysis allows understanding the differences between the dynamical and the friction behaviors changing among the robots. The results of the comparison would be useful for future improvement in manufacturing and maintaining of the machines. A further contribution of this work is an analysis about the repeti-

tiveness of the parameters estimation in several repetitions performed in different days and during the warming stage of the robot.

The results of this chapter have been presented in [132]

5.1 DATA ACQUISITION AND ELABORATION

Two robots of the same model EFFORT ER3A C-60 (chapter 3) were used for the experiments, named "Robot 1" and "Robot 2". The data acquisition was repeated four times for each robot on different days. To have comparable environmental conditions, all tests were executed in the morning after at least 8 h of robot rest and with an environmental temperature of about 20 °C. For each movement, a PC-based acquisition system collected the motor's positions and torques with a frequency of 4 ms. After the data collection, identification and analysis were developed using MATLAB programming environment. The data collected during all the repetitions of the 24 different cycles were analyzed to estimate inertial and friction parameters.

As already stated in chapter 4, the friction parameters are highly dependent on temperature. The temperature of the gearbox is not available in most of the industrial robots in the market. Thus, it is necessary to estimate both the temperature and the thermal model from the experimental data. The estimation is possible by executing several warm-up tests with different root-mean-squared values of the joint torque, as proposed in [21] for a first-order thermal model and extended by [77] for second-order models. The mathematical formulation is already explained in section 4.3, but to better understand this phenomenon it is summarized below. The second-order thermal model has two thermal capacities, which exchange heat linearly with the temperature differential. The exchanged heat with the environment is assumed linearly dependent on the temperature of the thermal capacities. Finally, the heat generated by friction is the product $W_f = \tau_f \dot{\theta}$. Thus, the temperature evolves as described by the following transfer function:

$$\frac{T(s)}{W_f(s)} = \frac{c_1}{s + \tau_1^{-1}} + \frac{c_2}{s + \tau_2^{-1}} \tag{154}$$

where s is the Laplace variable, T is the gearbox temperature, and τ_1, τ_2, c_1, and c_2 are model parameters.

According to [63], when one robot is actuated on a repetitive working cycle, sufficiently shorter than the time constants t_1 and t_2, the joint temperatures changes as the step response of (154) where the heat is given by the root-mean-square value $W_{f,rms}$ computed in the working cycle:

$$T = T_0 + \left(c_1\tau_1 \left(1 - e^{-\frac{t}{t_1}}\right) + c_2\tau_2 \left(1 - e^{-\frac{t}{t_2}}\right) \right) W_{f,rms} \tag{155}$$

where t is the time and T_0 is the initial temperature. Given an estimated temperature, the friction changes as:

$$\tau_f = \tau_o \left(1 + \alpha \left(T - T_0\right)\right) \tag{156}$$

where α is a constant dependent on the thermal model and τ_o is friction torque at the initial temperature T_0. By combining the polynomial equation (88) with (156), we obtain a friction estimation formula as

$$\tau_f = \left(1 + \alpha \left(T - T_0\right)\right) \left(a_0 \operatorname{sign}\left(\dot{\theta}\right) + a_1 \dot{\theta} + a_2 \dot{\theta}^2 \operatorname{sign}\left(\dot{\theta}\right) + a_3 \dot{\theta}^3\right) \tag{157}$$

this relation is valid both for constant velocity situations as well as for transients, as described in Algorithm 1. Line 3 of the algorithm is the finite difference equation obtained by discretizing (154).

It is worth noticing that, in the special case of repetitive short cycles the friction torque changes as:

$$\tau_f = \left(1 + \alpha W_{f,rms} \left(c_1 \tau_1 \left(1 - e^{-\frac{t}{t_1}}\right) + c_2 \tau_2 \left(1 - e^{-\frac{t}{t_2}}\right)\right)\right) \tau_o \tag{158}$$

Thanks to (155) and (157), it is possible to identify the parameters α, c_1, c_2, τ_1, and τ_2 of the thermal model following the procedure described in [63], where a genetic algorithm tunes the model parameter to fit multiple warming cycles with different values of $W_{f,rms}$.

Algorithmus 1 : Evolution of the estimated temperature

Input : velocity $\dot{\theta}(k)$; estimated temperature $T(k-1)$ and $T(k-2)$ at the steps $k-1$ and $k-2$; discretization period δt

Output : estimated temperature $T(k)$ at step k

1 $\tau_f(k) = \left(1 + \alpha \left(T(k-1) - T_0\right)\right) \left(a_0 \operatorname{sign}\left(\dot{\theta}(k)\right) + a_1 \theta(\dot{k}) + a_2 \dot{\theta}^2(k) \operatorname{sign}\left(\dot{\theta}(k)\right) + a_3 \dot{\theta}^3(k)\right)$;

2 $W_f(k) = \tau_f(k) \theta(\dot{k})$;

3 $T(k)$ =thermal model$(W_f(k), T(k-1), T(k-2), \delta t)$;

Figure 37 shows the complete model used in torque estimation. The dynamic model (42) uses the position, velocity, and acceleration of the joints and the measure of external force (if any) to estimated the actual torque. The friction component τ_f uses the temperature prediction obtained by the thermal model thanks to Algorithm 1. A discrepancy between the estimation and the measured torque is inevitable; however, if this value is considerable, some other factor intervene: contacts with the environment or a malfunction of the mechanical transmissions. Specialized algorithms can deal with these situations (e.g., [133, 134]).

The comparison between the estimated torque and the measured one can be the input of virtual force sensors, collision detectors, or maintenance supervisor.

It is worth pointing out that, once the thermal model is estimated, it is possible

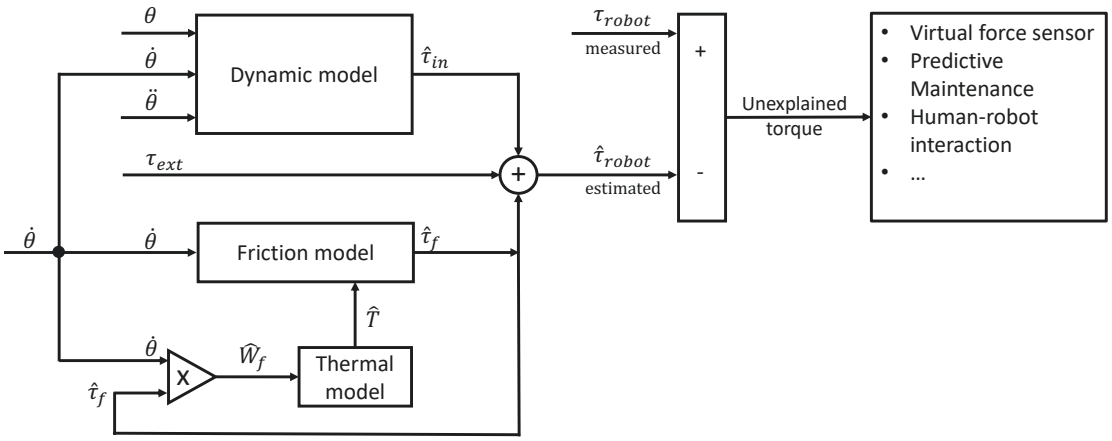
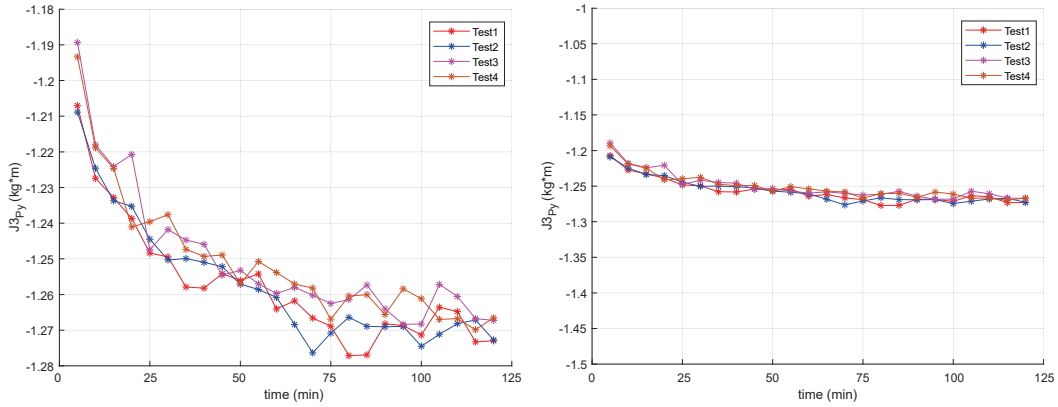


Figure 37: Scheme of the interconnections between the dynamic, the friction, and the thermal model and the possible use for advanced applications (predictive maintenance, virtual force sensor, and human–robot interaction). Symbols with the "hat" marks the estimated values, symbols without the "hat" are real values.

In order to fit the complete robot model, the data collected during experiments were analyzed to estimate inertial and friction parameters. Inertial parameters should be independent of the temperature and their numerical estimation would be expected to give constant results. The friction instead depends on temperature which increases with the robot activity, therefore a time-dependent trend with asymptotic behavior is expected for these parameters. The results show that the estimated dynamic parameters also had an asymptotic behavior; however, the range of the variation is generally minimal.

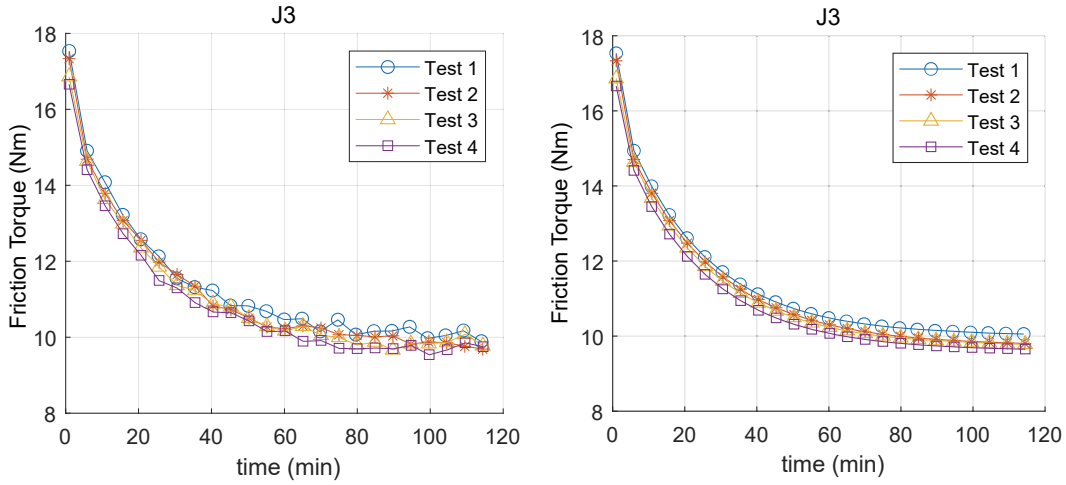
Examples of the results for inertial parameters are presented in Figure 38. The figure clearly shows that the estimated values of parameters change with cycles, but the variation is very little and the steady-state value is reached after about 15 cycles that correspond approximately to 1.5 h of the experiment.

The coefficient a_i of (88) change with behavior similar to (158). This trend is due to the model mismatches introduced by the thermal effect and the resulting identification error. In fact, the friction torque of Joint 3 at 60% of the velocity and its estimation are reported in Figure 39. It is worth pointing out that the friction torque at the cold state can be 70% higher than the final value, highlighting the relevance of the phenomenon.



(a) (b)

Figure 38: Experimental data. (**a**) Evolution of the identified values P_y of Joint 3 for all the tests on "Robot 2". (**b**) The evolution of the same parameter, but slightly widening the scale of the value. It is worth noting that the difference between the initial and final estimations is less than 5%.



(a) (b)

Figure 39: (**a**) Mean value of friction torque versus time for each test on "Robot 1". Experimental data, Joint 3, and velocity at 60%. (**b**) Fitting of the data using (158).

5.2 RESULTS

As first results, Figure 8 (section 2.4) reports the different friction torque for different velocities in cold and hot conditions for the six joints. According to our experiments, the estimated temperature obtained by the thermal model could change from about 15 °C (cold state) to about 50 or 60 °C (depending on the

joint) in hot state. Moreover, these values were confirmed by the data collected using temperature sensors located near the gears.

As stated before, the described analysis was applied to two manipulators of the same model named "Robot 1" and "Robot 2". The results of the friction estimation on "Robot 1" for all the six joints at 60% of the maximum velocity are presented in Figure 40. The identification was performed for each test repetition as well as on the merge of the complete set of the collected data (mixed data). It is worth noting that the behavior of the robot is quite repetitive for all the joints. Each of them has regular heating which leads to a steady condition in about 1–2 h depending on the joint.

The time constant analysis of the six joints (see for example Table 8 in section 4.3.1) shows that one of them is shorter than the other one. While t_{w1} is in the range of 20–30 min, t_{w2} is in the range of 2–3 min, depending on the joint number. Moreover, the different time constants estimation, coming from all the tests, is mainly located at a narrow range for each joint. The Mix data are the results of curve fitting by merging the data of all tests. Based on the results of Figure 40, it is proved that the accuracy of the friction estimation is improved by the mixed data results. The estimation of the temperature during the working cycle permits predicting the correct friction contribution. As an example Figure 41 shows the difference in torque obtained by moving the robot with the same trajectory, but at different thermal conditions. It is worth noting how taking the temperature into account permits to have an excellent prediction of the torque both in hot and cold conditions.

The same procedure was implemented on "Robot 2". The results of the estimation on the friction-temperature characteristic are completely comparable to those obtained for "Robot 1". The curves exhibit similar trends and the ranges of the torque value are also equivalent. A further comparison between the two robots was performed on the estimation analysis of the inertial parameters. Results for some dynamic parameters are reported in Figure 42. The estimated values of these parameters for the two robots are reported for each of the 24 cycles. It is worth noting that they reach a stable value after a few cycles. The four repetitions of the test provide quite similar data. Thence, it is confirmed that the identification procedure is reasonably robust and suggests that the inertia parameters estimation can be performed after a short warming-up period of about 30–50 min (depending on the robot model).

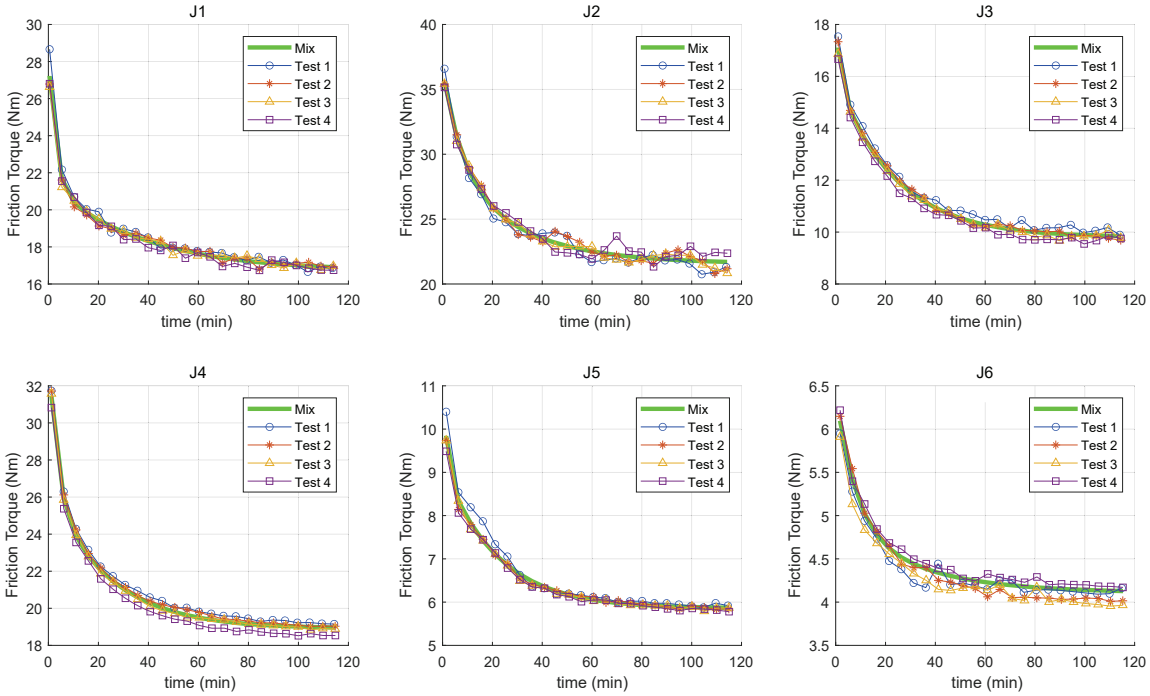


Figure 40: Friction torque versus time for all joints in Tests 1–4 performed on "Robot 1", with the velocity at 60%. The Mix curve is the results of the curve fitting by merging the data of each test.

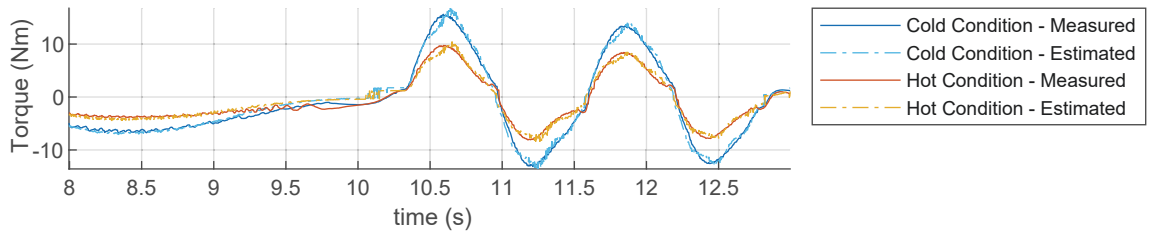


Figure 41: Example of measured and predicted torque (dynamics plus friction) in cold and hot conditions on the same trajectory with variable velocity (Joint 5, Test 1).

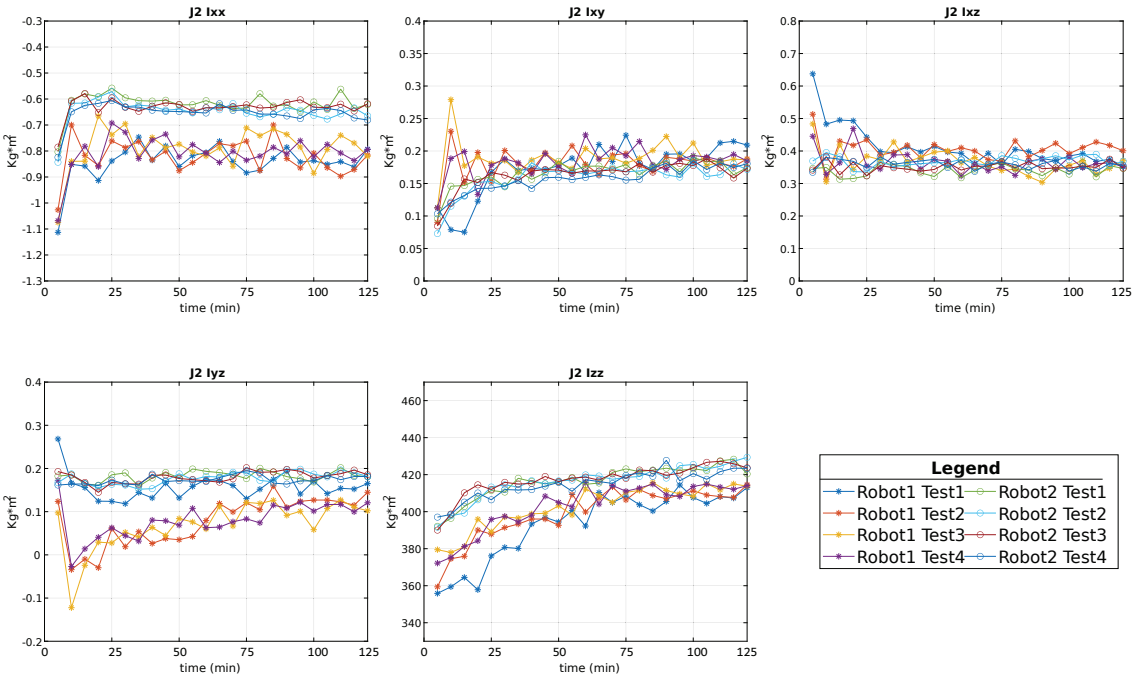


Figure 42: Evolution of the identified values Ixx, Ixy, Ixz, Iyz, **and** Izz of Joint 2 for all the tests on "Robot 1" and "Robot 2". The values of the parameters are repeatable and quite similar between the two robots.

The results obtained from these analyses show the repetitiveness of the experimental procedure used in this work. The parameters identified by the model are comparable between the tests carried out on different days and also comparable with those obtained on different robots but of the same model. It is of utmost importance to obtain a mathematical model describing the joints torques variations taking into account the temperature alterations that may occur during the working operation. This could be useful for multiple applications, such as predictive maintenance, advanced control strategies, the realization of virtual force sensors, and human–robot interaction. Figure 43 shows a clear example of the model implementation for predictive maintenance. During one of the tests carried out on "Robot 2", a mechanical problem was found in the first joint. After about 60 min on the second day of the tests, the friction torque had an unexpected increase and then it settled at a higher value compared with the previous tests. The measurement carried out the day after shows that the same movement required more energy from the very start. A possible explanation for this phenomenon may be a mechanical problem inside the transmission. In this specific case, to restore the correct operation of the machine, it was sufficient to execute the inspection of the joint following the mechanical maintenance manual provided by the manufacturer. Further analysis of this issue is shown at the bottom of Figure 42. It is possible to note how the unexpected increase in friction torque results in a sudden change in the friction parameters of the dynamic model. The phenomenon

just described suggests how the model provided in this work may be used for the preventive maintenance of industrial manipulators. By carrying out the identification procedure on a new robot, it is possible to obtain an initial condition state of the machine. Repeating this procedure over time, it is possible to compare the new results with the initial ones to check if the components have been damaged, helping the user to perform the necessary maintenance. It is worth pointing out that the unexpected increase is about the 32% of the expected value, while the change due to thermal effect is 60%/70%, thus the damage cannot be detected by using a model that neglects the thermal effect.

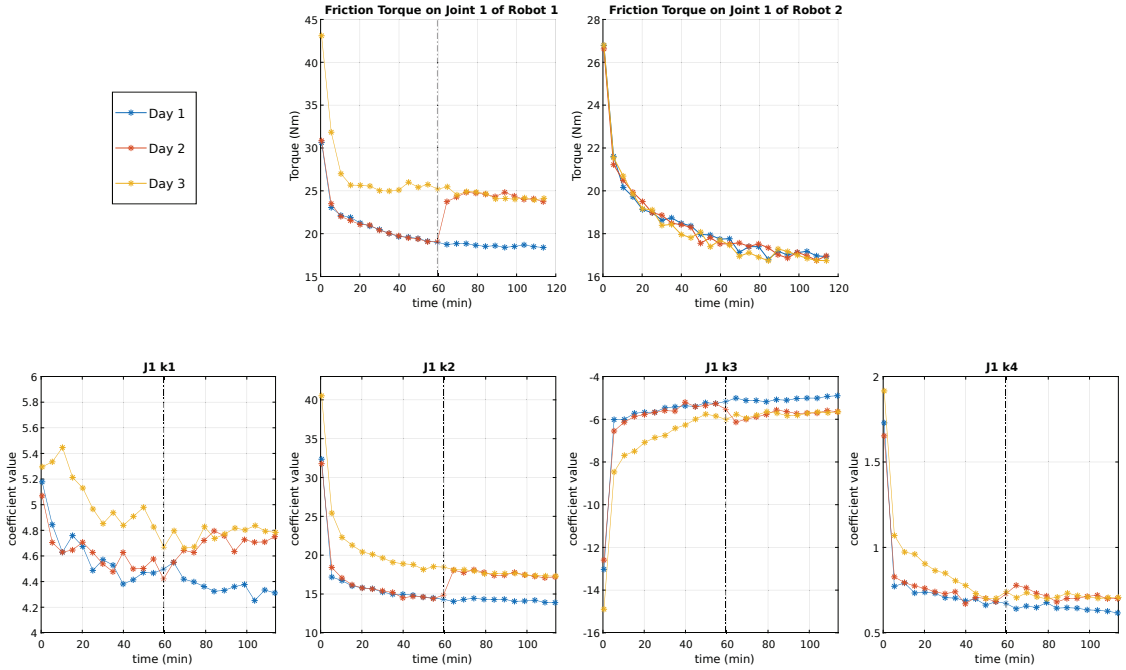


Figure 43: (**Top**) The friction torque versus time for "Robot 1" and "Robot 2". During one of the tests on "Robot 2", a mechanical problem occurred. It is possible to see the unexpected increase in torque on the left-side graph. The graph on the opposite side shows the ordinary behavior of "Robot 1" performing the same tests. (**Bottom**) The evolution of the friction parameters (Equation (4)) during the tests performed on "Robot 2". It is evident how the mechanical problem results in a change in the model values.

5.3 DISCUSSION

The experimental campaign lasted several days using two robots of the same kind. The experimental data show the repeatability of the thermal phenomenon and the robust identification of the parameters of the dynamics and thermal models.

This repeatability allows detecting unforeseen changes of the friction torque due to minor mechanical damages, as shown in Figure 43. It is worth stressing

that the friction changes from cold to warm conditions are more significant than the changes due to the damages. Thus, one of the main advantages of integrating a thermal model is to provide a higher discrimination level than models that neglect temperature effects, allowing early detections of torque changes due to aging and damages.

Another advantage of the algorithm is the ability to predict the required torque, improving the performance of control algorithms based on precomputed torque.

The main drawback is the requirement of an extended test campaign on a representative batch of each robot type.

The study of the robot dynamics and, in particular, the friction torques is crucial to understand the robot behavior, as mentioned in the Introduction. This information is helpful to analyze the aging and appearance of possible defects during the robot activities. As is well known, the design of a suitable exciting trajectory is essential to estimate the robot parameters correctly.

This works shows the effectiveness of the dynamics model, coupled with the temperature-based friction model, to match experimental data in multiple robots. The experimental campaign, performed on several days, showed consistent results. The experiments shows that the estimated parameters are biased during the first part of the thermal transient due to friction changes. Thus, a model that neglects temperature effects can work properly if and only if the model is identified and used at the same thermal steady-state. Otherwise, a temperature estimator is required to compensate for its effect.

Therefore, this work could help people understand the robot behavior changes during working cycles and develop a dynamic model of the robot, including the thermal effect.

The model can effectively estimate the friction variation due to temperature. Since this variation is up to 70%, it is crucial to consider it in the torque estimation to discriminate it from other changes due to possible damages or aging. It is worth pointing out that this discrimination ability could be relevant in advanced maintenance and performance monitor.

CONCLUSIONS

This thesis discusses problems related to robot dynamics modeling and dynamics model calibration. It also aims to partially fill the gap between the robotic research community and industrial players. Particular emphasis has been given to friction modeling considering thermal effects. The repeatability of the results has been validated on two standard Efort Industrial Robots of the same model. As a result, the differences between dynamics and friction behaviors occurring between the two have been compared and analyzed. This analysis may be useful for future improvements in both manufacturing and maintenance of the machines.

Chapters 1 and 2 deal with the rigid body dynamics model derivation, reduction, and calibration. Chapter 1 describes the generation of the regression matrix and a particular method based on QR factorization and SVD decomposition to reduce the model parameters. In Chapter 2 the concept of dynamics model calibration has been described and the generation of the excitation trajectories has been explained in detail. To better identify friction and inertial parameters, the trajectories have been divided into three stages. The first is based on the finite Fourier series and it is optimized for inertial parameter estimation. The second is specially designed for friction estimation, and finally the third is a warm-up trajectory designed to quickly increase the temperature of the robot over time. The proposed trajectories are then used to populate the dynamics regression matrix to estimate the dynamics parameters set.

Chapter 3 shows the robots used in the experiments, including their main characteristics and the proprietary software environment. Due to the limitations of such software, it was necessary to develop a ROS-Industrial Driver (detailed in Appendix A) to allow communication between ROS and the system, thus providing complete automation of the test procedure and data collection.

In Chapter 4 the friction dependence on temperature has been investigated. It has been shown that the variation of friction can be modeled and predicted by knowing the mechanical characteristics of the robot and its work cycle. The main goal of this study in the context of the overall thesis work was to research and develop new friction models that could explain the behavior of robotics joints where standards models fail.

The last Chapter (5) shows the effectiveness of the dynamics model coupled with the temperature-based friction model on the two test robots. Repetitive tests

performed on different days showed that the inertial and friction parameters can be robustly estimated and that the value of the measured joint friction can be used to estimate the unexpected conditions of the joints.

This thesis aims to be an important help in understanding the robot behavior variations during working cycles. It could also be a useful stepping stone to develop a dynamic model of the robot that includes thermal effects. The proposed models can effectively estimate the friction variation due to temperature. Since this variation is up to 70% of the total torque, it is crucial to consider it in the torque estimation to discriminate it from other changes due to possible damages or component aging and wearing. It is worth pointing out that this distinction could be relevant in many applications including sensorless identification of collisions, predictive maintenance programs, and human-robot interaction.

BIBLIOGRAPHY

[1] Måns Östring. *Identification, diagnosis, and control of a flexible robot arm*. Division of Automatic Control, Department of Electrical Engineering . . . , 2002.

[2] F Caccavale and P Chiacchio. Identification of dynamic parameters and feedforward control for a conventional industrial manipulator. *Control Engineering Practice*, 2(6):1039–1050, 1994.

[3] I Iglesias, MA Sebastián, and JE Ares. Overview of the state of robotic machining: Current situation and future potential. *Procedia engineering*, 132: 911–917, 2015.

[4] Ulrich Schneider, Mahdi Momeni-K, Matteo Ansaloni, and Alexander Verl. Stiffness modeling of industrial robots for deformation compensation in machining. In *2014 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 4464–4469. IEEE, 2014.

[5] G Duelen and K Schröer. Robot calibration—method and results. *Robotics and Computer-Integrated Manufacturing*, 8(4):223–231, 1991.

[6] Jean-Michel Renders, Eric Rossignol, Marc Becquet, Raymond Hanus, et al. Kinematic calibration and geometrical parameter identification for robots. *IEEE Transactions on robotics and automation*, 7(6):721–732, 1991.

[7] John M Hollerbach and Charles W Wampler. The calibration index and taxonomy for robot kinematic calibration methods. *The international journal of robotics research*, 15(6):573–591, 1996.

[8] Chandra Sekhar Gatla, Ron Lumia, John Wood, and Greg Starr. An automated method to calibrate industrial robots using a virtual closed kinematic chain. *IEEE Transactions on robotics*, 23(6):1105–1116, 2007.

[9] In-Chul Ha. Kinematic parameter calibration method for industrial robot manipulator using the relative position. *Journal of mechanical science and technology*, 22(6):1084, 2008.

[10] Pasquale Chiacchio, Lorenzo Sciavicco, and Bruno Siciliano. The potential of model-based control algorithms for improving industrial robot tracking performance. In *Proceedings of the IEEE International Workshop on Intelligent Motion Control*, volume 2, pages 831–836. IEEE, 1990.

[11] Stig Moberg. *Modeling and control of flexible manipulators*. PhD thesis, Linköping University Electronic Press, 2010.

[12] Jun Wu, Jinsong Wang, and Zheng You. An overview of dynamic parameter identification of robots. *Robotics and computer-integrated manufacturing*, 26(5): 414–419, 2010.

[13] Bruno Siciliano, Lorenzo Sciavicco, Luigi Villani, and Giuseppe Oriolo. *Robotics: modelling, planning and control*. Springer Science & Business Media, 2010.

[14] Morten Lind, Johannes Schrimpf, and Thomas Ulleberg. Open real-time robot controller framework. In *Proc. CIRP Conf. Assembly Technology and Systems-Responsive, customer demand driven, adaptive assembly*, pages 13–18, 2010.

[15] Vicente Mata, Francesc Benimeli, Nidal Farhat, and Angel Valera. Dynamic parameter identification in industrial robots considering physical feasibility. *Advanced Robotics*, 19(1):101–119, 2005.

[16] J. M. Hollerbach. A recursive lagrangian formulation of manipulator dynamics and a comparative study of dynamics formulation complexity. *IEEE Transactions on Systems, Man, and Cybernetics*, 10(11):730–736, 1980. doi: 10.1109/TSMC.1980.4308393.

[17] John YS Luh, Michael W Walker, and Richard PC Paul. On-line computational scheme for mechanical manipulators. *Journal of Dynamic Systems, Measurement, and Control*, 102(2):69–76, Jun 1980. ISSN 0022-0434. doi: 10.1115/1.3149599. URL https://doi.org/10.1115/1.3149599.

[18] William M. Silver. On the equivalence of lagrangian and newton-euler dynamics for manipulators. *The International Journal of Robotics Research*, 1 (2):60–70, 1982. doi: 10.1177/027836498200100204. URL https://doi.org/10.1177/027836498200100204.

[19] John J Craig. *Introduction to robotics: mechanics and control, 3/E*. Pearson Education India, 2009.

[20] Wikipedia contributors. Parallel axis theorem — Wikipedia, the free encyclopedia. https://en.wikipedia.org/w/index.php?title=Parallel_axis_theorem&oldid=1010787717, 2021. [Online; accessed 10-March-2021].

[21] Giovanni Legnani, Luca Simoni, Manuel Beschi, and Antonio Visioli. A new friction model for mechanical transmissions considering joint temperature. In *International Design Engineering Technical Conferences and Computers and Information in Engineering Conference*, volume 50183, page V006T09A020. American Society of Mechanical Engineers, 2016.

[22] Christopher G Atkeson, Chae H An, and John M Hollerbach. Estimation of inertial parameters of manipulator loads and links. *The International Journal of Robotics Research*, 5(3):101–119, 1986.

[23] Maxime Gautier and Wisama Khalil. On the identification of the inertial parameters of robots. In *Proceedings of the 27th IEEE Conference on Decision and Control*, volume 3, pages 2264–2269. IEEE Austin, 1988.

[24] Maxime Gautier and Wisama Khalil. Direct calculation of minimum set of inertial parameters of serial robots. *IEEE Transactions on robotics and Automation*, 6(3):368–373, 1990.

[25] Gianluca Antonelli, Fabrizio Caccavale, and Pasquale Chiacchio. A systematic procedure for the identification of dynamic parameters of robot manipulators. *Robotica*, 17(4):427–435, 1999.

[26] Javier Ros, Xabier Iriarte, and Vicente Mata. 3d inertia transfer concept and symbolic determination of the base inertial parameters. *Mechanism and machine theory*, 49:284–297, 2012.

[27] CM Pham and Maxime Gautier. Essential parameters of robots. In *[1991] Proceedings of the 30th IEEE Conference on Decision and Control*, pages 2769–2774. IEEE, 1991.

[28] X Iriarte, J Ros, V Mata, and J Aginaga. Determination of the symbolic base inertial parameters of planar mechanisms. *European Journal of Mechanics-A/Solids*, 61:82–91, 2017.

[29] Maxime Gautier. Numerical calculation of the base inertial parameters of robots. *Journal of robotic systems*, 8(4):485–506, 1991.

[30] Nikolaos A Bompos, Panagiotis K Artemiadis, Apollon S Oikonomopoulos, and Kostas J Kyriakopoulos. Modelling, full identification and control of the mitsubishi pa-10 robot arm. pages 1–6, 2007.

[31] Wisama Khalil and Etienne Dombre. *Modeling, identification and control of robots*. Butterworth-Heinemann, 2004.

[32] Christopher G. Atkeson, Chae H. An, and John M. Hollerbach. Estimation of inertial parameters of manipulator loads and links. *The International Journal of Robotics Research*, 5(3):101–119, 1986. doi: 10.1177/027836498600500306. URL https://doi.org/10.1177/027836498600500306.

[33] Wikipedia contributors. Qr decomposition — Wikipedia, the free encyclopedia. https://en.wikipedia.org/w/index.php?title=QR_decomposition&oldid=1023213167, 2021. [Online; accessed 20-May-2021].

[34] Wikipedia contributors. Singular value decomposition — Wikipedia, the free encyclopedia. https://en.wikipedia.org/w/index.php?title=Singular_value_decomposition&oldid=1022188613, 2021. [Online; accessed 20-May-2021].

[35] Brian Armstrong. On finding exciting trajectories for identification experiments involving systems with nonlinear dynamics. *The International journal of robotics research*, 8(6):28–48, 1989.

[36] C Presse and Maxime Gautier. New criteria of exciting trajectories for robot identification. In *[1993] Proceedings IEEE International Conference on Robotics and Automation*, pages 907–912. IEEE, 1993.

[37] Jan Swevers, Chris Ganseman, D Bilgin Tukel, Joris De Schutter, and Hendrik Van Brussel. Optimal robot excitation and identification. *IEEE transactions on robotics and automation*, 13(5):730–740, 1997.

[38] Kyung-Jo Park. Fourier-based optimal excitation trajectories for the dynamic identification of robots. *Robotica*, 24(5):625–633, 2006.

[39] Wolfgang Rackl, Roberto Lampariello, and Gerd Hirzinger. Robot excitation trajectories for dynamic parameter estimation using optimized b-splines. In *2012 IEEE international conference on robotics and automation*, pages 2042–2047. IEEE, 2012.

[40] K Otani. Motion planning and modeling for accurately identifying dynamic parameters of an industrial robotic manipulator. *24th ISIR, 1993*, 1993.

[41] Miguel Diaz-Rodriguez, Xabier Iriarte, Vicente Mata, and Javier Ros. On the experiment design for direct dynamic parameter identification of parallel robots. *Advanced Robotics*, 23(3):329–348, 2009.

[42] Xia Hong, Chris J. Harris, Sheng Chen, and Paul M. Sharkey. Robust nonlinear model identification methods using forward regression. *IEEE Transactions on Systems, Man, and Cybernetics-Part A: Systems and Humans*, 33(4): 514–523, 2003.

[43] G Calafiore, Marina Indri, and Basilio Bona. Robot dynamic calibration: Optimal excitation trajectories and experimental parameter estimation. *Journal of robotic systems*, 18(2):55–68, 2001.

[44] Ngoc Dung Vuong and Marcelo H Ang Jr. Dynamic model identification for industrial robots. *Acta Polytechnica Hungarica*, 6(5):51–68, 2009.

[45] Maxime Gautier. Dynamic identification of robots with power model. In *Proceedings of international conference on robotics and automation*, volume 3, pages 1922–1927. IEEE, 1997.

[46] Philippe Poignet and Maxime Gautier. Comparison of weighted least squares and extended kalman filtering methods for dynamic identification of robots. In *Proceedings 2000 ICRA. Millennium Conference. IEEE International Conference on Robotics and Automation. Symposia Proceedings (Cat. No. 00CH37065)*, volume 4, pages 3622–3627. IEEE, 2000.

[47] Alexandre Janot, Pierre-Olivier Vandanjon, and Maxime Gautier. A generic instrumental variable approach for industrial robot identification. *IEEE Transactions on Control Systems Technology*, 22(1):132–145, 2013.

[48] Francesc Benimeli, Vicente Mata, and Francisco Valero. A comparison between direct and indirect dynamic parameter identification methods in industrial robots. *Robotica*, 24(5):579–590, 2006.

[49] Ljung Lennart. System identification: theory for the user. *PTR Prentice Hall, Upper Saddle River, NJ*, 28, 1999.

[50] J. Swevers, C. Ganseman, J. De Schutter, and H. Van Brussel. Experimental robot identification using optimised periodic trajectories. *Mechanical Systems and Signal Processing*, 10(5):561–577, 1996. ISSN 0888-3270. doi: https://doi.org/10.1006/mssp.1996.0039. URL https://www.sciencedirect.com/science/article/pii/S0888327096900394.

[51] Jingfu Jin and Nicholas Gans. Parameter identification for industrial robots with a fast and robust trajectory design approach. *Robotics and Computer-Integrated Manufacturing*, 31:21–29, 2015. ISSN 0736-5845. doi: https://doi.org/10.1016/j.rcim.2014.06.004. URL https://www.sciencedirect.com/science/article/pii/S0736584514000441.

[52] J. Swevers, W. Verdonck, and J. De Schutter. Dynamic model identification for industrial robots. *IEEE Control Systems Magazine*, 27(5):58–71, 2007. doi: 10.1109/MCS.2007.904659.

[53] J. Swevers, C. Ganseman, J. De Schutter, and H. Van Brussel. Generation of periodic trajectories for optimal robot excitation. *Journal of Manufacturing Science and Engineering*, 119(4A):611–615, Nov 1997. ISSN 1087-1357. doi: 10.1115/1.2831194. URL https://doi.org/10.1115/1.2831194.

[54] D. Kostic, Bram de Jager, M. Steinbuch, and R. Hensen. Modeling and identification for high-performance robot control: an rrr-robotic arm case study. *IEEE Transactions on Control Systems Technology*, 12(6):904–919, 2004. doi: 10.1109/TCST.2004.833641.

[55] J. Swevers, C. Ganseman, D. B. Tukel, J. de Schutter, and H. Van Brussel. Optimal robot excitation and identification. *IEEE Transactions on Robotics and Automation*, 13(5):730–740, 1997. doi: 10.1109/70.631234.

[56] N.D. Vuong and M.H. Ang Jr. Dynamic model identification for industrial robots. 2009.

[57] G. Antonelli, F. Caccavale, and P. Chiacchio. A systematic procedure for the identification of dynamic parameters of robot manipulators. *Robotica*, 17(4):427–435, 1999. doi: 10.1017/S026357479900140X.

[58] A. Calanca, L. M. Capisani, A. Ferrara, and L. Magnani. Mimo closed loop identification of an industrial robot. *IEEE Transactions on Control Systems Technology*, 19(5):1214–1224, 2011. doi: 10.1109/TCST.2010.2077294.

[59] The MathWorks, Inc. Find minimum of function using genetic algorithm. https://www.mathworks.com/help/gads/ga.html, 2021. [Online; accessed 2021-FEB-27].

[60] The MathWorks, Inc. Find minimum of constrained nonlinear multivariable function. https://www.mathworks.com/help/optim/ug/fmincon.html, 2021. [Online; accessed 2021-FEB-27].

[61] G. Legnani and I. Fassi. *Robotica industriale*. Città Studi, 2019. ISBN 9788825174281.

[62] Luca Simoni, Manuel Beschi, Giovanni Legnani, and Antonio Visioli. On the inclusion of temperature in the friction model of industrial robots. *IFAC-PapersOnLine*, 50(1):3482–3487, 2017.

[63] Roberto Pagani, Giovanni Legnani, Giovanni Incerti, and Matteo Gheza. Evaluation and modeling of the friction in robotic joints considering thermal effects. *Journal of Mechanisms and Robotics*, 12(2), 2020.

[64] Efort intelligent Equipment Co.,Ltd Official Page. URL http://www.efort.com.cn/en/. Accessed Aug-2021.

[65] R Stribeck. Kugellager für beliebige belastungen (ball bearings for arbitrary loads). *Mitteilungen aus der Centralstelle für wissenschaftlichtechnische Untersuchungen, HS Hermann, Berlin*, 1900.

[66] R. Stribeck. Die wesentlichen eigenschaften der gleit- und rollenlager (characteristics of plain and roller bearings. *Zeit. des VDI 46*, 1902.

[67] Jun Young Yoon and David L Trumper. Friction modeling, identification, and compensation based on friction hysteresis and dahl resonance. *Mechatronics*, 24(6):734–741, 2014.

[68] M. Ruderman. Presliding hysteresis damping of lugre and maxwell-slip friction models. *Mechatronics*, 30:225–230, 2015.

[69] L. Freidovich, A. Robertsson, A. Shiriaev, and R. Johansson. Lugre-model-based friction compensation. *IEEE Transactions on Control Systems Technology*, 18(1):194–200, 2010.

[70] K. J. Åström, C. Canudas de Wit, H. Olsson, and P. Lischinsky. A new model for control of systems with friction. *IEEE Transaction on Automatic Control*, 40(3):419–425, 1995.

[71] K. J. Åström and C. Canudas de Wit. Revisiting the lugre model. *IEEE Control Systems Magazine*, 28(6):101–114, 2008.

[72] V. Lampaert, F. Al-Bender, and J. Swevers. A generalized maxwell-slip friction model appropriate for control purposes. In *Proceedings of IEEE International Conference on Physics and Control*, volume 4, pages 1170–1177, St. Petersburg (Russia), 2003.

[73] Luca Simoni, Manuel Beschi, Giovanni Legnani, and Antonio Visioli. Friction modeling with temperature effects for industrial robot manipulators. In *2015 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), Hamburg, Germany, 28 Sept.-2 Oct., 2015*, pages 3524–3529. IEEE, 2015.

[74] Fredrik Bagge Carlson, Anders Robertsson, and Rolf Johansson. Modeling and identification of position and temperature dependent friction phenomena without temperature sensing. In *2015 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), Hamburg, Germany, 28 Sept.-2 Oct., 2015*, pages 3045–3051. IEEE, 2015.

[75] André Carvalho Bittencourt and Patrik Axelsson. Modeling and experiment design for identification of wear in a robot joint under load and temperature uncertainties based on friction data. *IEEE/ASME transactions on mechatronics*, 19(5):1694–1706, 2013.

[76] Luca Simoni, Manuel Beschi, Giovanni Legnani, and Antonio Visioli. Modelling the temperature in joint friction of industrial manipulators. *Robotica*, 37(5):906–927, 2019.

[77] Giovanni Legnani, Giovanni Incerti, Roberto Pagani, and Matteo Gheza. Modelling and evaluation of the friction in robotic joints considering thermal effects. In *International Design Engineering Technical Conferences and Computers and Information in Engineering Conference*, volume 59230, page V05AT07A062. American Society of Mechanical Engineers, 2019.

[78] Roberto Pagani, Fabrizio Padula, Giovanni Legnani, Ryan Loxton, and Antonio Visioli. A fractional model of the friction-temperature behavior in robot joints. In *2019 7th International Conference on Control, Mechatronics and Automation (ICCMA)*, pages 157–161. IEEE, 2019.

[79] Roberto Pagani, Giovanni Legnani, Giovanni Incerti, Manuel Beschi, and Monica Tiboni. The influence of heat exchanges on friction in robotic joints: Theoretical modelling, identification and experiments. In *International Design Engineering Technical Conferences and Computers and Information in Engineering Conference*, volume 83990, page V010T10A053. American Society of Mechanical Engineers, 2020.

[80] Francesco Jatta, Giacomo Legnani, and Antonio Visioli. Friction compensation in hybrid force/velocity control of industrial manipulators. *IEEE Transactions on Industrial Electronics*, 53(2):604–613, 2006.

[81] Emanuele Lubrano and Reymond Clavel. Thermal calibration of a 3 dof ultra high-precision robot operating in industrial environment. In *2010 IEEE international conference on robotics and automation, Anchorage, AK, USA, May 3-7, 2010*, pages 3692–3697, Anchorage, AK, USA, 2010. IEEE.

[82] Chunhe Gong, Jingxia Yuan, and Jun Ni. Nongeometric error identification and compensation for robotic system by inverse calibration. *International Journal of Machine Tools and Manufacture*, 40(14):2119–2137, 2000.

[83] Héctor Manuel Moya-Cessa and Francisco Soto-Eguibar. *Differential equations: an operational approach*. Rinton Press, Incorporated, 2011.

[84] André Carvalho Bittencourt, Erik Wernholt, Shiva Sander-Tavallaey, and Torgny Brogårdh. An extended friction model to capture load and temperature effects in robot joints. In *2010 IEEE/RSJ International Conference on Intelligent Robots and Systems, Taipei, Taiwan, October 18-22, 2010*, pages 6161–6167. IEEE, 2010.

[85] S Ghidini, M Beschi, and N Pedrocchi. A robust linear control strategy to enhance damping of a series elastic actuator on a collaborative robot. *Journal of Intelligent & Robotic Systems*, pages 1–15, 2019.

[86] E Villagrossi, L Simoni, M Beschi, N Pedrocchi, A Marini, L Molinari Tosatti, and A Visioli. A virtual force sensor for interaction tasks with conventional industrial robots. *Mechatronics*, 50:78–86, 2018.

[87] A. Tepljakov. *Fractional-order Modeling and Control of Dynamic Systems*. Springer, 2017.

[88] C. M. Ionescu. *The Human Respiratory System: An Analysis of the Interplay between Anatomy, Structure, Breathing and Fractal Dynamics*. Springer, Princeton, NJ, 2013.

[89] Y. Q. Chen. Ubiquitous fractional order controls? In *Proceedings 2nd IFAC Workshop of Fractional Differentiation and its Applications*, pages 481–492, Porto (P), 2006.

[90] A. Oustaloup, P. Lanusse, J. Sabatier, and P. Melchior. CRONE control: Principles, extensions and applications. *Journal of Applied Nonlinear Dynamics*, 2 (3):207–223, 2013.

[91] I. Tejado, S. H. HosseinNia, and B. M. Vinagre. Adaptive gain-order fractional control for network-based applications. *Fractional Calculus and Applied Analysis*, 17(2):62–482, 2014.

[92] A. A Dastjerdi, B. M. Vinagre, Y-Q. Chen, and S. H. HosseinNia. Linear fractional order controllers; a survey in the frequency domain. *Annual Reviews in Control*, 2019.

[93] F. Padula and A. Visioli. Inversion-based feedforward and reference signal design for fractional constrained control systems. *Automatica*, 50(8):2169–2178, 2014.

[94] J. Sabatier, M. Moze, and C. Farges. LMI stability conditions for fractional order systems. *Computer and Mathematics with Applications*, 59(5):1594–1609, 2010.

[95] F. Padula and A. Visioli. *Advances in Robust Fractional Control*. Springer, 2014.

[96] I. Podlubny. *Fractional differential equations*. Academic Press, 1999.

[97] L. R. Evangelista and E. Kaminski Lenzi. *Fractional Diffusion Equations and Anomalous Diffusion*. Cambridge University Press, 2018.

[98] J. D. Gabano and T. Poinot. Fractional modeling applied to non destructive thermal characterization. In *Proceedings 18th IFAC World Congress*, pages 13972–13977, Milan (I), 2011.

[99] R. Malti, J. Sabatier, and H. Akcay. Thermal modeling and identification of an aluminum rod using fractional calculus. In *Proceedings of the IFAC Symposium on Systems Identification*, pages 958–963, Saint-Malo (France), 2009.

[100] C. A. Monje, Y-Q. Chen, B. M. Vinagre, D. Xue, and V. Feliu. *Fractional-order systems and controls*. Springer, 2010.

[101] J. Sabatier, P. Lanusse, P. Melchior, and A. Oustaloup. *Fractional Order Differentiation and Robust Control Design*. Springer, 2015.

[102] A. Carvalho Bittencourt, E. Wernholt, S. Sander-Tavallaey, and T. Brogardh. An extended friction model to capture load and temperature effects in robot joints. In *Proceedings IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 6161–6167, Taipei (Taiwan), 2010.

[103] André Carvalho Bittencourt and Svante Gunnarsson. Static friction in a robot joint - modeling and identification of load and temperature effects. *Journal of Dynamic Systems, Measurement, and Control*, 134(5), Jul 2012. ISSN 0022-0434. doi: 10.1115/1.4006589. URL https://doi.org/10.1115/1.4006589. 051013.

[104] Jeong-Jong Lee, Young-Kyoun Kim, Hyuk Nam, Kyung-Ho Ha, Jung-Pyo Hong, and Don-Ha Hwang. Loss distribution of three-phase induction motor fed by pulsewidth-modulated inverter. *IEEE Transactions on Magnetics*, 40(2):762–765, 2004.

[105] A. Montazeri, C. West, S. D. Monk, and C. J. Taylor. Dynamic modelling and parameter estimation of a hydraulic robot manipulator using a multi-objective genetic algorithm. *International Journal of Control*, 90(4):661–683, 2017.

[106] Manuel Beschi, Enrico Villagrossi, Nicola Pedrocchi, and Lorenzo Molinari Tosatti. A general analytical procedure for robot dynamic model reduction. In *2015 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 4127–4132. IEEE, 2015.

[107] P. K. Khosla and T. Kanade. Real-time implementation and evaluation of the computed-torque scheme. *IEEE Transactions on Robotics and Automation*, 5(2):245–253, 1989. doi: 10.1109/70.88047.

[108] Giovanni Carabin and Lorenzo Scalera. On the trajectory planning for energy efficiency in industrial robotic systems. *Robotics*, 9(4):89, 2020.

[109] Simone Pasinetti, Cristina Nuzzi, Matteo Lancini, Giovanna Sansoni, Franco Docchio, and Alberto Fornaser. Development and characterization of a safety system for robotic cells based on multiple time of flight (tof) cameras and point cloud analysis. In *2018 Workshop on Metrology for Industry 4.0 and IoT*, pages 1–6. IEEE, 2018.

[110] Eloise Matheson, Riccardo Minto, Emanuele GG Zampieri, Maurizio Faccio, and Giulio Rosati. Human–robot collaboration in manufacturing applications: A review. *Robotics*, 8(4):100, 2019.

[111] James Cannan and Huosheng Hu. Human-machine interaction (hmi): A survey. *University of Essex*, 2011.

[112] Maxime Gautier and Wisama Khalil. Direct calculation of minimum set of inertial parameters of serial robots. *IEEE Transactions on robotics and Automation*, 6(3):368–373, 1990.

[113] Maxime Gautier and Wisama Khalil. Identification of the minimum inertial parameters of robots. pages 1529–1530, 1989.

[114] Enrico Villagrossi, Giovanni Legnani, Nicola Pedrocchi, Federico Vicentini, Lorenzo Molinari Tosatti, Fabio Abbà, and Aldo Bottero. Robot dynamic model identification through excitation trajectories minimizing the correlation influence among essential parameters. In *2014 11th International Conference on Informatics in Control, Automation and Robotics (ICINCO)*, volume 2, pages 475–482. IEEE, 2014.

[115] Jan Swevers, Chris Ganseman, Joris De Schutter, and Hendrik Van Brussel. Generation of periodic trajectories for optimal robot excitation. 1997.

[116] Lőrinc Márton and Franciscus van der Linden. Temperature dependent friction estimation: Application to lubricant health monitoring. *Mechatronics*, 22(8):1078–1084, 2012.

[117] Sören Andersson, Anders Söderberg, and Stefan Björklund. Friction models for sliding dry, boundary and mixed lubricated contacts. *Tribology international*, 40(4):580–587, 2007.

[118] Phil R Dahl. A solid friction model. Technical report, Aerospace Corp El Segundo Ca, 1968.

[119] Vincent Lampaert, Farid Al-Bender, and Jan Swevers. A generalized maxwell-slip friction model appropriate for control purposes. In *2003 IEEE International Workshop on Workload Characterization (IEEE Cat. No. 03EX775)*, volume 4, pages 1170–1177. IEEE, 2003.

[120] Michael Ruderman. Presliding hysteresis damping of lugre and maxwell-slip friction models. *Mechatronics*, 30:225–230, 2015.

[121] Gianni Ferretti, Gianantonio Magnani, and Paolo Rocco. Single and multistate integral friction models. *IEEE Transactions on Automatic Control*, 49 (12):2292–2297, 2004.

[122] T Piatkowski. Gms friction model approximation. *Mechanism and Machine Theory*, 75:1–11, 2014.

[123] Leonid Freidovich, Anders Robertsson, Anton Shiriaev, and Rolf Johansson. Lugre-model-based friction compensation. *IEEE Transactions on Control Systems Technology*, 18(1):194–200, 2009.

[124] C Canudas De Wit, Hans Olsson, Karl Johan Astrom, and Pablo Lischinsky. A new model for control of systems with friction. *IEEE Transactions on automatic control*, 40(3):419–425, 1995.

[125] Karl Johanastrom and Carlos Canudas-De-Wit. Revisiting the lugre friction model. *IEEE Control systems magazine*, 28(6):101–114, 2008.

[126] Niels Mallon, Nathan van de Wouw, Devi Putra, and Henk Nijmeijer. Friction compensation in a controlled one-link robot using a reduced-order observer. *IEEE Transactions on Control Systems Technology*, 14(2):374–383, 2006.

[127] André Carvalho Bittencourt and Svante Gunnarsson. Static friction in a robot joint—modeling and identification of load and temperature effects. *Journal of Dynamic Systems, Measurement, and Control*, 134(5), 2012.

[128] Don M Pirro, Martin Webster, and Ekkehard Daschner. *Lubrication fundamentals, revised and expanded*. CRC Press, 2016.

[129] Matthias Neubauer, Hubert Gattringer, and Hartmut Bremer. A persistent method for parameter identification of a seven-axes manipulator. *Robotica*, 33(5):1099, 2015.

[130] Bharat Bhushan. Frictional heating and contact temperatures. In *Modern Tribology Handbook, Two Volume Set*, pages 257–294. CRC Press, 2000.

[131] Zhao Zhongyu. *The study of friction variation with temperature in a harmonic drive system: modeling and control*. PhD thesis, Concordia University, 2006.

[132] Lei Hao, Roberto Pagani, Manuel Beschi, and Giovanni Legnani. Dynamic and friction parameters of an industrial robot: Identification, comparison and repetitiveness analysis. *Robotics*, 10(1):49, 2021.

[133] Paulo Flores and Jorge Ambrósio. On the contact detection for contact-impact analysis in multibody systems. *Multibody System Dynamics*, 24(1): 103–122, 2010.

[134] Mohamed Slamani and Ilian A Bonev. Characterization and experimental evaluation of gear transmission errors in an industrial robot. *Industrial Robot: An International Journal*, 2013.

[135] Robox S.p.A. (2017) Official Page. URL http://www.robox.it. Accessed Aug-2021.

A

APPENDIX I

This appendix reports the complete documentation related to the ROS Industrial Robox driver developed during the Ph.D.

A.1 ROS - ROBOT OPERATING SYSTEM

ROS is a meta-operating system for robots, something between an operating system (OS) and a middleware. It provides not only standard operating system services like hardware abstraction, contention management and process management, but also high-level functionalities such as asynchronous and synchronous calls, centralized database, a robot configuration system and so on. ROS was born after the necessity of having a standard programmable environment, common among as much robots as possible. Before this system, every robot designer and robotics researcher would spend considerable amounts of time designing the embedded software within a robot, as well as the hardware itself. The main idea of a robotics OS is to offer standardized functionalities performing hardware abstraction, such as a conventional OS for PC.

A.1.1 *ROS-Industrial*

ROS-Industrial is an open-source project that extends the advanced capabilities of ROS to manufacturing automation and robotics. The ROS-Industrial repository includes interfaces for common industrial manipulators, grippers, sensors, and device networks. It also provides software libraries for automatic sensors calibration, process path/motion planning, applications like Scan-N-Plan (tools that enable real-time robot trajectory planning from 3D scan data), developer tools, and training curriculum that is specific to the needs of manufacturers. ROS-Industrial provides a one-stop location for manufacturing-related ROS software. It possesses software robustness and reliability that meets the needs of industrial applications. It has to be clarified that it does not replace any one technology entirely, rather it combines the relative strengths of ROS and existing technology, combining ROS high-level functionality with the low-level reliability and safety of an industrial robot controller. Others ROS-Industrial's goals are to create standard interfaces to stimulate "hardware-agnostic" software development (using standardized ROS messages), provide an easy path to apply cutting-edge research in industrial applications (using a common ROS architecture) and pro-

vide simple, easy-to-use, well-documented application programming interfaces. The mainly benefits of ROS-Industrial are:

Powerful functionality within ROS: Custom inverse kinematics for manipulators, advanced 2D (image) and 3D (point cloud) perception, rich toolset for development, simulation, and visualization;

New applications: Advanced perception for identifying robot work pieces as opposed to hard tooling, scan-N-Plan algorithms based upon advanced perception and just-in-time planning, Model-based approaches that permit automated programming for thousands of unique CAD parts;

Simplification of robot programming to the task levels: It eliminates path planning and teaching. Collision-free, optimal paths are automatically calculated given tool path way points, applying abstract programming principles to similar tasks;

Costs reduction: Exactly like ROS, ROS-Industrial is an open-source software and it has open-source licenses that allow commercial use without restrictions. It also allows the standardization of robot and sensor interfaces across many industrial platforms.

As previously mentioned, the main goal of the ROS-Industrial program is to provide ROS interfaces to many different kinds of industrial equipment, including PLCs, Robot Controllers, Servos, Human Machine Interfaces, IO Networks, etc. Regarding the branch of robots, the ROS-Industrial distribution contains metapackages for several industrial vendors, such as *ABB, Adept, Fanuc, Kuka, Motoman, Robotiq* and *Universal Robots*. Each of them exposes different functional interfaces, consequently the capability of each can differ. In any case the mainly interfaces developed by the vendors are: *position streaming* (joint positions are streamed to the controller and move velocity is fixed by the controller), *trajectory downloading* (a full trajectory of joint positions is downloaded to the controller, the trajectory includes velocity constraints which are adhered to by the controller), *trajectory streaming* (similar to position streaming, except that trajectory velocity constraints are adhered to by the controller), *torque control* (real-time interface and direct torque commands are sent to the controller), *MoveIt Pkg* (robot specific geometry must be defined in an associated manipulator package, this information is used for path planning purposes and collision checking). Let consider the difference between the term *streamer* and *downloader*. The first one correspond to a method that sends independent joint values to the controller in separate threads, meanwhile the second one correspond to a method that sends an entire trajectory as a sequence of messages to the robot controller. Each of the interfaces listed above is developed on the basis of the cores package of ROS-Industrial packages.

A.2 ROS-INDUSTRIAL PACKAGES

Installing ROS-I packages can be done using package managers or building from the source code. ROS-Industrial core stack contains packages that provide nodes and libraries for communication with industrial robot controllers. It also includes utilities and tools that are useful for industrial robotics and automation applications. In particular this stack includes the following set of ROS packages:

industrial-core: It contains packages and libraries for supporting industrial robotic systems. The stack consists of nodes for communicating with industrial robot controllers, industrial robot simulators, and also provides ROS controllers for industrial robots;

industrial_deprecated: This package contains nodes, launch files, and so on that are going to be deprecated. The files inside this package will delete soon from the repository;

industrial_msgs: It simply contains message definitions, which are specific to the ROS-Industrial packages;

simple_message: This is a part of ROS-Industrial stacks, which is a standard message protocol containing a simple messaging framework for communicating with industrial robot controllers;

industrial_robot_client: This package contains a generic robot client for connecting to industrial robot controllers, which is running an industrial robot server and can communicate using a simple message protocol;

industrial_robot_simulator: It simulates the industrial robot controller, which follows the ROS-Industrial driver standard. Using this simulator, it is possible to simulate and visualize the industrial robot;

industrial_trajectory_filters: This package contains libraries and plugins for filtering the trajectories, which are sent to the robot controller.

The source code of each package can be found on *GitHub* (a development platform from open source to business, usable to host and review code, manage projects, and build software alongside millions of other developers) at the following link: `https://github.com/ros-industrial/industrial_core.git`

Block diagram

From the ROS-I wiki page (`http://wiki.ros.org/Industrial`) it is possible to find a simple block diagram representation of ROS-I packages, which are organized on top of ROS (Figure 44.) A brief description of each of the layers is summarized below.

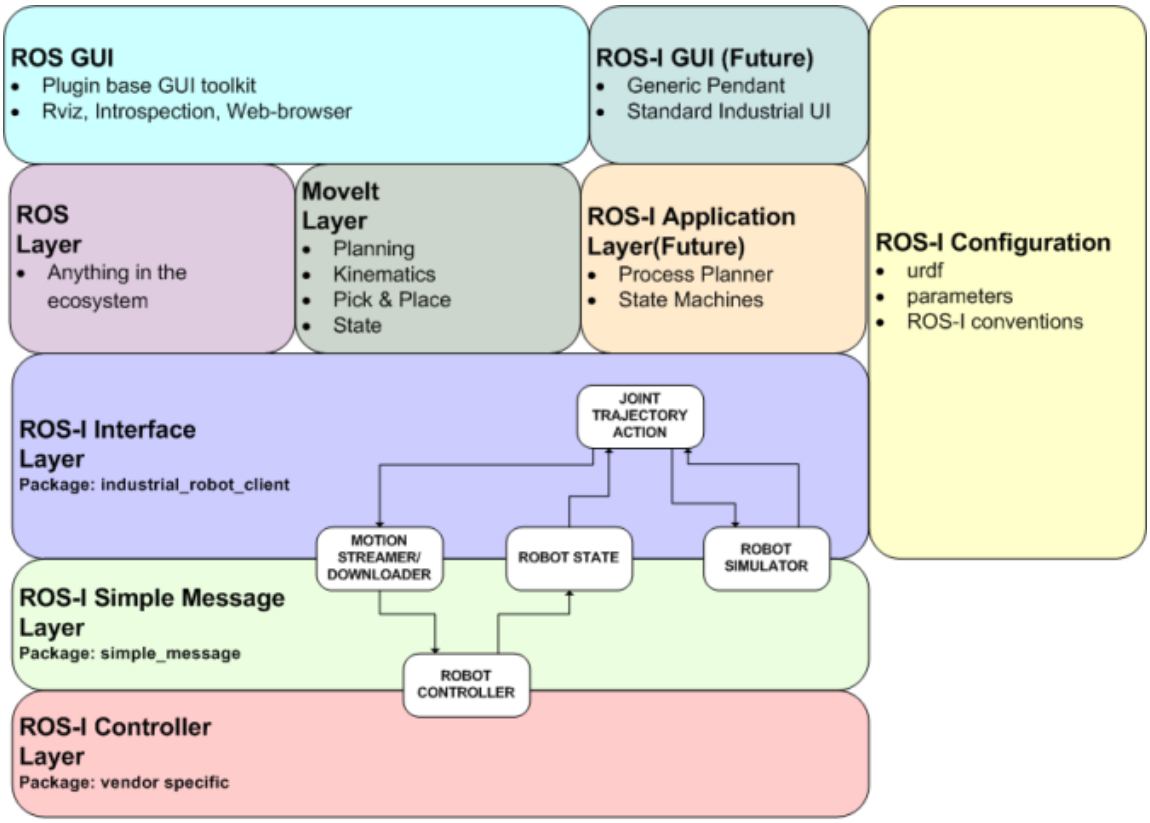


Figure 44: ROS-Industrial block diagram

The ROS GUI: This layer includes the ROS plugin-based GUI (Graphical User Interface) tools layer, which consists of tools such as *RViz*, *rqt_gui*, and so on;

The ROS-I GUI: These GUIs are standard industrial user interface for working with industrial robots which may be implemented in the future;

The ROS Layer: This is the base layer in which all communications are taking place;

The MoveIt! Layer: The *MoveIt!* layer provides a direct solution to industrial manipulators in planning, kinematics, and pick and place;

The ROS-I Application Layer: It consists of an industrial process planner, which is used to plan what is to be manufactured, how it will be manufactured, and what resources are needed for the manufacturing process;

The ROS-I Interface Layer: It consists of the industrial robot client, which can connect to the industrial robot controller using the simple message protocol;

The ROS-I- Simple Message Layer: This is the communication layer to the industrial robot, which is a standard set of protocol that will send data from the industrial robot client to the robot controller and vice versa;

The ROS-I Controller Layer: It has vendor-specific industrial robot controllers.

After discussing the basic concepts, it is possible to start focusing on the most important elements for interfacing an industrial robot to ROS using ROS-Industrial.

Industrial_robot_client package

The industrial robot client nodes are responsible for sending robot position/trajectory data from ROS to the robot controller. The industrial robot client converts the trajectory data to *simple_message* and communicates to the robot controller using the *simple_message* protocol (explained in the next section). The robot controller runs a server, the ROS-I robot client nodes are connecting to this server and start communicating with it.

The *industrial_robot_client* package contains various classes to implement industrial robot client nodes (these classes can be expanded through typical `C++` derived-class functionality to implement robot-specific functionalities that differ from this reference implementation). The main functionalities that a client should have are: update the robot current state from the robot controller and also send joint trajectories/joint position message to the controller. There are two main nodes that are responsible for getting robot state and sending joint trajectory/position values:

The *robot_state* node: It is responsible for publishing the robot's current position, status, and so on;

The *joint_trajectory* node: It subscribes the robot's command topic and sends the joint position commands to the robot controller via the simple message protocol.

State publishing

The *RobotStateInterface* class contains methods to publish the current robot position and status at regular time intervals after receiving the position data from the robot controller. This class is primarily a wrapper around the *MessageManager* class, which listens to the *simple_message* robot connection and processes each incoming message handling using *MessageHandlers*. The *JointRelayHandler* is a *MessageHandler* and its function is to publish the joint position in the *joint_states* topic. When a new message is received, it is processed as follows:

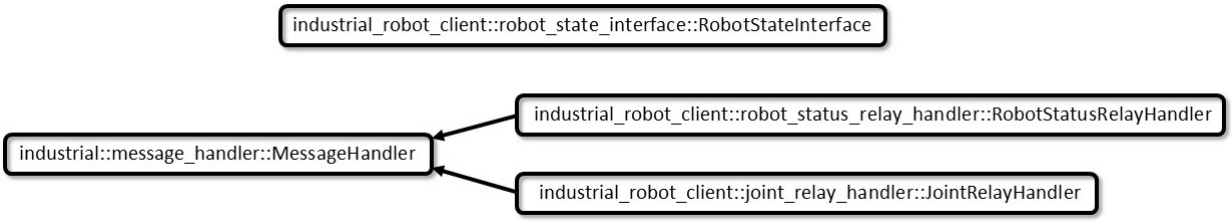


Figure 45: Industrial robot client state publishing APIs

1. Retrieve all joint positions from the robot message;
2. Call the *create_message()* method to generate ROS messages for publishing;
3. Publish messages on the appropriate topic;
4. Reply to the robot controller, if handshake is requested.

The *RobotStatusRelayHandler* is another *MessageHandler*, which can publish the current robot status info in the *robot_status* topic. In any case messages are published at a rate determined by the robot controller code. When new messages are received over the *simple_message* connection, they are immediately published on the appropriate topics.

Position command

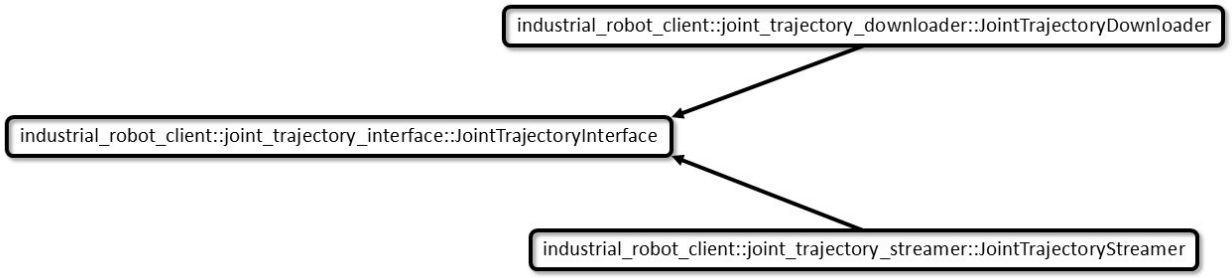


Figure 46: Industrial robot client position command APIs

The *JointTrajectoryInterface* class contains methods to send the robot's joint position to the controller when it receives a ROS trajectory command. This class is an abstract base-class, providing methods for translating ROS messages into *simple_message* messages. However, it only provides an interface specification for transferring the messages to the robot controller. The *JointTrajectoryDownloader* and *JointTrajectoryStreamer* classes provide the required methods to transfer positions to the robot controller, as described below:

JointTrajectoryDownloader: This class is derived from the *JointTrajectoryInterface* class and it implements a method called *send_to_robot()*. This method sends an entire trajectory as a sequence of messages to the robot controller, where the first and the last points are assigned special sequence value (Start/End). The robot controller will execute the trajectory only after getting all the point sequence from the client.

JointTrajectoryStreamer: This class is the same as the preceding class except in the implementation of the *send_to_robot()* method. This method sends trajectory points to the controller in parallel threads. Each position command is sent only after receiving a response from the server. In the robot side, a small position buffer could be implemented, storing some points before starting the execution in order to make the motion smoother. The robot motion starts as soon as buffering is completed or, if not present, after the first point arrives.

In summary, the *JointTrajectoryInterface* class subscribes to the command topic, and listens to the incoming message stream. Each incoming trajectory message is processed using the following steps:

1. Call *trajectory_to_msgs()* method to convert ROS message to *simple _message*. There is a loop over all trajectory points, which performs the following methods:
   a) Call *select()* method to select/reorder joints for robot communication;
   b) Call *transform()* method to alter the reported joint positions, if needed;
   c) Call *calc_speed()* method to calculate a single value velocity representation. The *simple_message* protocol expects a scalar velocity/duration, but the ROS topic provides a velocity value for each joint;
   d) Call *create_message()* method to create the *simple_message*.
2. Call *send_to_robot()* method to send trajectory to the robot controller.

Generic Robot-Interface Nodes

The *industrial_robot_client* package also provides generic nodes that run the basic reference implementation classes described above, which are:

robot_state: This node is running based on *RobotStateInterface*, which can publish the current robot states;

motion_download_interface: This node runs *JointTrajectoryDownloader*, which will download trajectory in sequence to the controller;

motion_streaming_interface: This node runs *JointTrajectoryStreamer*, which will send the joint positions in parallel using threading.

Simple_message package

The Simple Message protocol defines the message structure between the ROS driver layer and the robot controller. The protocol must meet certain requirements, including the format, which should be simple enough that code can be shared between ROS and the controller, for those controllers that support C/C++, or it must be simple enough to be decoded with the limited capabilities of the typical robot programming language, for the ones that do not support C/C++. The format should also allow for data streaming (ROS topic like) and for data reply (ROS service like).

A generic message is structured in this way:

PREFIX

- int LENGTH (HEADER + DATA) in bytes , excluding the LENGTH field itself.

HEADER

- int MSG_TYPE identifies the type of the message (table 9);
- int COMM_TYPE identifies the communications type (table 10);
- int REPLY CODE identifies the service replies (table 11).

DATA

- ByteArray DATA variable length data field, determined by message type and communications type.

The message protocol allows for an arbitrary data payload for message and communications types. However, the client/server model requires that both understand the data payload structure associated with the different message and communications types. A typical robot controller cannot use C++ classes, and thus the developer must understand the message protocol and payload data structure in order to parse it on the robot controller side. A more detailed example of a typed message structure and how it is used is explained below.

A.3 ROBOX DRIVER

The Robox driver interface was created with the aim of allowing a connection between the ROS environment and the Robox RP-1 controller. In order to achieve this goal it was necessary to exploit the *industrial_core* packages and to interact with the Robox development software. This driver communicates with ROS through the *simple_message* interface. Trajectories are streamed to the controller using a message format that captures all the ROS *JointTrajectoryPoint* data: joint positions, velocities, accelerations, and path timing. The controller buffers these points and then processes them in order to control the system.

ID	MSG_TYPE
0	INVALID
1	PING
2	GET_VERSION
10	JOINT_POSITION
11	JOINT_TRAJ_PT
12	JOINT_TRAJ
13	STATUS
14	JOINT_TRAJ_PT_FULL
15	JOINT_FEEDBACK

Table 9: MSG_TYPE identifiers

ID	COMM_TYPE
0	INVALID
1	TOPIC
2	SERVICE_REQUEST
3	SERVICE_REPLY

Table 10: COMM_TYPE identifiers

ID	REPLY CODE
0	INVALID
1	SUCCESS
2	FAILURE

Table 11: REPLY CODE identifiers

A.4 ROS SIDE

The ROS-Industrial core stack, as mentioned previously, contains packages that provide the communication layer to industrial robot controllers. The Robox driver development focused on modify as little as possible the base package provided by ROS-I consortium, in order to strictly adhere to its standard. The most significant change that has been made consists of the communication type between the computer and the controller. In fact the *industrial_core* uses by default a connection based on the TCP protocol, but the Robox controller needed the UDP protocol to receive and send messages. Fortunately the *simple_message* packages already contained the necessary C++ classes and methods able to manage this communication type. Once the code has been studied and understood it has been possible to modify it in order to change that protocol. The *industrial_core* has therefore changed at the "high level", so that its main functionalities remain unaltered.

There are two nodes that are responsible for the main driver's functionalities: the *robox_robot_state_feedback* and the *robox_motion_streaming_ interface*, which were created by inserting these few lines to the bottom of the *CMakeLists.txt* file:

```
add_executable(robox_motion_streaming_interface
src/ROBOX_joint_streamer_node.cpp)
target_link_libraries(robox_motion_streaming_interface
robox_driver
simple_message
${catkin_LIBRARIES})

add_executable(robox_robot_state_feedback
src/ROBOX_robot_state_node.cpp)
target_link_libraries(robox_robot_state_feedback
robox_driver
simple_message
${catkin_LIBRARIES})
```

In this way it is possible to run the nodes simply using the command *rosrun*, but there may be a need to launch them simultaneously and furthermore it is necessary to set the IP address on which the communication will take place, hence a *file.launch* has been create. In fact *roslaunch* is a ROS tool for easily launching multiple ROS nodes, as well as setting parameters on the Parameter Server. Below is shown this *Robox_robot_streaming_interface.launch* file.

```
<launch>

<!--
This launch file provides a socket-based connection to industrial robots
    controller that implements the standard ROS Industrial simple_message
```

```
    protocol. In this case the Robox RP-1 controller. Motion control is
    implemented by STREAMING path data to the robot.

Two nodes are started:
- robot_state : publishes current robot joint positions
- motion_streaming_interface : command robot motion by sending motion
    points to the robot
Usage:
Robox_robot_streaming_interface.launch robot_ip:=<value>
-->

<!--
robot_ip: IP-address of the robot's socket-messaging server -->
<arg name="robot_ip" />

<!--
copy the specified IP address to the Parameter Server, for use by nodes
    below -->
<param name="/robot_ip_address" type="str"
value="$(arg robot_ip)"/>

<!--
robot_state: publishes robot joint positions (from socket connection to
    robot) -->
<node pkg="robox_driver" type="robox_robot_state_feedback"
name="robox_robot_state_feedback"/>

<!--
motion_streaming_interface: sends robot motion commands by STREAMING path
    to robot (using socket connection to robot) -->
<node pkg="robox_driver" type="robox_motion_streaming_interface"
name="robox_motion_streaming_interface"/>

</launch>
```

Motion node

The *robox_motion_streaming_interface* is the node responsible to subscribe the robot's command topic and to send the joints position commands to the robot controller. Once started, it checks for the connection with the controller and, if the operation is successful, it waits for a trajectory to be managed. The trajectory points are subscribed on the *joint_path_command (trajectory_msgs/JointTrajectory)* ROS topic, which has the following structure:

- *std_msgs/Header header*
  Standard metadata for higher-level stamped data types. This is generally used to communicate timestamped data in a particular coordinate frame.
  - uint32 seq: sequence ID, consecutively increasing ID;
  - time stamp: two-integer timestamp that is expressed as *stamp.sec* (seconds) or *stamp.nsec* (nanoseconds);
  - string frame_id: this data is associated with 0 (no frame) or 1 (global frame).
- *string[ ] joint_names*
  A string vector that contains the name of the joint.
- *trajectory_msgs/JointTrajectoryPoint[ ] points*
  - float64[ ] positions
  - float64[ ] velocities
  - float64[ ] accelerations
  - float64[ ] effort
  - duration time_from_start

Once received the data, according to what explained previously about the *industrial_robot_client* package, the node has to translate this ROS message into a *simple_message* message.
More in detail, the *trajectory_msgs/JointTrajectoryPoint* message is mirrored with the *joint_traj_pt* message, which has the structure shown in Table 12.

Member	**Type**	**Value**	**size**
Message Type	JOINT_TRAJ_PT	11	4 bytes
Communications Type	Comm_type	ANY	4 byte
Reply Type	Reply type	ANY	4 byte
Data			
sequence	shared_int	ANY	4 byte
joints	shared_real[10]	ANY	40 byte
velocity	shared_real	ANY	4 byte
duration	shared_real	ANY	4 byte

Table 12: Joint trajectory point message

This point differs from the ROS trajectory point in the following ways:

- The joint velocity in an industrial robot standard way (as a single value). Velocity calculation computes the percentage of maximum speed for the "critical joint" (closest to velocity-limit).
- The duration is somewhat different than the ROS timestamp. The timestamp specifies when the move should start, while the duration is how long the move should take. A big assumption is that a sequence of points is continuously executed.

After the trajectory has been converted, it is ready to be sent to the controller. It has been chosen to use a *streamer* method to perform this operation, so the *robox_motion _streaming_interface* node sends independent joint values to the controller in separate threads. The client will send data one at a time. When it receives the server response it sends the next point. This allows the server (controller) to determine the rate at which points are sent. How these data will be stored and managed is explained in the "Controller side" section.

After trying and testing the driver with the *joint_traj_pt* message type, it was chosen to change it, because it limits the number of data sent to the controller, for example the accelerations are not taken into account although they can be written in the ROS topic. Before creating an own customized message it was searched something useful inside the *simple_message* protocol and it was found the *joint_traj_pt_full* message type, which has the structure shown in Table 13.

Member	**Type**	**Value**	**size**
Message Type	JOINT_TRAJ_PT_FULL	11	4 bytes
Communications Type	Comm_type	ANY	4 byte
Reply Type	Reply type	ANY	4 byte
Data			
robot_id	shared_int	ANY	4 byte
sequence	shared_int	ANY	4 byte
valid_fields	shared_int	ANY	4 byte
time	shared_real	ANY	4 byte
positions	shared_real[10]	ANY	40 byte
velocities	shared_real[10]	ANY	40 byte
accelerations	shared_real[10]	ANY	40 byte

Table 13: Joint trajectory point full message

The *robot_id* represent the robot identifier, while the *valid_fields* is a bit-mask that indicates which “optional” fields are filled with data (1 = time, 2 = position,

4 = velocity, 8 = acceleration). With this type of structure a significant amount of the ROS message can be exploited, certainly more than before. The change of message type was possible by modifying the code of the *JointTrajectoryStreamer* and *JointTrajectoryInterface* classes, in particular the *trajectory_to_msgs()*, *select()*, and *create_message()* methods are changed in order to manage the data differently.

Feedback node

The *robox_robot_state_feedback* is the node responsible to publish the current robot position and status at regular time intervals after receiving the position data from the robot controller. Once started, as in the previous node, it checks for the connection with the controller and if the operation is successful it waits for the feedback data, which is a *simple_message* message and it has the following structure:

Member	**Type**	**Value**	**size**
Message Type	JOINT_POSITION	10	4 bytes
Communications Type	Comm_type	ANY	4 byte
Reply Type	Reply type	ANY	4 byte
Data			
sequence	shared_int	ANY	4 byte
joints	shared_real[10]	ANY	40 byte

Table 14: Joint message

Once the data is received, the conversion to be performed is more or less the same as the motion case, but this time it happens exactly in the opposite way. The node has to translate this message into a ROS message and publish it on the appropriate topic in order to allow the checking of the joint state. In this project it was chosen to check only the position of the joints.

A.5 CONTROLLER SIDE

After realizing how the driver works regarding the ROS environment, it was necessary to create the controller interface. The device on which the communication takes place is the *Robox RP-1 controller* and its interface was made with the *RDE - Robox Development Environment* (summarized in the previous chapters and explained in detail at [135]). As with the ROS side it was possible to split the software application into two parts, one for the motion and one for the feedback. Therefore, two tasks are written by using the Robox R3 language. The R3 programming language instructions are the classic instructions of the Structured

Text programming language, with the addition of some specific ones, strictly related to the problems that the R3 language is intended for.

Motion task

The *task_motion* is the task responsible to manage the incoming trajectory data. It is able to receive and store them inside the controller memory, it is also in charge of sending the correct reply to the ROS side with the view to allowing the flow of data, according with the streamer method.

In order to develop this task, first of all it was necessary to define the variables that are used for the communication:

- A structure used to receive data from the client

```
LIT size_buffer_receive 68
STRUCT_P buff_data
U32 LENGHT
I32 MSG_TYPE
I32 COM_TYPE
I32 Reply_TYPE
I32 sequence
FLOAT Joints_DATA[10]
FLOAT velocity
FLOAT duration
END_STRUCT_P
buff_data buff_receive
```

- A structure used to answer to the client

```
LIT size_buffer_receive 16
STRUCT_P buff_reply
U32 LENGHT
I32 MSG_TYPE
I32 COM_TYPE
I32 Reply_TYPE
END_STRUCT_P
buff_reply buff_answer
```

- A structure used to store each trajectory data received from the client

```
STRUCT_P buff_array
FLOAT velocity
FLOAT duration
FLOAT Positions[10]
```

```
END_STRUCT_P
buff_array trajectory[100]
```

- Variables used to set the communication detail and other ones used by the functions

```
string ip\_server\_addr = "IP" ; IP server address
U32 udp_server_port = Port ; Port server number
U32 udp_client_port
I32 codOpen, codSend, codRecv, reso
U32 ip_server_num
U32 ip_client_num
```

Once the task is started, the function *initTask()* is called, which has mainly two objectives:

- Open the server communication

```
; IP SERVER string conversion
reso = str_to_ipaddr(ip_server_addr, ip_server_num)
; diagnostic
if (reso = 0)
; Error
alarm_set(4000) ; "4000 err. adress Server"
endif

; Let's open SERVER comunication
codOpen = UDP_OPEN_SERVER(udp_server_port, ip_server_num)

; diagnostic
if (codOpen < 0)
; Error
alarm_set(4002,0,codOpen) ; "4002 err. udp_open (client)"
endif
```

The function *UDP_OPEN_SERVER()* opens a server type UDP communication. It has the following syntax:

```
I32 udp_open_server(U32 port, I32 addr)
port: port number on which the server is listening.
addr: interface IP address on which the server is listening.

Result: Communication management handle. It is a number from 0 to
    99 that uniquely identifies the communication. It is used in
```

```
    all other UDP management functions. Negative values indicate
    an error
```

The function *str_to_ipaddr()* converts the IP address string into an integer as requested by the previous function.

```
I32 str_to_ipaddr(STRING address, U32 addr)
address: Variable containing the string to be converted.
addr: Variable in which the function writes the result.

Result: <> 0 if the conversion was successful.
0 otherwise.
```

- Checking if the connection is ready to be used

```
bool handshakeDone = false
codRecv = 0
while(NOT handshakeDone)
codRecv = UDP_RECV_FROM(codOpen, buff_answer, 1, udp_client_port,
    ip_client_num)
if(codRecv > 0)
handshakeDone = true
codSend = UDP_SEND_TO(codOpen, buff_answer, 1, udp_client_port,
    ip_client_num)
end_if
dwell(0.005) ; 5 ms delay
end_while
```

As mentioned in the ROS side section, the *Motion node* needs to check if the controller is connected. With the aim of carrying out this operation the ROS client starts to send, at regular time interval, a single constant value to the robot controller IP address, until it receives an answer of the same value as the one sent. The functions used to allow communication are described below

```
I32 udp_recv_from(I32 idx, STRING buffer, I32 lenBuffer, U32
    remPort, U32 remIP )
idx: UDP communication handler number.
buffer: Generic buffer intended for containing the received data
    package. It is generally an user structure.
lenBuffer: Number of maximum bytes to be received.
remPort: Number of the remote port from which data was received
    (it is written by the function itself).
remIp: IP address from which data was received (it is written by
    function itself).
```

```
    Result: > 0 (success) represents the number of characters
        received. Negative values indicate an error.

    I32 udp_send_to(I32 idx, STRING buffer, I32 lenData, U32 remPort,
        U32 remIP)
    idx: UDP communication handler number.
    buffer: Generic buffer containing the data package to be
        transmitted. It is generally an user structure.
    lenBuffer: Number of maximum bytes to be transmitted.
    remPort: Number of the remote port to which data is transmitted.
    remIp: IP address to which data is transmitted.

    Result: > 0 (success) represents the number of characters sent.
        Negative values indicate an error.
```

When the task is initialized, the connection is stable and the program is ready to start the *MAIN_LOOP*. A single message, containing the joints information, is received and if the operation is successful the data is stored in an array of structure at the index provided by the message itself (the sequence number). Before receiving another data the task has to send a reply through the same IP address and port number. Once the client receives the answer, it can send a new message and the loop starts again until the trajectory points are finished.

```
_MAIN_LOOP_

; let's get the message from the client
codRecv = UDP_RECV_FROM(codOpen, buff_receive, size_buffer_receive,
    udp_client_port, ip_client_num )
; wait for the server reception
if (codRecv <= 0)
; Error
ALARM_SET(4003,0,codRecv)  ; "4003 error recv_from (server)"
endif

if (codRecv > 0)
; ...question receive OK
; let's prepare the answer
buff_answer.LENGHT = size_buffer_answer - 4 ; the LENGHT data is not
    considered
buff_answer.MSG_TYPE = buff_receive.MSG_TYPE ; same MSG_TYPE
buff_answer.COM_TYPE = 3 ; service reply
buff_answer.Reply_TYPE = 1 ; success

; storing of the trajectory point
```

```
trajectory[buff_receive.sequence].velocity = buff_receive.velocity
trajectory[buff_receive.sequence].duration = buff_receive.duration
i=0
while(i<10)
trajectory[buff_receive.sequence].Positions[i] =
    buff_receive.Joints_DATA[i]
i=i+1
end_while

; Send the answer
codSend = UDP_SEND_TO(codOpen, buff_answer, size_buffer_answer,
    udp_client_port, ip_client_num)

if (codSend < 0)
; Error
ALARM_SET(4004,0,codSend)  ; ; "4004 error udp_send_to (server)"
endif
endif

END_MAIN_LOOP
```

According to the decision of changing the ROS side message type, it was necessary to make changes to the motion task too. It was altered the structure used to receive the trajectory from the client and the one used to store that data:

```
LIT size_buffer_receive 152
STRUCT_P buff_data
U32 LENGHT
I32 MSG_TYPE
I32 COM_TYPE
I32 Reply_TYPE
I32 robot_id
I32 sequence
I32 valid_fields
FLOAT time
FLOAT Joints_DATA[10]
FLOAT velocities[10]
FLOAT accelerations[10]
END_STRUCT_P
buff_data buff_receive

STRUCT_P buff_array
FLOAT time
FLOAT Positions[10]
FLOAT Velocities[10]
```

```
FLOAT Accelerations[10]
END_STRUCT_P
buff_array trajectory[100]
```

Afterwards it was only changed the storing of the trajectory point inside the *MAIN_LOOP* in order to save all the velocities and accelerations values coming from the client.

Feedback task

The *feedback_task* retrieves the controller state (robot position) and sends it to the client connected to the state server port. It does not wait for any request from the client, at the moment when the connection is established it will automatically start sending the information at regular time intervals.
The variables used with this task are the following:

- A structure used to send the feedback data to the client

```
LIT size_buffer_feedback 60
STRUCT_P buff_fb
U32 LENGHT
I32 MSG_TYPE
I32 COM_TYPE
I32 Reply_TYPE
I32 sequence
FLOAT Joints_DATA[10]
END_STRUCT_P
buff_fb buff_feedback
```

- Variables used to set the communication detail and other ones used by the functions.

```
string ip_server_addr_fb = "IP" ; IP server address
U32 udp_server_port_fb = Port ; Port server number
U32 udp_client_port_fb
I32 codOpen_fb, codSend_fb, codRecv_fb, reso_fb
I32 i_fb
U32 ip_server_num_fb
U32 ip_client_num_fb
```

Once the task is started, exactly as for the motion case, the function *initTask()* is called, which has the same goal of the previous case. The only difference is the server port number. If the connection is stable, the *MAIN_LOOP* can start. It

simply retrieves the Joints positions from the controller and it sends the data to the client.

```
_MAIN_LOOP_

; prepare the feedback message

buff_feedback.LENGHT = size_buffer_feedback - 4 ; the LENGHT data is not
    considered
buff_feedback.MSG_TYPE = 10
buff_feedback.COM_TYPE = 1 ; Topic
buff_feedback.Reply_TYPE = 0 ; Invalid
buff_feedback.Joints_DATA[0] = CP(1)
buff_feedback.Joints_DATA[1] = CP(2)
buff_feedback.Joints_DATA[2] = CP(3)
buff_feedback.Joints_DATA[3] = CP(4)
buff_feedback.Joints_DATA[4] = CP(5)
buff_feedback.Joints_DATA[5] = CP(6)
buff_feedback.Joints_DATA[6] = CP(7)
buff_feedback.Joints_DATA[7] = CP(8)
buff_feedback.Joints_DATA[8] = CP(9)
buff_feedback.Joints_DATA[9] = CP(10)

; Send the feedback
codSend_fb = UDP_SEND_TO(codOpen_fb, buff_answer_fb,
    size_buffer_answer_fb, udp_client_port_fb, ip_client_num_fb)

if (codSend_fb < 0)
; Error
ALARM_SET(4004,0,codSend_fb)  ; ; "4004 error udp_send_to (server)"
endif

dwell(1) ; delay that defines the interval for sending data

END_MAIN_LOOP
```

The keyword *CP* gives access to the real position value of the respective joint detected by the transducer. It has this syntax:

```
REAL cp (I32 n)
n: Index of the axis on which the real position is to be read
Min: 1
Max:32
```

In this project it was chosen to check only the position of the joint, according to the *joint_position simple_message* message of ROS-Industrial. With this solution

it is possible to check up to ten joint values at the same time. In the probable case in which they are less than this, the task will simply send a zero value for the missing joints.

Once the functionality of the driver has been checked (also on different robots), it has been possible to use it in the Experiment. Another potentiality of the use of the driver is the possibility to interface the robot with other ROS packages. Just to show an example Figure 47), show the interface with *MoveIt!*, which allow the automatic planning of the motion in an simulation environment.

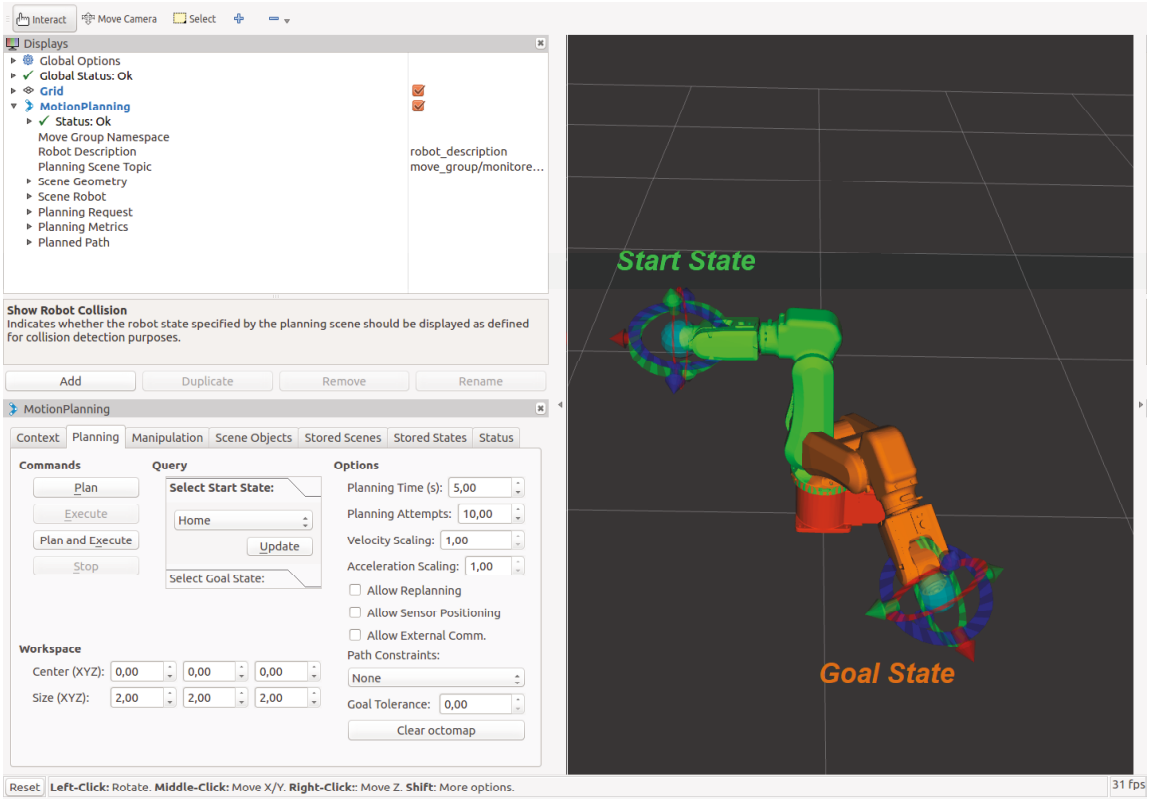


Figure 47: MoveIt! - Rviz motion planning interface

DICHIARAZIONE DI CONFORMITÀ DELLE TESI
PER IL CONSEGUIMENTO DEL TITOLO DI DOTTORE DI RICERCA
(DICHIARAZIONE SOSTITUTIVA DI ATTO NOTORIO E DI CERTIFICAZIONE
(artt. 46-47 del D.P.R. 445 del 28.12.00 e relative modifiche)

Il/La sottoscritto **PAGANI ROBERTO** nato il **02/04/1992** a **BRESCIA** in Provincia di **BRESCIA**
Dottorato di ricerca in **INGEGNERIA MECCANICA E INDUSTRIALE (DRIMI)**

a conoscenza del fatto che in caso di dichiarazioni mendaci, oltre alle sanzioni previste dal Codice Penale e dalle Leggi speciali per l'ipotesi di falsità in atti ed uso di atti falsi, decade dai benefici conseguenti al provvedimento emanato sulla base di tali dichiarazioni,

DICHIARA

sotto la propria responsabilità, ai fini dell'ammissione all'esame finale per il conseguimento del titolo di Dottore di ricerca,

di essere a conoscenza che,

in conformità al Regolamento dell'Università degli Studi di Brescia per l'ammissione all'esame finale ed il rilascio del titolo per il conseguimento del titolo di Dottore di Ricerca, è tenuto a depositare all'U.O.C. Dottorati e Scuole di Specializzazione:

- n. 1 copia in formato cartaceo della propria tesi di dottorato e **l'esposizione riassuntiva (*abstract)* in italiano, se la redazione della tesi è stata autorizzata in lingua straniera;**
- n. 2 copie della tesi su DVD o CD-ROM per il deposito presso le Biblioteche Nazionali di Roma e di Firenze;

DICHIARA inoltre

- che il contenuto e l'organizzazione della tesi sono opera originale e non compromettono in alcun modo i diritti di terzi,
- **che sarà consultabile** immediatamente dopo il conseguimento del titolo di Dottore di ricerca, in quanto non è il risultato di attività rientranti nella normativa sulla proprietà industriale, non è stata prodotta nell'ambito di progetti finanziati da soggetti pubblici o privati con vincoli alla divulgazione dei risultati, e non è oggetto di eventuali registrazioni di tipo brevettale o di tutela.
- che l'Università è in ogni caso esente da qualsiasi responsabilità di qualsivoglia natura, civile, amministrativa o penale e sarà tenuta indenne da qualsiasi richiesta o rivendicazione da parte di terzi;
- che la tesi in formato elettronico (DVD o CD-ROM) è completa in ogni sua parte ed è del tutto identica a quella depositata in formato cartaceo all'U.O.C. Dottorati e Scuole di Specializzazione dell'Università degli Studi di Brescia e inviata ai Commissari. Di conseguenza va esclusa qualsiasi responsabilità dell'Ateneo per quanto riguarda eventuali errori, imprecisioni od omissioni nei contenuti della tesi.

Dichiara inoltre di essere consapevole che saranno effettuati dei controlli a campione. Eventuali discordanze od omissioni potranno comportare l'esclusione dal dottorato di ricerca;

Luogo e data

Firma del dichiarante