Fakultät für Maschinenwesen
Lehrstuhl für Angewandte Mechanik

Hierarchical Joint Control of Humanoid Robots

Sensing, Actuation and Communication Systems

Valerio Favot

Vollständiger Abdruck der von der Fakultät für Maschinenwesen der
Technischen Universität München zur Erlangung des akademischen Grades eines

Doktor-Ingenieurs (Dr.-Ing.)

genehmigten Dissertation.

Vorsitzender: Prof. dr. ir. Daniel Rixen

Prüfer der Dissertation:

1. Prof. Dr.-Ing. habil. Heinz Ulbrich, i. R.
2. Prof. Carlo L. Bottasso, Ph.D.

Die Dissertation wurde am 03.03.2016 bei der Technischen Universität München eingereicht
und durch die Fakultät für Maschinenwesen am 23.10.2016 angenommen.

To my parents and my sister,
whose unconditional support and love
have helped me through my entire life.

Abstract

The focus of this thesis is joint control for humanoid robots from a mechatronic perspective. Simulations and experiments were conducted using the robot LOLA developed at the Institute of Applied Mechanics of the Technischen Universität München. The structure of the resulting decentralized sensor/actuation network and real-time communication is described in detail. A novel implementation of a motor velocity algorithm are presented. The elastic joint model parameters are estimated during experiments on the real robot. Methods for joint control to compensate for gear elasticity are discussed and compared.

Zusammenfassung

Thema der Arbeit ist die mechatronische Implementierung von Gelenkreglern für humanoide Roboter. Als Experimentier-Plattform steht der am Lehrstuhl für Angewandte Mechanik der Technischen Universität München entwickelte Zweibeiner LOLA zur Verfügung. Die Struktur des dezentralen Sensor/Antriebsnetzwerkes ist im Detail beschrieben. Die Implementierung der Kommunikationsprotokolle für die Sensordatenerfassung, Verarbeitung und Echtzeit-Kommunikation wird vorgestellt. Eine neue Implementierung für einen Motor Geschwindigkeitsschätzer wird präsentiert. Die Parameter des elastisches Gelenkmodels werden unter Zuhilfenahme von experimentellen Daten geschätzt. Schließlich werden verschiedene Regelungsstrategien für die elastischen Gelenkgetriebe diskutiert und verglichen.

Acknowledgment

In this thesis I present a large part of the work I've undertaken during six years of research at the Institute of Applied Mechanics, Technische Universität München. Many people contributed to the LOLA project and made this work possible.

The first mention is for my supervisor Professor Heinz Ulbrich for the opportunity to work on this research project, for his support and guidance and for providing an excellent and inspiring research environment. I'd also like to thank Professor Carlo Bottasso and Professor Daniel Rixen for serving on my thesis defence committee.

Having the opportunity to work with many talented people at the Institute of Applied Mechanics was a pleasure. I am deeply grateful to the LOLA research group for their motivation, their supportive encouragement and for giving me the possibility to learn so much. The precision, dedication and eclectic research approach of Dr. Sebastian Lohmeier has always been a reference point for me. I'd like to thank Dr. Thomas Buschmann for his patience, support and guidance. I've had the possibility to learn so much from his vast knowledge in the fields of robotics and program design. They have been an invaluable source of inspiration from the beginning of my research project. I'm deeply thankful to Dr. Markus Schwienbacher for his help, encouragement and understanding. We have had many joyful times as well as hard moments that we manged to overcome. I would like to thank the other members of the robotics group namely Dr. Alexander Ewald, Robert Wittmann, Arne Hildebrand, Dr. Jörg Baur, Christoph Schütz and Dr. Julian Pfaff for their help and inspiring exchange of ideas.

I'd like to thank the staff of the institute's mechanical and electrical workshops. Simon Gerer, Philip Schneider and Tobias Schmidt have done incredible work manufacturing and, sometimes, repairing parts of LOLA. The great experience, enthusiasm and pragmatic approach of Georg "Schorsch" Mayr for solving any kind of electronics related problem has been a great source of inspiration to me. I'm thankful to Dr. Thomas Thümmel for his support during my time at the institute and for his project management capabilities. I'd like to express my deepest thanks to Dr. Thomas Buschmann, Dr. Markus Schwienbacher, Dr. Alexander Ewald and Robert Wittmann for reading the draft of this thesis and for their helpful comments and suggestions.

Finally, I would like to thank my parents, Sergio and Claudia and my sister Federica for their constant support, encouragement and unconditional love.

January 21, 2017 Valerio Favot

" I've seen things you people wouldn't believe.
Attack ships on fire off the shoulder of Orion.
I watched C-beams glitter in the dark near the Tannhauser gate.
All those moments will be lost in time...
like tears in rain... "

– Roy Batty, "Blade Runner"

Contents

1 Introduction

The concept of automation, as we understand it today, was introduced in the 1940s by the Ford Motor Company. This term incorporates two fundamental concepts: the use of a mechanical device capable of performing a specific task and the use of some kind of controlling system to direct the device's actions. Such kind of devices drastically changed industrial production and *Robots* are the state of the art in industrial automation. The machines currently used in the industry are fixed robots which execute a very specific task with high precision and at high speed.

Another category of robots that have been developed in various forms, and for different purposes are *Mobile Robots*. They can be roughly divided between wheeled and legged robots. Under the latter category, different types of machines have been realized with a variable number of limbs. The two legged variant are generally of humanoid form and try to reproduce human sensing and motion capabilities.

The first examples of a legged humanoid form robots date back more than 50 years to the 1970s with the presentation of WABOT-I by Professor Ichiro Kato at Waseda University. Subsequent interest in humanoids robots has grown constantly, especially in the last 30 years with numerous companies and universities having contributed widely to the development of this research field.

The idea of a machines capable of working, facilitating or cooperating with humans in daily activities is very appealing. The fields of application for humanoid robots are many and varied. Service robotics is one of the most promising, where such machines can be potentially used in industrial, medical and office environments. Those are spaces tailored for human activities and having working companions with the same abilities and shape as humans seems to be the best overall solution. Moreover, the use of human language and imitation of natural human gestures can facilitate the acceptance of these machines.

In recent years, the DARPA[1] Robotic Challenge[2] (DRC) has promoted the deployment of humanoid robots in harsh environments or in case of catastrophes. In dangerous situations, the use of capable remotely controlled machines instead of humans can be very advantageous and reduce the risk to human operators or rescue teams.

From an engineering point of view, robotics in general and human-like robots in particular, are the best example of a complete mechatronic system. Such systems represent the perfect intersection of different disciplines such as mechanics, electronics, information and sensor technology, control theory, computer programming, artificial intelligence, speech synthesis and expression recognition. All of these research fields contribute to the development and improvement of humanoids in terms of performance and human interaction.

1 U.S. Department of Defense's research and development

2 DARPA Robotics Challenge. U.S. Defense Advanced Research Project Agency. Oct. 2013. URL: http://www.theroboticschallenge.org.

1.1 Literature Review and Related Works

Short Review of the Cultural Background

The word "ROBOT" has its origin in the 1920's play "Rossum's Universal Robots" from the Czech author Karel Čapek's [144] and laterally means "*forced worker*" in the Czech language.

Nevertheless, the concept of a human-like machine have much older roots. Al-Jazari and Leonardo Da Vinci each designed an "AUTOMATON" in the 13th and 15th century respectively. The "*Golem*", mentioned in the Bible and Jewish folklore, is the perfect servant worker made of inanimate matter and created to fulfill the orders of its master.

In the 20th century the idea of human-like robots has been brought to the general public through many science-fiction novels and films. The works of Isaak Asimov and Philip K. Dick, among many others, deserves to be mentioned for having inspired the imagination of generations of scientists, novelists and the public with innovative and original ideas about humanoid robots and their integration into society.

In recent years, the majority of exposure of about humanoid robots to the general public has been through science-fiction films like *Terminator*, *Bicentennial Man*, *I, Robot*, *A.I.*, *Ex-Machina* etc.. Classic films such as *Star Wars* and *Star Trek*, also deserve to be mentioned. Although excellent examples of science-fiction plays, they tend to provide unrealistic expectations of the actual capabilities of such machines and, in some cases, can evoke a misplaced perception of threat by this technology. To promote general acceptance of human-like robots, the public must be educated about their real performance and limits. This task must be accomplished by the researchers who develop and work with these machines, since they alone are aware of the actual limits and possibilities of this technology.

Relevant Technical Literature

The body of literature concerning humanoid robotics is wide and constantly increasing. Hardware and software improvements which have enabled the implementation of complex machines and algorithms have also contributed to the growth in this field.

The books [124, 125] must be cited as being comprehensive references for developing, analyzing, modelling, simulating and control of robots.

Lewis [84] addresses the control problem, applied to various types of robots. The authors present a thorough discussion regarding the analysis and implementation of control methods from the classic PID[3] to more advanced model based algorithms such as robust or adaptive controllers.

In Kemp et al. [79] a general discussion about the application of humanoids is given. The authors present the results achieved by different research groups on various humanoid platforms. They describe the work on legged robots, manipulation and the problem of communication between humans and robots. Specific information on legged robots can be found in Kajita and Espiau [73]. The authors present specific methods for modeling, simulation and control approaches for bipedal and multi-legged robots.

3 Acronym for Proportional Integral Derivative controller

In Gienger [58] and Lohmeier [88], the authors give a comprehensive description of hardware design aimed towards the realization of humanoid-legged robots. Buschmann [20] presents a detailed survey of simulation tools, algorithms for gait, step and trajectory generation, stabilization and control of humanoid robots. In Schwienbacher [118] efficient algorithms for kinematic computation, self-collision avoidance and angular momentum tracking for biped robot are discussed.

Historically, the research on bipedal humanoid robots started in 1973 with the development of WABOT-I at Waseda University by Kato. The system was hydraulically actuated and the first working robot of human-like scale. Many other robots were subsequently developed at Waseda University, with WABIAN-2 being the last one [103].

Since then, Japan has a long history of research groups at universities and industry developing humanoid robots prototypes.

Inoue from JSK-Laboratory at Tokyo University, in cooperation with Kawada Industries, built the robot H6 and successor H7. Important characteristics of these robots are the actuated toe joint, the development of a vision system for autonomous walking [101, 102]. In 2010, Urata et al. [143] developed a high power electrically actuated robot. In 2007, Nakanishi et al. [99] built a complex experimental Musculo-Skeletal humanoid.

Kawada Industries also worked with the Japanese National Institute of Advanced Industrial Science and Technology (AIST) research center on the “Humanoid Robotics Project" (HRP). During this collaboration, the full-size human-like robots HRP-2 to HRP-4 were developed. While HRP-2 [76] and HRP-3 [77] have the appearance of the robots depicted in the manga series “PATLABOR", the last model HRP-4 has the appearance of a young female. HRP-2 is one of the most commercially successful of such platforms. Many research groups all over the world have purchased a prototype and focused on developing software algorithms for motion and human-robot collaboration.

The car manufacturer Toyota have presented several so called “Partner Robots". The project is focus on service robotics to support humans in medical, mobility, housework and work spheres. The company have also presented in 2006 a fast running robot [134] which could reach 7 km/h.

Honda started the development of humanoid robots in 1986. The research was initially conducted in secrecy and focused on bipedal locomotion only. In this phase, the company developed the so called “E-series” (E0 to E6) machines. The successive P-series (P1 to P3), developed between 1993 and 1997, also had actuated arms and grippers. P1 to P3 were of height between 190 cm and 160 cm and were the first examples of fully actuated man-like robots. In 2000 Honda presented its last model of humanoid: ASIMO. At a height of 120-130 cm, the ASIMO series is smaller than its predecessors and has a number of Degrees of Freedom between 26 and 57, depending on the generation. Honda ASIMO robots are probably the most influential from a research perspective as well as being the most famous prototypes known to the general public. In addition, the mechanical, sensor and actuation approaches as well as the motion planning and control strategies applied by the company are the most adopted. While many details on the development, hardware and software

of Honda's robots has been at first release only in the form of patents, in recent years some scientific papers have been published [135, 136, 137, 138]. The running speed of ASIMO is 10 km/h [135] which is the highest speed currently achieved by a humanoid robot.

The company Boston Dynamics, recently acquired by Google Inc., has developed two prototypes of anthropomorphic robots based on a project originally founded by DARPA. The first one is PETMAN [6], developed based on the same technology used to build the four leg robot BIGDOG. PETMAN has hydraulic actuators and can walk in a straight line at the speed of 7 km/h. The second humanoid built by the company in 2013, is the Agile Anthropomorphic Robot ATLAS [1]. ATLAS is 180 cm tall, weighs almost 150 kg, has 28 hydraulically actuated Degrees of Freedom and has an external power supply.

In 2000 the Korea Advanced Institute of Science and Technology (KAIST) started the development of the bipedal robot KHR-0 under the supervision of Professor Oh Jun-ho. The first prototypes had no upper body, until the development of KHR-2 in 2004. The successive year, KAIST presented the first version of its HUBO robot prototype. The following model HUBO-2 is 130 cm height, weighs almost 45 kg, has 41 Degrees of Freedom and can walk at the speed of 1.5 Km/h. The last version of the robot is DRC-HUBO which was developed to participate at the DRC. DRC-HUBO is capable of bipedal walking and is also equipped with leg mounted wheels for rapid movement. The team won the DRC in 2015 with DRC-HUBO.

In 2010, the German Aerospace Center (DLR) presented the DLR BIPED [106] and in 2014 the humanoid TORO [47]. TORO is based on the DLR-KUKA-LIGHTWEIGHT-ROBOT-III and has electrically actuated joints with integrated torque sensors. The use of joint torque sensors adds a certain level of compliance with the operational environment and further facilitates interaction and collaboration with humans.

At the Institute of Applied Mechanics of the Technischen Universität München, Gienger [58] and Löffler [86] have developed the humanoid robot JOHNNIE. JOHNNIE is 175 cm tall, weighs 50 kg, has 14 driven joints actuated by electrical DC motors and was the first biped robot capable of walking autonomously in an unknown environment.

The biped walking machine LOLA, developed by Lohmeier [88] and Buschmann [20, 25, 26], is the second generation bipedal walking machine developed at the Institute of Applied Mechanics[4]. Based on the experience garnered with JOHNNIE, LOLA's hardware was drastically improved in comparison with JOHNNIE. LOLA is characterized by extremely lightweight construction and that the 25 joints are actuated by high power density Permanent Magnet Brushless Synchronous Motors. LOLA is 180 cm tall, weighs 60 kg and can walk 3.6 km/h. LOLA is equipped with a stereo camera head [88] and can walk autonomously. Obstacle recognition vision software was developed by von Hundelshausen [146] at the Autonomous Systems Technology Institute[5] and adapted to LOLA. The capabilities of the robot of walking in unknown environments and of recognizing and avoiding obstacles were demonstrated in more than 25 presentations at the *Hannover Messe* 2010[6].

4 Contributors to the development of LOLA are also: Dr.-Ing. Markus Schwienbacher, Georg Mayr, Dr.-Ing. Mathias Bachmayer, the author and students.

5 Universität der Bundeswehr. `http://www.unibw.de/lrt8/index_html-en`

6 The world's largest industrial trade fair (http://www.hannovermesse.de).

1.2 Overview of the Thesis

In this thesis the problem of efficient joint control for humanoids robots is addressed from a mechatronic prospective. The platform used in is the humanoid robot LOLA, developed at the Institute of Applied Mechanics of the Technischen Universität München. A picture of the robot can be seen in Figure 2.1 and in Figure 2.2 the joint structure of the system is shown.

The main objectives of this work are: *a*) the development and implementation of a hierarchical network of distributed sensing and actuation units and *b*) the implementation of control methods to efficiently control the robot's joints in the stiff and elastic case.

These aspects of development are frequently considered of "minor" interest in complex robotic project such as humanoids or mobile robotic platforms. Typically, other *"high level"* aspects are the primary focus of the researchers, such as the optimization of motion and, in particular for bipedal locomotive machines, the feasibility of walking. However, the challenges of generating stable movements and, the stabilization and control of such complex machines are extremely important and have no trivial solution.

Nevertheless, the reliability of the electronics, communication systems and actuation devices is of fundamental importance to achieve stability, robustness and performance with mechatronic systems. To enable the development of new strategies and to solve the problem of controlling such complex systems, the devices in the underlying sensor, communication and power subsystems must be carefully chosen and be capable of delivering the performance required by the project objectives.

A unified and scalable method of interfacing system sensors and actuators is introduced in this thesis. The sensing/actuation network is implemented with custom-made interface boards developed at the Institute of Applied Mechanics. These boards are the key nodes for data exchange with the central control unit of the system, the sensors and the actuators. The communication protocol implemented to interface the sensors, actuators and the central control unit are presented. The functionality of the "bare metal" software developed for the custom made boards is presented. To efficiently read the motor position sensors, a custom made interface implemented using programmable logic devices has also been developed. Different algorithms for local motor velocity estimation are discussed and simulated. The best performing methods are implemented in the available programmable logic devices and tested in the system, improving the capabilities of the robot control and system analysis possibilities. The actuation train of the robot's joints are modelled taking into account gear elasticity. The model parameters are estimated using and implementation of the recursive least square algorithm. The latter is suitable for both the off-line and real-time estimation of joint parameters. Various control algorithms are presented and analyzed through simulation and experiments. The results achieved with the classical PID joint control method are also discussed. Moreover, two model based control laws are presented to improve the performance of the control system considering joint elasticity.

Chapters Summary

This thesis is divided into five chapters.

In Chapter 2 an overview of the humanoid robot LOLA is presented. The major focus is on the hardware used for joint motion and control, i. e., the actuation train of the system.

A custom designed local multi-axis joint controller and sensor interface device is presented. The communication protocols used to read the sensor data, control the actuation units and communicate with the central control unit are presented.

In Chapter 3 the development and implementation of the motor position sensor interface is discussed. Taking advantage of the signal processing capabilities of the joint controller, an implementation of an efficient velocity estimation algorithm is presented.

In Chapter 4 the control structure of LOLA is presented, focusing on the low level joint controllers. A model for the actuation train of the elastic joints of the robot is presented and a method for the estimation of the joint model parameters is proposed.

In Chapter 5 three joint control methods are discussed. The classic PID controller and two model based control laws are analyzed in simulation and experiments.

In Chapter 6 the summary and final conclusions of this work are presented in addition to suggestions for future development of similar projects are given.

2 Actuation, Sensor and Control Network of Lola

2.1 Introduction

The development of a humanoid robot is a complex mechatronic task. On the one side the mechanical structure of the anthropomorphic system must provide satisfactory dynamical and robust characteristics while, on the other side, the electronic system architecture must be reliable and deliver high computational performance.

The number of sensors installed on humanoid robots is constantly growing to improve the performance and extend the perception capabilities of the systems. Many different kinds of sensors, depending on the desired task and characteristics of the system, are available nowadays with different interfaces. The quantity of information that these systems must process, evaluate and react to is consequently increasing.

The number of joints in humanoids robots is growing as more and more functions are integrated in order to realize “human-like" movements and behaviors, i. e., biped walking and complex manipulation tasks. All these Degrees of Freedom (DoF) must be controlled.

For these reasons the complexity of the electronic system and the processing requirements are increasing as well. As in every mechatronic system, humanoid robots generally have an electronics architecture composed of: at least one Central Control Unit (CCU), Actuation Units (AUs) and Sensors.

Usually, the following kinds of topologies are realized:

- Centralized architecture: all sensor signals are connected to a general purpose I/O board which also is in general responsible for delivering the commands from the CCU to the actuation devices [87].

- Decentralized architecture: specialized circuit boards distributed on the robot connect the sensors and receive the commands for the actuation devices from the CCU. The communication between CCU and distributed boards is via a communication bus [103, 151].

- A combination of both centralized and decentralized: some sensors are directly connected to the CCU via an I/O-card, others to distributed control boards [76, 77].

- Complete decentralized architecture: the sensors and actuators are connected to local controllers which have cross communication capabilities. No CCU is present [131].

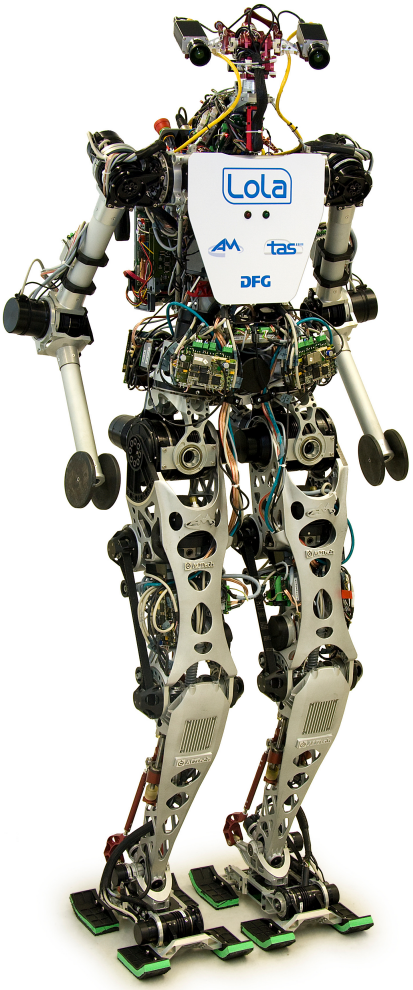


Figure 2.1: Photo of the humanoid robot Lola.

2.2 State of the Art

The humanoid bipedal robot JOHNNIE, realized at the Institute of Applied Mechanics of the Technischen Universität München (see [58, 86, 87]) between 1998 and 2003, has a centralized PC and a dedicated PCI/IO card to control the motors and read the sensor data. In the last years, most humanoid robotics projects have adopted a decentralized architecture.

ASIMO, developed by HONDA [151] uses PCs for vision and speech recognition. A processor is used for control and planning with support from DSP boards and PCI-I/O cards. HRP-3, from AIST and KAWADA Industries [77], integrates a central control for the upper body, head, arm and hands of the robot and a decentralized structure for controlling chest and legs. The communication bus used is CAN bus.[1]

ARMAR-III [13], by University of Karlsruhe is equipped with a PC and four PC/104 connected via Gigabit Ethernet. Moreover twelve distributed dedicated DSP/FPGA boards are used for motor control.

1 Communication Area Network, is a message-based protocol, originally designed for multiplex vehicle communication and developed by Robert Bosch GmbH.

The DLR-KUKA-LIGHTWEIGHT-ROBOT-III (DLR-LWR-III) [68], by DLR, has a decentralized motor control with dedicated control boards connected by a SERCOS-II bus[2]. Subsequently, on the basis of DLR-LWR-III, two humanoid robots have been developed. JUSTIN [105] is a human-like torso with a head and two DLR-LWR-III arms, four PCs and decentralized motion controllers. The research platform implements many different kinds of communication buses to resolve the complex heterogeneous architecture: SERCOS-II, EtherCAT[3], Gigabit Ethernet and CAN. The DLR's BIPED ROBOT [106] uses a modification of the DLR-LWR-III, a PC running VxWorks[4], a SERCOS-II bus and an RS485 bus[5] in order to read the data from the two force torque sensors.

The DLR HAND ARM SYSTEM [72], implements a heavily distributed system of dedicated FPGA[6]/CPLD[7] boards for motor control and sensor data processing. A real-time host running QNX[8] is connected via Spacewire[9] buses to FPGA boards. The latter communicate with the lowest control boards over a BiSS bus.[10]

ROBONAUT 2 [40], developed by NASA and General Motors, has a decentralized motor control system with dedicated FPGA-PowerPC[11] boards, connected by a custom bus interface to a central PowerPC CPU.

The experience gained performing experiments with JOHNNIE, have shown the limits of its electronic architecture. Having a centralized I/O card, all the sensors must be connected with a complicated and long wiring. The sensor feedback signals (many of them are analogue to be digitized on the I/O board) and motor command cables, cross the whole robot, increasing the possibility of errors due to noise and disturbances. In order to avoid these problems, the electronics architecture of LOLA has been designed as a decentralized network (see [49, 88]).

2.3 Mechatronic System Overview

The development of a humanoid robot is a complex task. Many aspects of different engineering domains (mechanics, electronics, IT, etc.) must be considered and optimized

2 Serial real-time Communication System [8]. It is a standardized open interface for the communication between industrial controls, motion devices and input output devices. The second version of the interface use fiber-optic as communication medium.

3 Ethernet for Control Automation Technology. It is an Ethernet-based fieldbus system invented by Beckhoff Automation. The protocol is standardized in IEC 61158.

4 VxWorks is a real-time operating system (RTOS) developed as proprietary software by Wind River.

5 ANSI/TIA/EIA-485 is a standard defining the electrical characteristics of transmitter and receivers for use in digital multipoint system.

6 Field Programmable Gate Arrays, it is an integrated circuit designed to be configured i. e., programmed by a customer or a designer after manufacturing.

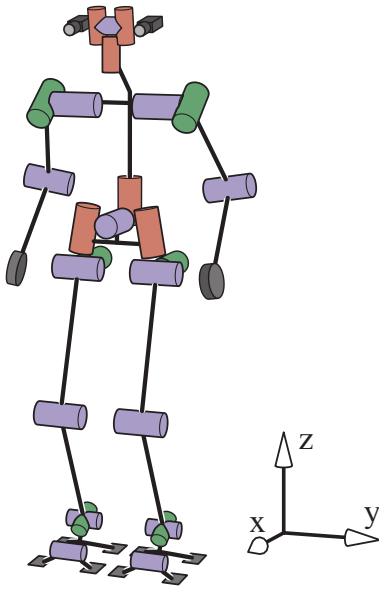
7 Complex Programmable Logic Device, it is a non volatile programmable logic device.

8 QNX is a commercial real-time Unix-like real-time operating system. Originally developed by the company Quantum Software Systems in the early 1980s (later QNX Software Systems), the company has been acquired by the company BlackBerry in 2010.

9 Spacewire is a spacecraft communication network based on the IEEE 1355.

10 Bidirectional Serial Synchronous, it is a real-time communication protocol. It enables a digital, serial communication between controller, sensor and actuator.

11 PowerPC (Performance Optimization With Enhanced RISC - Performance Computing) is a RISC instruction set architecture created in 1991 by an alliance of the companies Apple, IBM and Motorola.



Joint	# DoFs	Axis
Head	3	Pan
		Tilt
		Vergence
Shoulder	2	Flexion/extension
		Adduction/abduction
Elbow	1	Flexion/extension
Pelvis	2	Internal/external rotation
		Adduction/abduction
Hip	3	Internal/external rotation
		Adduction/abduction
		Flexion/extension
Knee	1	Flexion/extension
Ankle	2	Adduction/abduction
		Dorsiflexion/plantarflexion
Foot (toe)	1	Flexion/extension
Total	25	

Figure 2.2: Lola joints: schematic (left). List of actuated axis (right).

in order to obtain a system with satisfying performance. In the following subsections a brief description of the robot LOLA, considering different topics as the mechanical electrical and control design, is given.

2.3.1 Mechanics

The mechanical characteristic of a robot play a key part in the static and dynamic performance of the system. In Lohmeier [88] a complete and comprehensive description of the hardware development of a humanoid robot (with focus on LOLA) is given. Main overall goals of the project are the development of a human-like robot capable of walking at a speed of 5 km/h. The geometric proportions of the robot are based on human anatomy (Figure 2.1).

Therefore, LOLA has 25 electrically actuated DoFs (Figure 2.2, left), that try to reconstruct the structure of the human body. The legs have 7 DoFs each (3 in the hip, 1 in the knee, 2 in the ankle and 1 in the foot), the pelvis has 2 DoFs and each arm has 3 DoFs (2 in the shoulder and 1 in the elbow). The head has 3 actuated DoFs: pan, tilt and vergence angle (see [88, 89, 90]). For the mechanical design of the biped robot, the main, but not all, priority design goals have been:

- Minimum overall mass.
- High center of mass.
- Sufficient structural stiffness.
- High joint stiffness.
- Low moments of inertia of the leg links.

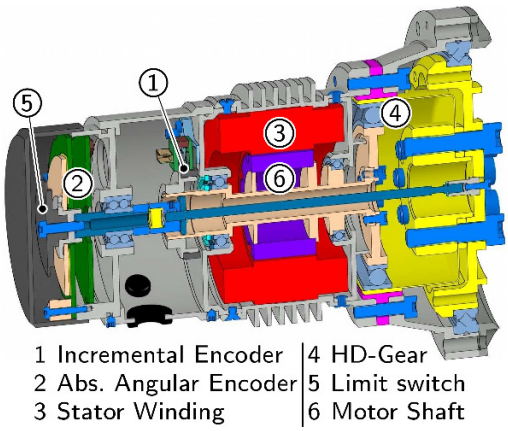


Figure 2.3: CAD section hip yaw joint from [48].

All of those contribute to improve the dynamic performance and acceleration capabilities of the robot.

2.3.2 Actuation

Another important aspect of a high dynamic biped robot is the type and characteristics of the actuation units. In order to provide a tailored solution to the actuation problem which incorporate characteristics such as *a*) compactness, *b*) simple integration in the mechanic structure, *c*) high power density, *d*) high acceleration capabilities and *e*) robustness, frameless high performance *Permanent Magnet Brushless Synchronous Motors* (PMSM) from *Parker Bayside* [14] in combination with a gear transmission are used in all joints of the robot.

All actuation units have structure similar to the one shown in Figure 2.3, but their size is adapted depending on the motor and the type of transmission used to satisfy the requirements of each link [90].

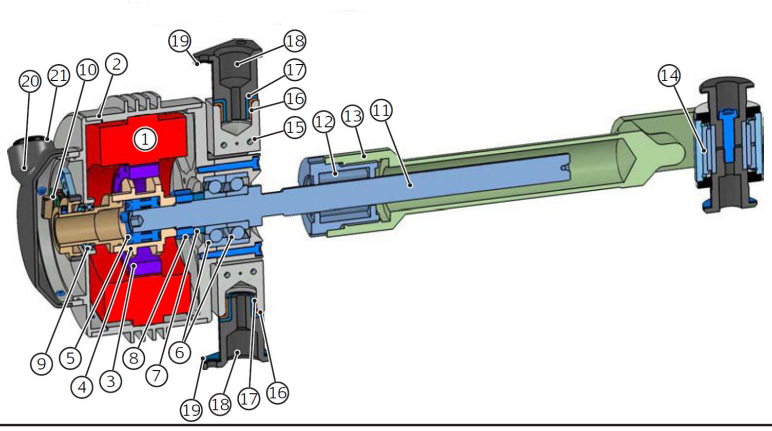
Except for the knee and ankle, all joints employ Harmonic Drive gears [4] (HD) as power transmission devices (see Figure 2.3). In case of the knee joint (see Figure 2.4), the actuation unit is composed of a PMSM and a screw-based linear transmission drive. The ankle joint employs a parallel mechanism composed of 2 PMSM, 2 synchronous belt and 2 screw-based linear transmission. The 3 DoFs of the robot's head are actuated with 3 PMSM motors, HD gears for the tilt and pan joints and a lever coupler mechanism for the vergence joint that allows the use of only one motor to move the 2 cameras of the robot.

2.3.3 Sensors

To reliably control a humanoid robot, different types of sensors are needed. They are used to capture the state of the system and deliver the information to stabilize the walking machine.

The sensor system of LOLA is composed of:

- Incremental position encoders.
- Absolute position encoders.



1 Stator	8 Locknut	15 Spider (thigh)
2 Motor housing	9 Roller screw bearing B	16 Plain bearing
3 Permanent magnet rotor	10 Incremental encoder	17 Steel raceway
4 Rotor receptacle	11 Roller screw shaft	18 Cantilever axis (thigh)
5 Cone clamping	12 Roller screw nut	19 Adjustment shim
6 Roller screw bearing A	13 Immersion tube	20 Protective cover
7 Bearing spacer	14 Universal joint (shank)	21 Cable grommet

Figure 2.4: CAD section knee joint from [88].

- Force torque sensors.
- Inertial measurement unit (IMU).
- High resolution cameras.
- Light barrier.
- Temperature sensors.

Except for the 3 joints on the head of the robot, two type of position sensors are used for each actuated DoF. An incremental encoder [7] is used to measure the motor position and supply the necessary information to the power electronics to control the PMSM motor. They are rotary magnetoresistive encoders which are characterized by a high resolution, robustness and a high dynamic response.

On the link side an absolute encoder delivers the position of the load. It is used as reference positioning and initialization devices and for the control of the joints. It is an Extended Channel Interpretation (ECI) devices and utilize an inductive measurement principle. On Lola two sizes of absolute encoders are used. Those mounted on the ankles and toes joints have a resolution of 16 bit while those on all other joints have a resolution of 17 bit. Both have a measurement accuracy of 0.1 °, a cutoff frequency 6 kHz and a latency for continuous sampling of position values of 5 μs.

The three motors on the head of the robot, while not having any link position sensors, have 3 hall sensors which are used for the motor's initialization.

Every joint also incorporates a limit switch to avoid problems of self-collision and cable wind-up.

An IMU [10] is mounted in the chest of the robot. This sensor unit measures the linear acceleration, angular velocities of the upper body of the robot and computes its orienta-

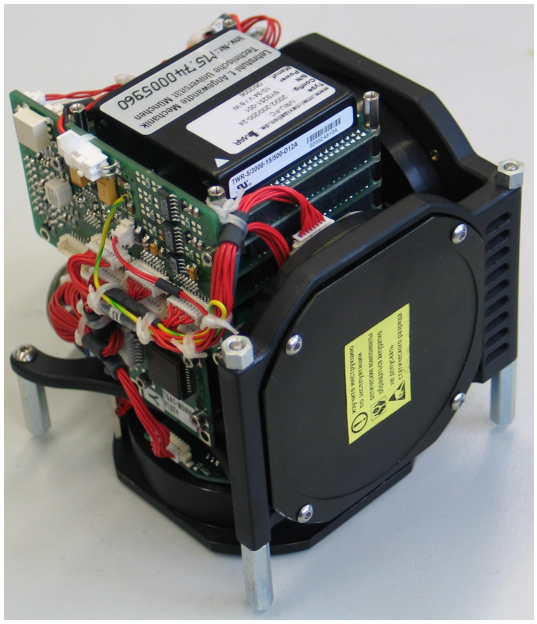


Figure 2.5: Photo of the IMU *iVRU-FC-C167*.

tion. These measurements are some of the most critical for the robustness and reliability of the robot stabilization while walking. The sensor, depicted in Figure 2.5, consists of three open-loop fiber-optic gyroscopes and three MEMS accelerometers. A sensor fusion algorithm, which compensates also for bias errors, is fully integrated into the sensor.

Fundamental measure for every walking robot are the ground reaction forces and moments. LOLA is equipped with two six-axes force torque sensors (FTS). A CAD section of the device is shown in Figure 2.6. The sensors are highly integrated in the feet of the robot and have been designed as supporting elements of the foot structure. This custom made sensor has been developed at the Institute of Applied Mechanics. In [88, 117] a complete and comprehensive description of the development, calibration and characterization of the sensor is presented. The sensor uses a monolithic transducer in the form of a Maltese cross with four shear beams. The force/torque sensing elements are 16 metal foil of strain gauges integrated into eight Half Single Wheatstone Bridges. The sensor integrates all necessary electronics for offset compensation, signal conditioning and digitizing electronics.

The vision system of the robot consists of two high resolution cameras [9] with a resolution of 5 MPixel. The cameras are connected via Gigabit Ethernet to an external server cluster.

Every AU is also equipped with a temperature sensor to monitor the temperature of the integrated motor.

2.3.4 Electronics

A reliable and high performance computing system is a key aspect of every mechatronic system. The elaboration of the system state, i. e., sensor reading, the control of the system,

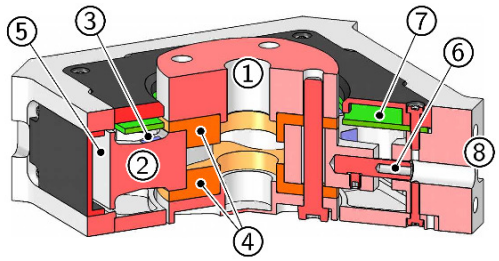


Figure 2.6: CAD section of the force torque sensor from [48].

the gait and trajectory generation and system safety, for the user and the system itself, are some of the most important functions that must be designed and implemented in the electronic system. For LOLA, a decentralized electronics topology has been developed and its schematic representation is depicted in Figure 2.7. In [88] a brief description of the system is presented, while in [49] more details of the system are described and some performance measures are shown.

The electronic system is composed of four main parts:

- Central Control Unit.
- Distributed Sensor Control Board (DSCB).
- Communication System.
- Vision Processing Server (VPS).

Every component have its specific purpose. The CCU is a Core 2 Duo Mobile, 2.33 GHz, running the QNX Neutrino real-time operating system. It is mounted on the back of the robot on an embedded Mini-ITX board and is connected via a communication bus to nine DSCBs. The CCU is also connected to a monitoring PC and the VPS through an Ethernet Switch. The connection with the external PC allows to monitor the state of the robot and to send basic commands to it, in case the video system is not in use. The commands that can be send are for example the direction and the speed with which the robot should walk, the parameters for the local controllers or to start/stop the sensor data logging. The CCU is responsible for the trajectory generation of every joint of the robot and the stabilization of the system dynamic. This task can be considered as the *High Level Control* of the machine, while the control of the joints runs on every local controller, i. e., the DSCBs. The latter represents the *Low Level Control* of the machine.

The DSCBs are custom-made embedded boards designed at the Institute of Applied Mechanics (see [49]). They are able to interface all the sensors used on LOLA and to control up to three motors each. A complete description of the DSCBs is given in Section 2.4.

The DSCB also integrates the power electronics driver for the PMSM motors. The servo controllers are the commercially available ELMO servo drivers [3]. They are compact, reliable and versatile servo controllers. Current, velocity, position and dual loop controller algorithms are available to the user as single or cascaded control loop. The ELMO module

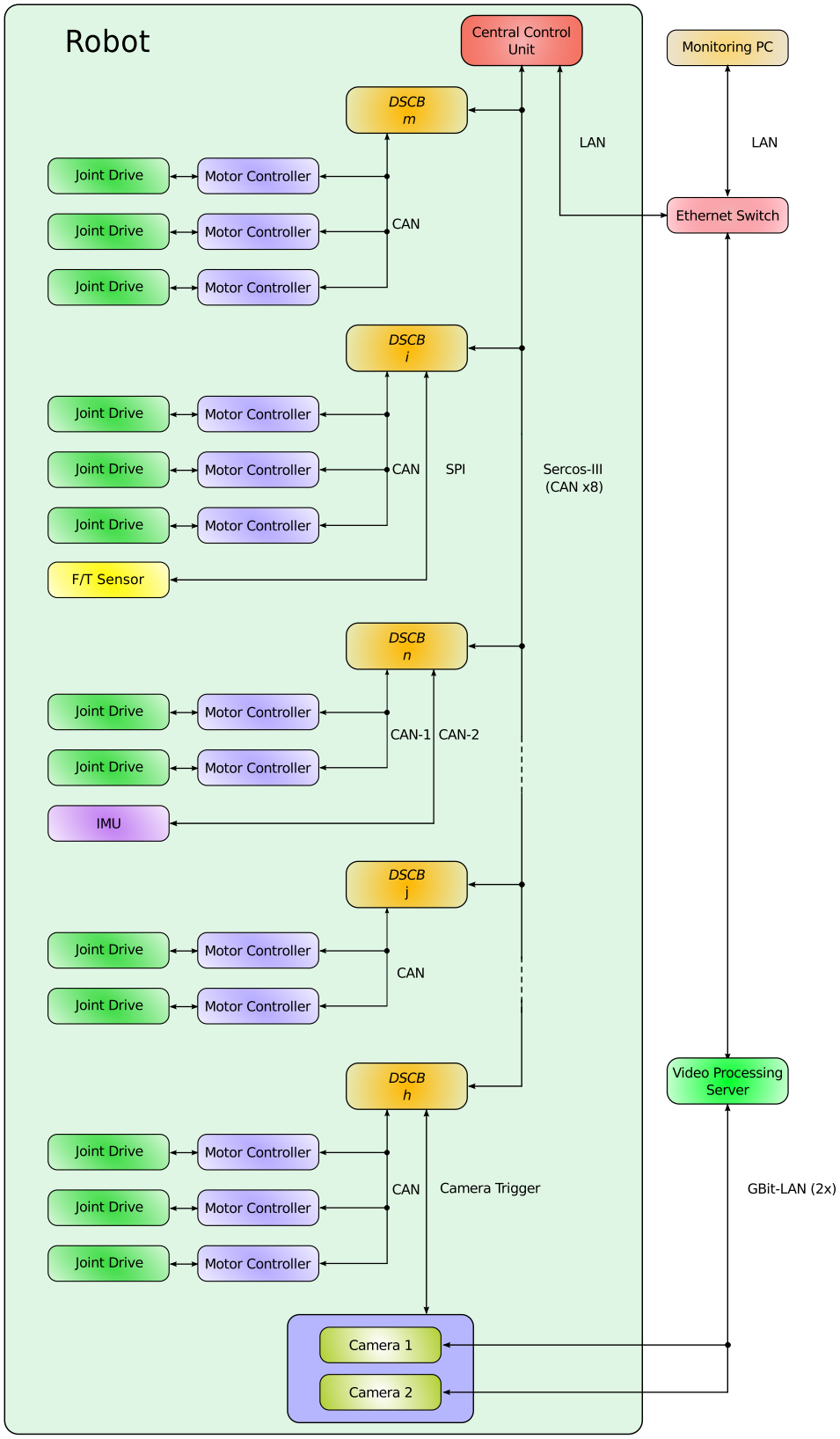


Figure 2.7: Electronics architecture of Lola. In the picture, the five DSCB variant are highlighted. For each of them a different combination of sensors and AUs are available.

Table 2.1: List of the data interfaces of Lola sensors and drives.

Sensor	Data Interface
Incremental Encoders	Quadrature Signals
Absolute Encoders	EnDat Protocol
Hall Sensors	3 Digital Signals
Force Torque Sensor	SPI
IMU	CAN
Temperature Sensor	Analog Signal
ELMO	CAN
CCU	Sercos-III

supports a wide range of position and velocity sensor configurations with digital as well as analogue outputs.

The communication between CCU and DSCB was initially implemented with 8 independent CAN buses which delivered sufficient performance and data throughput to let the robot walk at a moderate velocity. The revised implementation of the communication bus runs on a SERCOS-III bus [121],[12] which delivers better communication performance, higher data bandwidth and advanced features like synchronous and asynchronous communication channels.

Due to the large amount of data that need to be processed, the vision system requires an external computation system. The VPS is composed of three servers, each of which mounts a Dual Intel Xeon processors. Two of them are directly connected to the cameras of the vision system via a Gigabit-Ethernet connection. In order to let the robot walk autonomously, the VPS and CCU must exchange information and walking commands. The vision system needs some information about the current state of the robot to generate the walking commands in form of direction and speed. For more information about the vision system and autonomous walking of LOLA see [20].

In Table 2.1 the data interfaces that the DSCBs must support is listed, while Figure 2.7 shows the electronic architecture of LOLA highlighting the key components CCU, VPS, communication systems and the DSCB with different actuator and sensor configurations.

2.4 Distributed Sensor Control Boards

Having 25 fully actuated joints and 53 sensors with different communication interfaces, LOLA is a rather complex system. The AUs, sensors and power electronics devices are distributed on the robot body. All these components need to be supplied with power and connected to the CCU in order to exchange information, i. e., send the current sensor data and receive the new joints trajectories. This raises a problem for the wiring of the entire

12 SERCOS-III is the third generation of the Sercos Interface [8].

system. Cables increase the weight of the robot and, if not carefully done, the cabling over moving parts can cause major problems due to cable break or a possible reduction of the moving capacity of the machine. Therefore, reducing the number of cables that cross the robot to a minimum and designing a reasonable placement those problems can be reduced. Moreover, using long wires across the machine increases the possibility of failure due to electrical noise and disturbances, causing sensor transmission errors.

For these reasons a decentralized topology for the sensor control network of LOLA is applied. The requirements of this solution can be summarized as follows:

- The wiring across the robot must be minimized.
- All the units that *produce* information on the robot state (sensors), must be in the proximity of the device which *collects* this information.
- All sensors must be capable of internally performing the analogue to digital conversion of the measured signal.
- All sensors are capable to send their measurements in digital form.
- The only cables that cross the robot are for power supply (12 V for the electronics power generation and 80 V for the motor power) and the main communication bus.
- The power supply for the sensors and electronics, i. e., 5 V, 3.3 V and 1.2 V, are generated locally to reduce power losses due to long cables.
- The decentralized devices must be robust and compact in order to be integrated in the structure of the robot.

The key elements of the chosen decentralized topology are the DSCBs, which are a customized electronics system developed to satisfy the mentioned requirements. They are capable of interfacing all the sensors (Table 2.1), to interface the power drive units, i. e., the ELMO modules, to implement the communication with the CCU over the SERCOS-III bus and to supply the required power for the sensors and the servo controllers.

2.4.1 Hardware

The DSCB is a system of three stacked Printed Circuit Board (PCB), each of which has its specific function. Referring to Figure 2.8 they are identified as follows:

1. Control-Communication Board.
2. Power Electronics Support Board.
3. Incremental Encoder Module.

The Control-Communication board[13] is the core of the DSCB and its main components are: a Digital Signal Controller (DSC) MCF56F8367 from Freescale and a Spartan-3

13 The hardware development of the Control-Communication Board has been done by Dr.-Ing. Mathias Bachmayer.

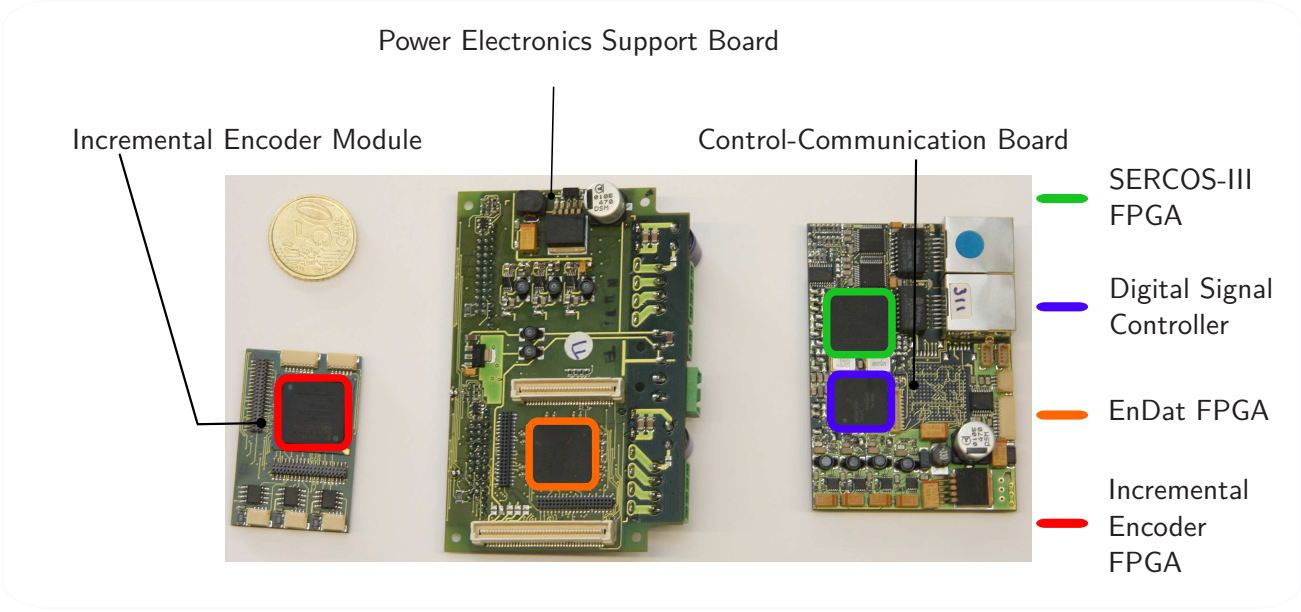


Figure 2.8: Distributed Sensor Control Board (DSCB): Incremental Encoder module, Power electronics Support Board and Control-Communication board.

xc3s200 FPGA from Xilinx. The FPGA implements the hardware stack for the communication with the CCU over the SERCOS-III bus as an IP-Core[14]. The latter has been developed by the company *AUTOMATA* [2].

The DSC is responsible for:

- Communicate with the CCU, implementing the SERCOS-III software communication stack[15].
- Collect the data from the connected sensors.
- Implement the motor control algorithms.
- Communicate with up to three ELMO modules.
- Implementation of safety features.

The DSC has two CAN ports. The first (CAN-1) is used for communication with the IMU and the second (CAN-2) is connected to the ELMO drives. Both ports can be seen in Figure 2.10. In the same figure the two Ethernet connectors for the Sercos communication are also highlighted.

The Power Electronics Support Board[16] has been designed to interface:

- up to three ELMO controllers over CAN,

14 In the electronic design an Intellectual Property Core is a reusable unit of logic, cell, or chip layout design. They are also addressed as *Hard Cores*. In the case of FPGA design an IP-Core, is written in a hardware description language, such as Verilog or VHDL, to add specific functionality to the logic design. They are also called *Soft Cores*

15 SERCOS-III software driver developed by *AUTOMATA*, the porting to MCF56F8367 and QNX has been realized at the Institute of Applied Mechanics of the Technischen Universität München.

16 The hardware development of the Power Electronics Support Board has been done by Georg Mayr

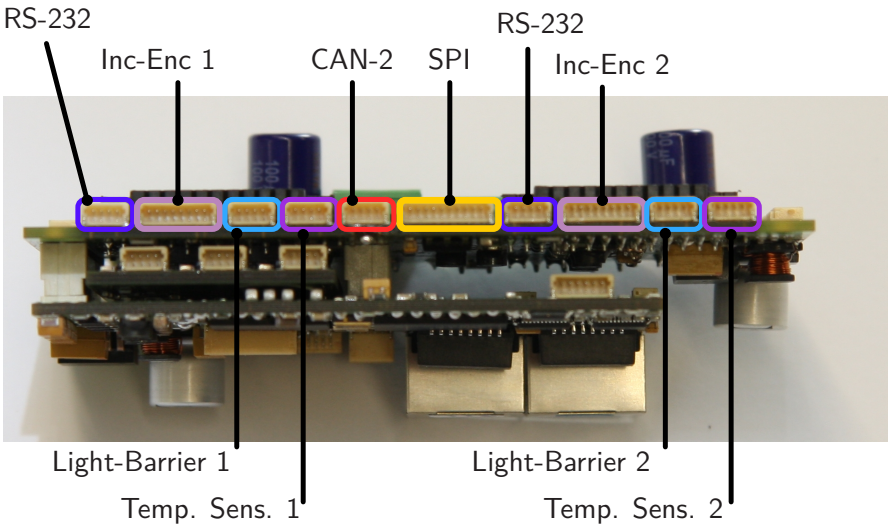


Figure 2.9: Distributed Sensor Control Board stack: front view.

- up to two Incremental Encoder,
- up to three EnDat Encoders,
- one Force Torque Sensor,
- up to two Light-Barriers,
- up to two Temperature Sensors.

Two out of three ELMO can be directly placed on the PCB as shown in Figure 2.11. The third ELMO is placed as close as possible to its DSCB and connected over the CAN connector on the same board.

The light-barriers and temperature sensor of the motors controlled by the ELMOs placed on the board are routed to the corresponding ELMO controller. For the third motor, the two sensors are directly wired to the ELMO drive.

On the board two Incremental Encoders can be connected. Their signals are routed to the respective ELMO drives and to the Incremental Encoder Module where a connector for the third encoder is also available. Three connectors for the link side absolute encoder are available. The EnDat[17] interface is implemented as IP-Core in the FPGA. In Figure 2.9 the communication connectors on the board are highlighted, while in Figure 2.10 the power connection can be seen.

Two RS-232 ports are used as programming and debugging interface for the ELMO modules. The SPI connector is used to interface one of the two force/torque sensors of the robot. It is connected to the SPI port of the DSC and implements a filter for impedance adaptation.

The Incremental Encoder Module[18] is an interface board able to read the position of up to three motors and to compute their velocity. The incremental encoder logic and the

17 The EnDat interface, by the company HEIDENHAIN, is a serial, digital, bidirectional communication interface for encoders.

18 The hardware development of the Incremental Encoder Module has been done by Georg Mayr. The system definition and IP-Core development have been done in this work.

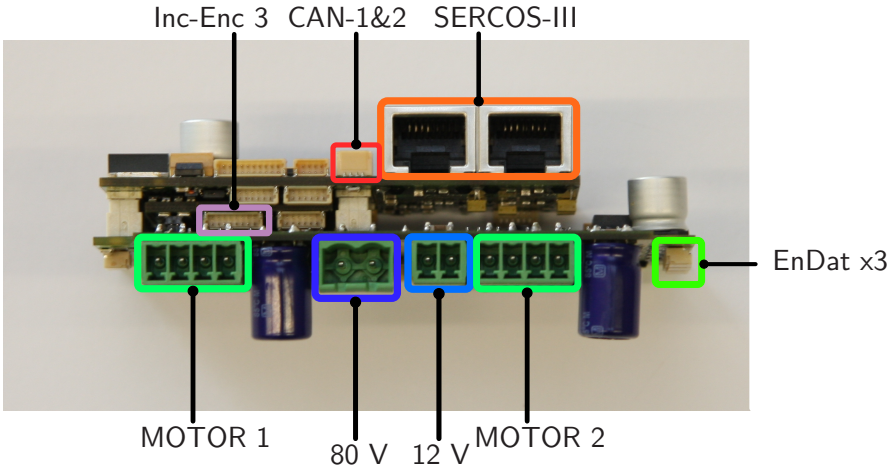


Figure 2.10: Distributed Sensor Control Board stack: back view.

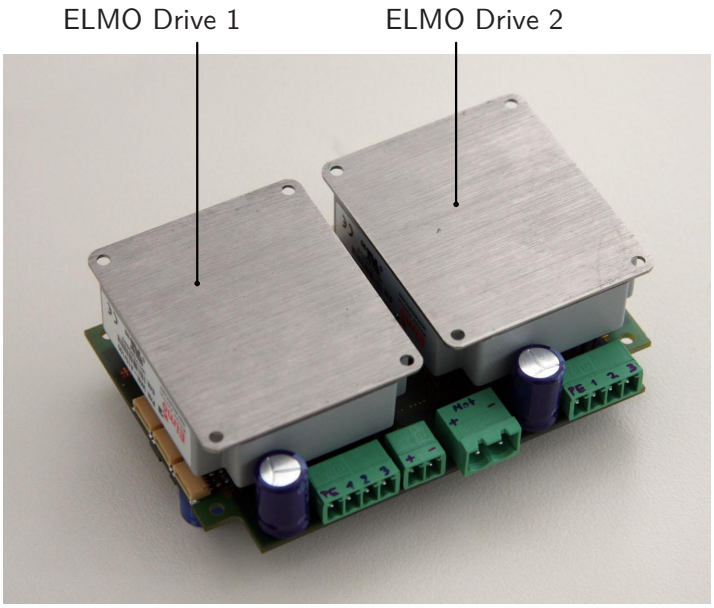


Figure 2.11: Distributed Sensor Control Board with ELMO servo drives: bottom view.

algorithm to compute the motor velocity are implemented as IP-Core in the Incremental Encoder FPGA. On the PCB three additional CAN ports for future development are also available.

On the head of the robot a DSCB is also used to control its three DoFs, read the sensor positions and generate the camera picture trigger signal. The motors have a lower power in comparison with the actuators used for the other joints. Therefore, they can be connected to less powerful ELMOs. The modules need a lower input power voltage, 24 V instead of

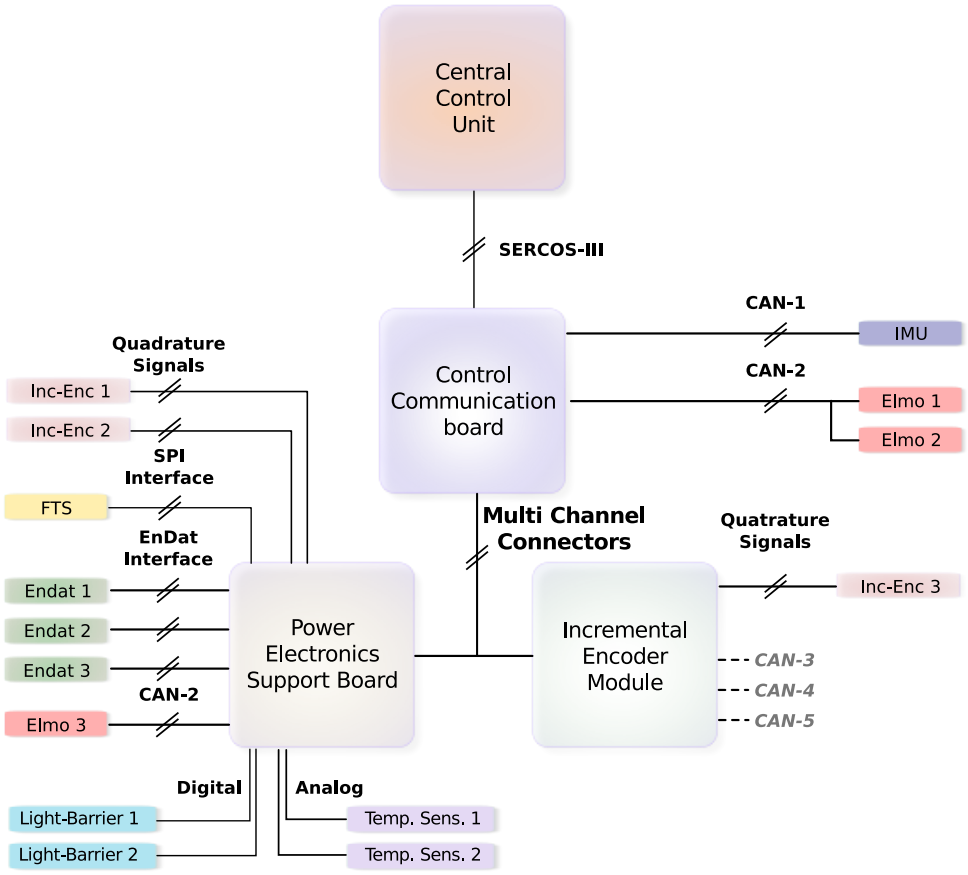


Figure 2.12: Distributed Sensor Control Board: diagram of the board's interfaces.

80 V. A dedicated board power module PCB[19] has been developed. It uses a DC/DC power converter and two connectors for the camera trigger signal.

In Table 2.2, Table 2.3 and Table 2.4 the main connections and chips of the Control-Communication Board, Power Electronics Support Board and Incremental Encoder Module are listed.

The three PCBs are connected through Multichannel Connectors (MC). All the peripherals and communication ports available for the DSC are routed to the MC. The MC are the main communication gateway between the DSC and the other components of the DSCB. The Inter Chip Communication[20] between the DSC and the three FPGAs is realized over the DSC's External Memory Interface (EMI). Each FPGA has its own address space which enables one chip at a time using a Chip Select signal. The DSC is the device that controls the communication on the EMI.

Figure 2.12 shows a schematic view of all the communication ports available on the DSCB and to which board they are connected.

19 Hardware development by Georg Mayr.

20 As Inter Chip Communication is intended the method used to exchange data between two or more chips which are placed on the same PCB or on different boards but connected via a common connector.

Table 2.2: Distributed Sensor Control Board: Control-Communication board features and components.

Control-Communication board		
Digital Signal Controller	DSC-MCF56F8367	clock 60 MHz
Sercos FPGA	FPGA xc3s200	clock 32 MHz
Available Connectors	Ethernet Ports	x2
	CAN	x2
	SPI	x3
	Rs-232	x2
	PWM outputs	x12
	Incremental Encoder	x2
	ADC 12bit	x4
	EMI Data Bus	16 bit
	EMI Address Bus	24 bit
	JTAG	x2 (1x DSC, 1x FPGA)

Table 2.3: Distributed Sensor Control Board: Power Electronics Support Board features and components.

Power Electronics Support Board		
EnDat FPGA	FPGA xc3s400	clock 32 MHz
Servo Drive	ELMO	x2
Available Connectors	EnDat Ports	x3
	CAN	x1
	SPI	x3
	Rs-232	x2
	Light-Barrier	x2
	Temperature sensor	x2
	Data Bus	16 bit
	Address Bus	24 bit
	JTAG	x1 (FPGA)

2.4.2 Communication Protocols

In the previous section, the structure and the components of the actuation, sensor and control network of Lola is described. All these devices use different types of communication protocols.

In this section, the relevant characteristics of the different interfaces and protocols are

Table 2.4: Distributed Sensor Control Board: Incremental Encoder Module features and components

Incremental Encoder Module		
Incremental Encoder FPGA	FPGA xc3ANs400	clock 32 MHz
Available Connectors	Incremental Encoder	x1
	CAN	x3
	Data Bus	16 bit
	Address Bus	24 bit
	JTAG	x1 (FPGA)

presented. Moreover, it is explained how they are implemented and integrated in LOLA using the DSBCs as the communication and data exchange nodes of the robotic system.

SPI

The SPI[21] interface is used to transfer data between the FTS and the DSC. Introduced by Motorola the interface has then become a *de facto* standard. It is a one to many communication protocols where one *master* controls the data exchange with at least one *slave*. Referring to Figure 2.13, where a system with one Master and three slaves is depicted, the SPI bus defines four logic signals:

- SCLK : Serial Clock.
- MOSI : Master Output, Slave Input.
- MISO : Master Input, Slave Output.
- SS : Slave Select.

SCLK is the synchronization clock that is generated by the Master. The MOSI signal is the master's output while the MISO is the output of the slave. The SS-signal is used to identify the slave for the next communication. The first three signals are shared between every slave on the bus, while every slave has its dedicated SS-signal.

The communication is full duplex and the master always initiates a transmission operation setting the SS-signal of the desired slave and starting a clock sequence on SCLK. During every clock cycle one bit is sent from the master to the slave on the MOSI and one bit in the opposite direction over the MISO signal. The duration of the communication transfer depends on the amount of data that is sent. It can be 8 bit, 12 bit or 16 bit depending on the slave. To stop the communication the master stops the clock signal and resets the SS-signal.

As an example of a possible SPI bus configuration in Figure 2.13 the actual topology of LOLA FTS sensor is shown. The DSC controls three devices using the same bus:

- The ADC measures the voltage pf each Wheatstone Bridges of the sensor and send this values to the DSC.

21 Serial Peripheral Interface.

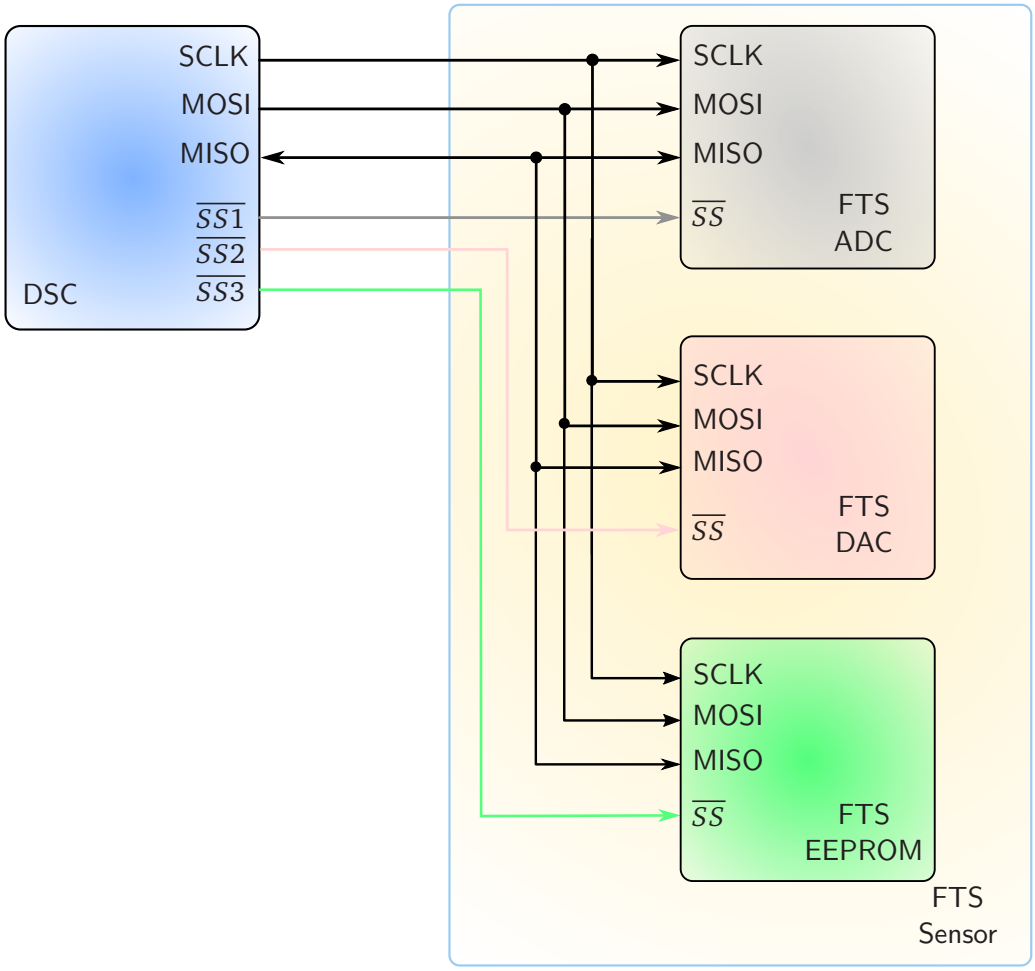


Figure 2.13: Diagram of the SPI communication between DSC and FTS.

- The DAC is used during the sensor initialization to compensate for offsets on each Half Single Wheatstone Bridge.
- The EEPROM stores the calibration matrix of the sensor which are used to calculate the force and torque information from the deformation ADC values.

To reduce the signal reflections and consequently possible transmission errors, each wire of the interface has been terminated using 120 Ω resistors. The frequency of the SCLK clock is set to 3.75 MHz. The SPI connection cable also supplies the sensor power voltage of 5 V and reference 0 V.

EnDat

The EnDat Interface is a differential bidirectional serial interface for encoders developed by HEIDENHAIN [5]. The detailed description of the interface is given in [67]. It is based on the RS-435 standard and can transmit position data from the encoder as well as send, update and store encoder parameters. The data is transmitted synchronously with the clock signal and requires only four signal lines for transmission. In Table 2.5 the signals used for this interface are listed.

Table 2.5: Interface signal between DSC and EnDat FPGA.

Bus	Number of bits	Signal Name	Purpose
Data Bus	16	$D0$-$D15$	Data Transmission
Address Bus	6	$A0$-$A5$	Component Address
Control Signals	8	M16	Select length of the data bus (8 or 16 bit)
		$\overline{Reset}$	FPGA Hardware reset
		$\overline{Int}$	EnDat Interrupt Output (Not Used)
		$\overline{Int6}$	EnDat Interrupt Input (Not Used)
		$\overline{Ready}$	Data ready (Not Used)
		$\overline{Strobe}$	Strobe Signal (Not Used)
		$\overline{RD}$	Read Request
		$\overline{WR}$	Write Request
Chip Select	1	$\overline{CS}$	Chip Select

The use of differential signals allows a clock frequency of up to 8 MHz[22] with high robustness against disturbances and noises. The transmission clock used for the EnDat encoders in Lola is set to 4 MHz in continuous send mode. Using this option the position of the encoder is constantly sent via the EnDat bus. The correctness of the transmission is improved using a Cyclic Redundancy Check (CRC).

A subsequent electronic component is required to implement, on one side, the communication with the encoder over the EnDat protocol and, on the other side, the communication with the DSC. The subsequent electronics is implemented as an IP-Core which can interface one encoder. The interface between the subsequent electronics and the application processor (the DSC in the case of Lola) is described in [96].

To reduce the number of chips and, consequently, the space required on the Power Electronics Support Board, a new FPGA program has been designed. The new FPGA core implements the logic to communicate with three encoders as three independent EnDat IP-Cores and a unified interface to exchange data with the DSC over the EMI interface. In Figure 2.14 a block diagram of the communication interface between the DSC and the EnDat FPGA is given. Every core shares the same Data Bus and Address Bus. Each of them is mapped with a specific address space in the DSC and it is selected asserting the Chip Select signal ($\overline{CSx}$). Figure 2.14 shows a diagram of the data interface between the DSC and the EnDat FPGA. In Table 2.5 and Table 2.6 the function of all the signals of the interface is briefly explained.

The EnDat encoders used are of type EnDat 2.1. The protocol provides some basic commands to communicate with the encoders:

1. Encoder transmit position values.
2. Encoder set read parameters:
    - Selection of the memory area.

22 For special applications the clock frequency can also be set to 16 MHz.

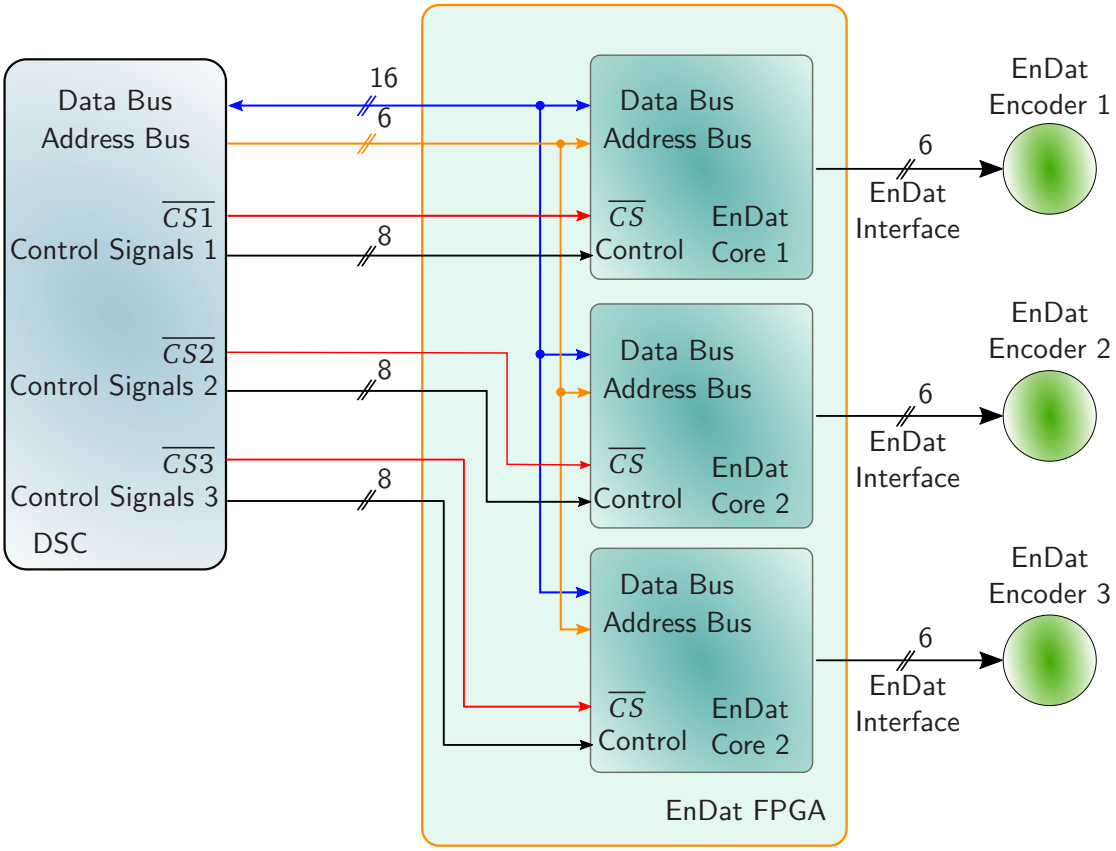


Figure 2.14: Interface between DSC and EnDat FPGA.

Table 2.6: Interface signal between EnDat FPGA and EnDat Encoder.

Bus	Number of bits	Signal Name	Purpose
EnDat Interface	6	Clock	Clock Signal (positive)
		$\overline{Clock}$	Clock Signal (negative)
		Data	Data (positive)
		$\overline{Data}$	Data (negative)
		Vcc	5 V (Encoder Power Supply)
		GND	0 V (Encoder Power Supply)

- Encoder receive parameters.
- Encoder transmit parameters.

3. Encoder receive reset.
4. Encoder test:
   - Encoder transmit test values.

- Encoder receive test commands.

The set of commands permits to setup the encoder, initialize it and receive the position information. Each command is described in detail in [67].

The task of the EnDat FPGA is to supply a standard communication interface between the application and the encoder. The IP-Core provides the following registers:

- Send register
- Receive register 1
- Receive register 2
- Receive register 3
- Configuration register 1
- Configuration register 2
- Configuration register 3
- Status register
- Interrupt mask
- Test register 1
- Test register 2

Every register has its specific purpose. To send a command to the encoder, it must be written into the *Send register*. The answer of the encoder will is stored in the *Receive register 1*. The *Status register* deliver information about the state of the encoder. The three *Configuration registers* are used to set the main communication parameters and option to configure the communication with the encoder. The *Interrupt register* enables or disables all the interrupt that can be triggered by the FPGA. The *Test registers* are used for test purposes.

A complete explanation of how the registers works and the meaning of every one of their bit is described in [96].

CAN

The Controller Area Network (CAN) bus is a message based protocol. It was originally developed by Robert Bosch GmbH for automotive applications and the latest version of the protocol is CAN 2.0. The protocol has been standardized in 1993 as ISO 11898. Nowadays the protocol is used in a large variety of applications from industrial automation to entertainment.

CAN is a *Multi-Master* serial bus and the connected devices are called nodes. The nodes are connected via two wires and use a differential standard. The first and the last nodes of the bus must be terminated with a 120 Ω resistor to avoid signal reflections. The clock is embedded in the data transmission and the nodes are able to autonomously lock themselves to the transmitted data message. The transmission rate varies from 40 kbps up to 1 Mbps.

The nodes are identified by a Message Identifier (Msg-ID), which can be 11 bit if the device supports CAN 2.0 part A [28] or 29 bit if the devices supports CAN 2.0 part B [29]. Every node can start a communication with each other nodes. If two nodes contemporary start to transmit, a communication collision happens. In this case the protocol implements a method of *auto arbitration* based on the Message Identifier: the node that sends a message with lower Msg-ID has higher priority and can proceed with the transmission.

Table 2.7: CAN Frame: fields description (CAN 2.0 Part A).

Field name	Length (bits)	Purpose
Start-of-frame	1	Start of frame transmission
Identifier	11	Message Identifier
Remote transmission request (RTR)	1	0
Identifier extension bit (IDE)	1	0 for 11 bit Message-ID; 1 for 29 bit Message-ID.
Reserved bit	1	Reserved bit (Set to 0)
Data length code (DLC)	4	Number of bytes of data (0-8 Bytes)
Data field	0-64 (0-8 Bytes)	Data to be transmitted
CRC	15	Cyclic redundancy check
CRC delimiter	1	Set to 1
ACK slot	1	Transmitter sends 1, receiver can assert a 0
ACK delimiter	1	Set to 1
End-of-frame (EOF)	7	Set to 1

The CAN standard has four frame types:

1. Data frame: containing node data for transmission.
2. Remote frame: to request the transmission of a specific identifier.
3. Error frame: transmitted by any node which detects an error.
4. Overload frame: inject a delay between data and remote frame.

A data frame is composed of different fields. Each of these fields has a specific purpose defined in the ISO standard and are listed in Table 2.7.

The dimension of the messages varies between 44 bit and 108 bit depending on the quantity of data that is sent. The number of Bytes which can be sent in a message varies between 0 Byte and 8 Byte.

In Figure 2.15 the configuration of the CAN bus to communicate with the ELMOs and the IMU is shown. The DSC has two independent CAN ports (CAN-1 and CAN-2). The first one is used for a point to point communication with the IMU. The latter is shared between up to three ELMOs.

The IMU also has an RS-232 interface. It must be used to initially setup the device and enable the CAN Interface communication. Once it has been configured, the CAN interface provides 14 memory slots for the sensor data output available for the device. Each of them is identified by an offset in the Message-ID. For Lola the IMU is configured to automatically send at a rate of 200 Hz the following data:

- 3 angular positions.

- 3 angular velocities.
- 3 angular accelerations.
- Information about the sensor's state.

The format of the data is a 16 bit Integer, thus four CAN messages are used to send all 10 values. The DSC periodically controls if new IMU data has been send. In this way, the communication over the CAN bus is optimized reducing the traffic to a minimum and receiving all the needed information.

The ELMO modules use the CANopen[23] protocol for communication. CANopen is a high level application layer protocol. It has been developed as a standard layer for embedded applications in automation. The standard describes the communication mechanisms (communication profile) and the functionalities of communicating devices (device profile). Every CANopen device have a standardized device description in the form of an *object-dictionary*. In the object-dictionary the features, data type, parameters functions etc. of the device are available over a Client-Server protocol.

The standard provides the following type of communication message, listed in order of priority:

- Network management (NMT): these messages are used to issue state machine change commands, e. g., to start and stop the devices connected on the network, to detect device bootups or error conditions.
- Synchronization Object (SYNC): a synchronization message is used to synchronize all the devices in the network and to trigger the execution of synchronous tasks.
- Time-Stamp Object (TimeStamp): usually, the Time-Stamp object represents an absolute time after midnight and the number of days since January 1, 1984. It is expressed in milliseconds.
- Emergency Object (EMCY): emergency messages are sent if a device encounters an internal fatal error situation and are transmitted from the concerned application device with high priority.
- Process Data Object (PDO): used for process real time data. The PDOs can be synchronous or asynchronous. The first are sent in responds to a SYNC message, while the seconds are triggered by a specific event or in response to a specific object of the dictionary.
- Service Data Object (SDO): used for setting and getting values from the object dictionary of a device. An SDO message exchange always follows the sequence: *a*) one device sends an SDO request to another device in the network, *b*) the second device sends one or more SDO message in response to the request.

The description of the implementation of the CANopen protocol for the ELMO modules is defined in [44]. The available parameters in the object-dictionary and implemented functionalities are defined in [42].

23 Version 4.0 of CANopen [27], published by the CAN in Automation Organization (CiA) in 1999, has become the European Standard EN 50325-4.

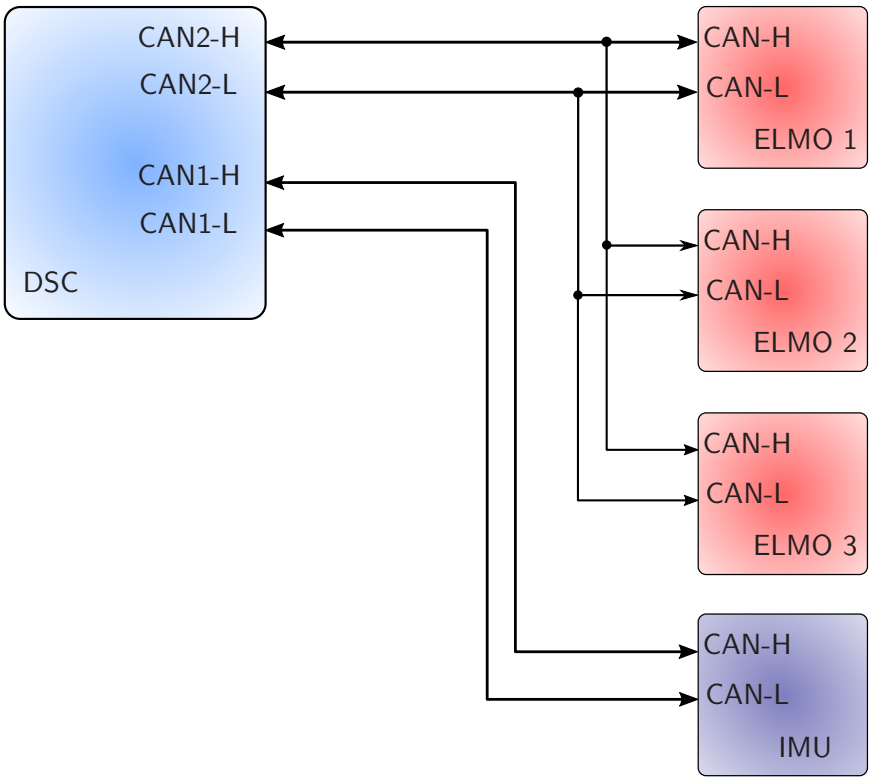


Figure 2.15: CAN Interface between DSC ELMO and IMU.

Sercos-III

SERCOS-III is a real-time Ethernet-based communication protocol developed for industrial automation. The specifications of the SERCOS-III interface are described in [121]. SERCOS-III supports the Fast-Ethernet standards[24], therefore the communication bandwidth is 100 $^{\text{Mbit}}/_{\text{s}}$. The bus can have either ring topology or line topology[25]. It is a synchronized Master-Slave protocol, where the communication cycle is triggered by the master.

The cycle time depends on the amount of data to be transferred and on the number of slaves connected to the bus. The minimum value is 31.25 μs with 2 slaves and 4 Byte of exchanged data.

SERCOS-III offers two different communication channels:

- Real-Time channel (RTC), used for synchronous and asynchronous communication in real time.
- Non Real-Time channel (NRT), used to transfer other types of data and to enable normal Ethernet connectivity.

Within one communication cycle two types of telegrams are sent: at least one Master Data Telegram (MDT) and at least one Acknowledge Telegram (AT). The first contains, among

24 IEEE 802.3 and ISO/IEC 8802-3 100Base-TX or 100Base-FX.
25 Ethernet itself has an inhered ring topology

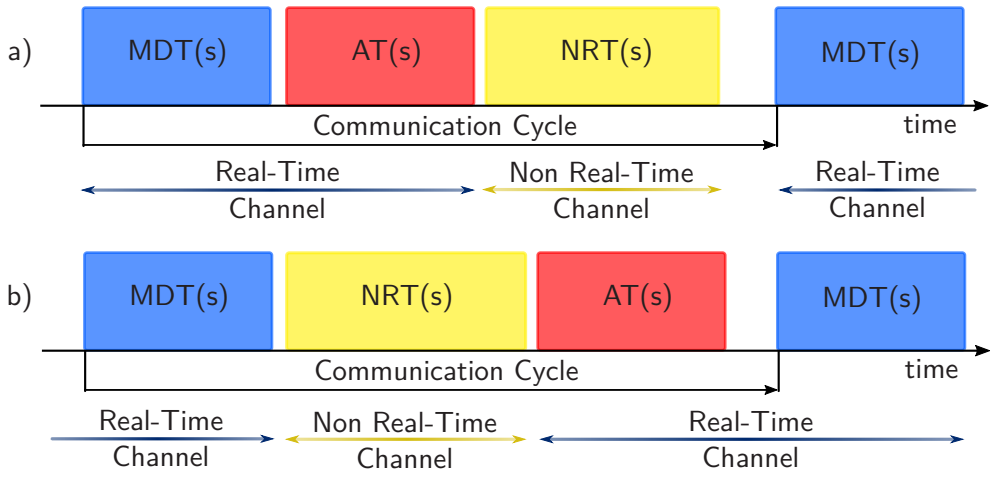


Figure 2.16: Sercos-III Communication Cycle.

other data, commands and new desired values for every slave. The second transports, among others, feedback sensor data and the status of the slave devices. The number of MDTs and ATs that are sent within a communication cycle, depends on the number of slaves connected to the network and can be up to four each. The rest of the communication cycle time is set free for other types of Ethernet telegrams, e. g., TCP-IP, UDP or Cross Communication (CC) telegrams between SERCOS-III devices. The arrangement of the telegrams transmission can be configured as shown in Figure 2.16. Configuration a) is the standard, while configuration b) is typically used if the slaves require a long time to get the sensor data. Both MDT and AT are sent by the master. The MDT contains information from the master to the slaves. The AT is sent as an empty telegram that is filled with the data by each slave in a predefined slot and sent back to the master.

In MDT and AT telegrams Real-Time Data (RTA) and the Service Channel (SVC) are embedded. The RTA is the data which is exchanged every cycle between master and slave. The SVC is used to send asynchronous data. Considering a servo system, an example of RTA are the new desired position set points for the slaves and the measured position values. The first are sent in the MDT and the latter in the AT. The SVC can be used to trigger a command or to send (or request) data that does not need to be sent every cycle.

The SERCOS-III telegrams are always broadcast messages, i. e., they are received from every device in the network. In the data fields of the MDT, and AT as well, each slave have its own well defined reading (for the MDT) and writing (for the AT) space. The slaves still have a specific address to be identified from the master. During the initialization of the bus each slave receives a SERCOS-III address, a number between 1 and 511. The slaves must also support Ethernet MAC addressing. While not essential for the bus operations a MAC address is used for NRT channel communication.

The protocol defines four Communication Phases (CP1 to CP4) which are necessary to reach the real-time communication stage. CP1 is a transition phase to CP2 where the SVC channel is available to the master to configure the slaves. CP3 is also a transition phase to CP4, the actual real-time stage of the bus. Once the communication bus reaches CP4, the real-time channel is available and the system is ready to start the control application.

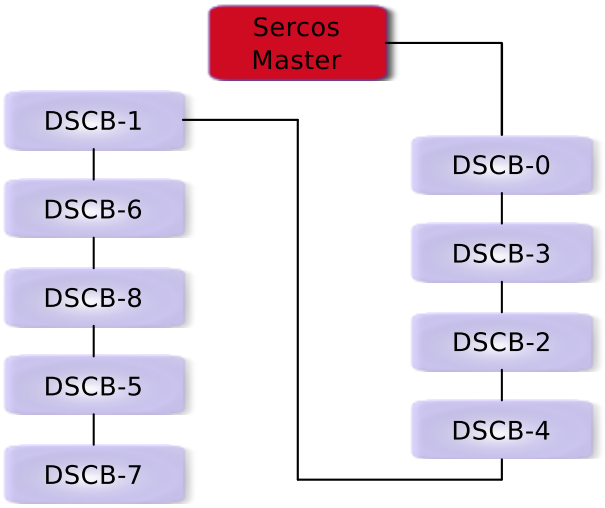


Figure 2.17: Sercos-III topology on Lola.

Every SERCOS-III slave has an internal data base. In the data base all the characteristic, capability and parameters of the slave are listed and organized in Identification Numbers (IDN). A detailed description of the IDNs is available in [123]. The data base is the core information structure of the SERCOS-III protocol. All exchanged data between master and slave are stored in it.

The protocol defines a number of standard parameters which must be available in every SERCOS-III slave device. At application level, the standard also describes a generic definitions and grouping of the IDNs defining the so called Generic Device Profile and Function Specific Profiles [122]. The goal of this classification is to define a slave device depending on its function. For example, specific IDNs are available in an actuator slave while an input output slave device must have other IDNs.

Nevertheless many IDNs are available for a custom definition of a slave. Those are useful to define extended characteristic or capability of a standard slave or to profile a non-standard device.

In Figure 2.17 the line topology of the LOLA SERCOS-III bus is shown. The SERCOS-III Master is a PCI[26] board developed by *AUTOMATA*. The DSCBs are connected in cascade and their position on the robot is listed in Table 2.8, under the column Position. In the same table the number of AUs and sensors for each DSCB is listed, as well as the MAC address.

Since the DSCBs of LOLA manage many custom components, it is convenient to define custom profiles for every configuration variant listed in Table 2.8. Specific IDNs have been introduced to characterize the DSCB's non-standard sensors like the FTS, the IMU and the cameras. Being an interface mainly developed for automation purposes, the SERCOS-III interface has specific IDNs dedicated to control the behavior of standard slaves and to receive information on their status, i. e., the *Drive Control Drive Status* IDNs. Nevertheless, to control the behavior of the DSCBs and ATs and to improve the flexibility and the debug

26 Peripheral Component Interconnect is a local computer bus for attaching hardware devices in a general purpose computer.

Table 2.8: List of the DSCB and their AUs and sensors in the Sercos-III network.

Slave	MAC Address (MAC in ASCII)	Position	Number of AUs	Available Sensors	Software Variant
DSCB-0	76:79:76:64:83:48 (L :O :L :A :S :0)	Right Shoulder	3	3 Incremental Encoders 3 Absolute Encoders	*m*
DSCB-1	76:79:76:64:83:49 (L :O :L :A :S :1)	Left Shoulder	3	3 Incremental Encoders 3 Absolute Encoders	*m*
DSCB-2	76:79:76:64:83:50 (L :O :L :A :S :2)	Front Right Pelvis	3	3 Incremental Encoders 3 Absolute Encoders	*m*
DSCB-3	76:79:76:64:83:51 (L :O :L :A :S :3)	Back Right Pelvis	2	2 Incremental Encoders 2 Absolute Encoders	*j*
DSCB-4	76:79:76:64:83:52 (L :O :L :A :S :4)	Right Knee	3	3 Incremental Encoders 3 Absolute Encoders 1 FTS	*i*
DSCB-5	76:79:76:64:83:53 (L :O :L :A :S :5)	Front Right Pelvis	3	3 Incremental Encoders 3 Absolute Encoders	*m*
DSCB-6	76:79:76:64:83:54 (L :O :L :A :S :6)	Back Left Pelvis	2	2 Incremental Encoders 2 Absolute Encoders IMU	
DSCB-7	76:79:76:64:83:55 (L :O :L :A :S :7)	Left Knee	3	3 Incremental Encoders 3 Absolute Encoders 1 FTS	*i*
DSCB-8	76:79:76:64:83:56 (L :O :L :A :S :8)	Head	3	3 Incremental Encoders 2 Cameras	*h*

capability of the robotic system, they have been extended with specific commands and variables. In Appendix B are listed all the standard and LOLA custom IDNs used to control the robot.

2.4.3 Software Functions

The DSCB must perform two basic, but important tasks which are: *a*) control the behavior of the joints of the robot and *b*) deliver the sensor data to the CCU. Besides them, they must monitor the state of the sensors, the state of the AUs, inform the CCU in case of errors and safely stop the joint control and the AUs. The software of the DSCBs is implemented as "Bare Metal" due to the limited memory resources available on the DSC.

The DSCB boards introduce an abstraction layer to the hardware of the robot taking care of all hardware-specific configuration tasks required by the peripherals of the robot, i. e., the sensors and AUs. It allows the CCU to access the robot hardware trough a simplified and

standardized interface making it transparent for the user. The user defines only the data that must be exchanged every cycle, while the DSCBs makes them available, carry all the necessary operation of signal processing and the required control actions for the robot joint.

As show in Table 2.8, the nine DSCBs used on LOLA must deal with different sensor and actuators combination. All boards have the same hardware interfaces and capabilities making them easily interchangeable. Nevertheless, 5 variants, or profiles from the SERCOS-III point of view, of the boards are defined via software. Depending on the number of AUs that must be controlled and depending on the type of sensors that must be interfaced the following variants are defined (see Figure 2.7):

1. Variant *m*: 3 motor side and 3 link side position sensors, 3 servo drivers (DSCB-0, DSCB-1, DSCB-2, DSCB-5).
2. Variant *j*: 2 motor side and 2 link side position sensors, 2 servo drivers (DSCB-3).
3. Variant *i*: 3 motor side and 3 link side position sensors, 1 FTS sensor, 3 servo drivers (DSCB-4, DSCB-7).
4. Variant *n*: 2 motor side and 2 link side position sensors, 1 IMU sensor, 2 servo drivers (DSCB-6).
5. Variant *h*: 3 motor side position sensors, 2 cameras, 3 servo drivers (DSCB-8).

Due to the many configuration options it is necessary to define a simple method to develop and configure the required functionality of the software for each DSCB variant. The DSCB software must satisfy two development criteria: *a*) *modularity*: the DSCB code is organized in functional modules which synergically work together enabling the execution of complicated tasks. A set of API[27] has been programmed to interface all the hardware components of the robot and to exchange data between the DSCBs and the CCU as well as between other software modules; *b*) *configurability*: each variant of the DSCB's software is defined at compile time as a mandatory configuration option. Joint control parameters and some sensor configuration options are available at run time. The latter are set during the robot initialization (for example the availability of current or velocity measurement for every joint) or while an experiment is performed (for example control gains).

The API is organized in the following function groups (FG[28]):

- DSC Configuration and System management.
- Communication.
- Sensor driver.
- Actuator driver.
- Joint control.
- Spindle kinematics.

27 Application Program Interface.

28 A list of all the file in the FGs are in Appendix C.

- Safety and error management.
- Utility and debug.

The DSC configuration and System management FG supplies functionalities to setup the DSC's hardware registers and prepare the controller to communicate and configure all the external components, i. e., AUs and sensors. The FG also defines and implements the FSM functionalities necessary for the system to work properly.

The communication FG is the core of the entire DSCB's software. This FG consists of low level drives which implement the basic hardware communication (drivers for CAN, SPI and EMI interfaces) and of high level communication drivers which are founded on the former. The latter supply a more convenient access to AUs and sensors of the robot delivering extensive information like error data and status data. The ELMO and IMU drivers both use the CAN API, the EnDat, the incremental encoders and the Sercos-III drivers are based on the EMI API and the FTS uses the SPI API.

The Sensor FG defines the high level function to read and elaborate the sensor data. It is based on the Communication FG, as is the Actuator driver FG which defines the functions to control the ELMOs.

The Joint control API defines all the control related function available to the user, i. e., position velocity and current control routines.

The Spindle kinematics API is used by the Joint Control and Sensor FGs to convert the sensor data from the joints actuated by a spindle, i. e., knees, ankles joints and also the head vergence joint, and the control data available for the relative AUs.

The Safety and error management FG supplies function, macros and error definitions to facilitate the error identification and manage a safe shutdown of the robotic system.

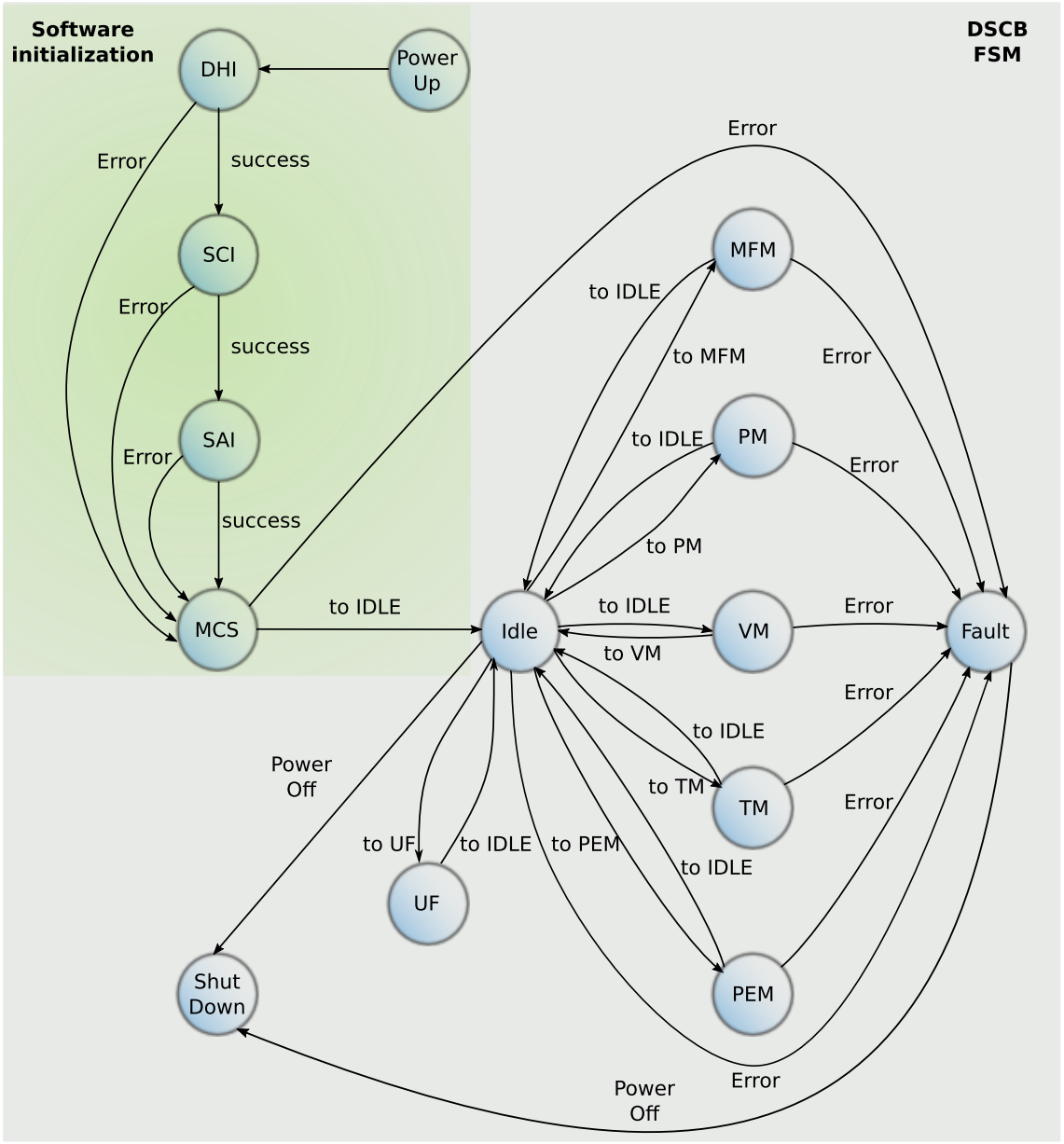
The Utility and debug API provides the DSC system with useful functions to debug and measure system performance. These functions provide timing measurement, interrupt request, file backup, among others.

The core of every DSCB unit implements a Finite State Machine (FSM) which controls the DSCBs state change and implements system commands. The different state and their transitions are depicted in Figure 2.18. After an initialization phase, the DSCB enters the main state of the FSM.

The followings sections describe the DSCB software features.

DSCB Software Initialization

When the robot system is switched on, the DSCBs executes a start-up procedure (Boot phase), which prepares the boards to receive new commands from the CCU. After the configuration of the DSC hardware peripheral (DSC HI phase in Figure 2.18), the Sercos-III system is initialized to the communication phase CP2 (SCI). The DSCBs then initialize and configure the sensors and actuators hardware (SAI). If all the previous steps complete successfully without errors, the DSCBs enters the *Main Communication Stage* (MCS). At this point, the DSCBs must perform three steps before being able to reach the real-time communication phase.



DHI	:	DSC Hardware Initialization
SCI	:	Sercos-III Communication Initialization
SAI	:	Sensor Actuators Initialization
MCS	:	Main Communication Stage
MFM	:	Motor Free Mode
PM	:	Position Mode
VM	:	Velocity Mode
TM	:	Torque Mode
PEM	:	Position ELMO Mode
UF	:	Utility Functions

Figure 2.18: DSCB Software States.

The CCU must send the commands to set the offset of every EnDat sensor,[29] then the BLDC motors must be aligned[30] and finally the head subsystem must perform a homing procedure.[31] If all the previous steps have been performed without errors, the CCU starts the transition to real-time communication mode, i. e., the Sercos-III protocol changes from CP2 to CP4. Once in real-time mode, the robot can be actuated and it is ready for walking experiments.

DSCB FSM

Once the initialization phase has been completed successfully, the DSCB enters in the MCS. The latter is the core function of the DSCB communication and control software. The MCS is the root function of the DSCB FSM. It implements eight operational states and nine utility functions. The state change and the utility function execution is controlled by the CCU via the Sercos-III communication protocol.

The CCU can send commands to the DSCB via RTC or SVC. The real-time command are sent using by the IDN *Drive Control* (see Table B.1). This IDN can only be used for FSM state changes. Via SVC, and using the IDN *DSC Command over Service Channel* (see Table B.1 and Table B.9 for the explanation of the bit's meaning), the CCU can trigger an FSM state change as well as a utility command. The latter is used to control the DSCB state also before the real-time mode is initialized. Moreover, the *DSC Command over Service Channel* extends the Sercos-III protocol, implementing the specific functionalities required to control the behavior of every robot joint. Information about the state of each DSCB and the joint that they control are send to the CCU using *Drive Status* (see Table B.1), via the RTM, or the *DSCB Current fsm state* (see Table B.1 and Table B.8 for the meaning of every bit) via the SVC.

The FSM has four control states, three service states and one fall back state in case of error. The main difference between the control states and the others is that the motors are enabled and the sensors send their data to the DSCB. In the following, the possible states of the FSM are listed and described.

1. **Service States**
    - *Idle* : default state of the FSM. The DSCB does not perform any action and waits for new commands from the CCU. It also serves as an intermediate state during the state change, i. e., before every state change the DSCB must first be set to idle and then the change to the new state can be performed.

29 The calibration of every EnDat sensor is performed using a kinematic calibration jig which has been designed by Dr.-Ing Markus Schwienbacher and assembled at the Institute of Applied Mechanics. More information on the jig and on the calibration procedure can be found in [118].

30 The alignment procedure is a fundamental setup step to control the BLDC electrical motors. The position of the permanent magnet on the rotor must be precisely known in order to initialize the electronic control of the machine and determine its initial position. To perform this step the position sensors of the motor are used.

31 Because the motors on the robot's head have no absolute position sensors, the incremental encoders must be calibrated. Knowing the exact distance, which has been previously measured and saved in the DSC memory, all the three joints of the head are moved from their current position until one of the limit switches is activated. Once they have reached the end of stroke, the current position is set as calibration offset, the exact zero position can be calculated.

- *Motor On* : motor initialization stage. Each motor joint performs the alignment procedure.
- *Motor Free Mode* : sensor read state without motor control. The DSCB reads the sensor data and send them to the CCU. The motor control is switched off. This state is particularly useful to check if all the sensors are sending plausible data.

2. **Control States**
   - *Position Mode* : main control mode of the robot. The position joint controller is computed by the DSCB or by the CCU. In this state the DSCB sends a desired velocity value to the AUs. The AUs then control the joint velocity and current.
   - *Position ELMO Mode* : control mode, the DSCB sends desired position value to the AUs. The latter controls position, velocity and current of the joints. The DSCB is used only as a feed trough device.
   - *Velocity Mode* : control mode, the DSCB controls the joints velocities and sends reference current trajectories to the AUs.
   - *Torque Mode* : control mode, the DSCB directly controls the current of the motors. The AUs are used as input output devices.
3. **Fall Back State**
   - *Fault Mode* : in the case that some hardware or software errors occur the DSCB automatically switches the motors off and reaches this state. The CCU can ask for error information for debugging purposes.

The FSM also integrates nine utility functions. Three are dedicated to the AUs, five are used to change sensors parameters and the last one is used to reboot the DSCB system.

1. **AUs Utility Function**
   - *ELMO Message Mapping* : sets the mapping of the PDOs messages. Depending on the chosen joint control different desired data are send to the AUs (i. e., position, velocity or current) in every control cycle. The function can also be used to set desired feedback values that can be sent from each or some AUs to the DSCB.
   - *Homing* : perform the homing procedure for the head joints.
   - *ELMO Binary Commands* : tunnels a *Binary Command* to the AUs (see [45] for more information).
2. **Sensor Utility Function**
   - *Reset FTS* : triggers the automatic FTS calibration.
   - *Reset IMU* : triggers a reset and recalibration if the IMU sensor.
   - *Set Offset* : set the joint position offset. Using the absolute position sensor calibration value, both joint and motor side position sensors are accurately set.
   - *Get Rated Current* : read the nominal current value used by the AU to control the motor current. This value is also needed in case of current feedback to translate the current sensor data into a value expressed in amperes.

- *Set Rated Current* : write the nominal current value used by the AU to control the motor current.

3. **System Utility Function**
   - *Reboot* : triggers a software reboot of the DSC.

All the utility functions can be triggered via the SVC and are executed only if the FSM is in IDLE state for safety reasons. In case one of the functions is triggered during one of the control modes, its execution is discarded.

DSCB Safety Function and Failure State

The DSCBs are the first component of the robot system that can detect hardware failures and communication errors or sensor errors. They must quickly intervene by disabling the motor power and informing the CCU of the failure event.

The possible source of errors are the AUs and sensors. In the case of AUs the possible failure sources are communication errors via the CAN bus, CANopen protocol errors, connected sensor errors or motor failures. Each of them is defined in [43], [44] and [42]. The error codes are also stored in the DSCB software to improve and simplify the debugging process.

The sensors can have different sources of errors which may depend on the communication system used or on the hardware. The EnDat protocol defines specific bits in the *Status Register*, which must be accessed every time a new position is read from the sensor, to identify the cause of the of the error [96]. The possible errors that can occur can be hardware errors (temperature, mechanical tolerance issues, etc.) or data transmission errors (CRC or internal sensor errors). The IMU errors that can be recognized are only communication time out errors. For the FTS possible errors are those due to communication over SPI, initialization errors, sensor saturation or sensor drifting. The latter is a rather slow process that is difficult to automatically recognize. A good strategy to avoid this problem is to perform a sensor calibration procedure between successive walking experiments. The chosen incremental encoder sensors are simple and robust sensors. To identify an error the position value variation between two successive sensor sampling cycles is controlled. Depending on the current joint velocity, the position variation is not allowed to be greater than a predefined value.

The last, and safety dedicated sensors, are the light barriers. They are connected to the ELMOs general purpose input and evaluated every control cycle. A software running on the AU detects if a light barrier has been activated and communicate the failure event to the DSCB over CAN.

Since Lola is built for laboratory conditions, in order to avoid injuries, environmental and self-damage the motors are shutdown as soon as an error occurs. Right after the motor shutdown operation, the DSCB enters into *Failure State*, collects the failure information and sends it to the CCU using the predefined Sercos-III *Shutdown Error* variable (see Table B.1). For a more complete and extensible error information and classification, a custom error variable has also been defined: the *Extra error information* (see Table B.2). The latter is a vector containing additional error information. The CCU can request the values of both variables via SVC. Examples of the provided extra information are: which of

the joints has encountered the error, the CANopen code of the ELMO failure, the current and previous value of the incremental encoder in case of a variation error and so on.

2.5 Chapter Summary

In this chapter the robotic system LOLA has been introduced. The mechanical structure of the system has been briefly described. The AUs used for every actuated joint and the different sensors of the robot have been presented. The main part of the chapter has been dedicated to the introduction and description of the DSCB functionality. The custom embedded board plays a major role in the sensor-control network of the robot. It is the key component between the physical state of the robot and the CCU. The different sensor interfaces requires a flexible system component that can read all the data and make it available to the main control unit.

Another key point of the chapter is the description of all the different digital communication protocols used. On the sensors side, their implementation is an important features of the sensor-control network. It allows the local digitization of the analogue sensor values and their transmission via robust digital communication channels. On the CCU communication side, the implementation of the SERCOS-III protocol enables a high bandwidth and high data density transfer.

The features of the DSCB "Bare Metal" software has been presented and described. All the functions that interface the sensors, the AUs and manage the CCU communication have been integrated in a finite state machine structure which enable the user to access all the parameters and features of the sensors and AUs in a simple and intuitive way using the SERCOS-III IDNs. The flexibility and extensibility of the software system are fundamental for a research platform like LOLA. They are achieved with a complete custom software development which has been necessary due to the many different sensor and communication protocols needed by the application.

3 Motor Position Measurement and Motor Velocity Estimation

3.1 Introduction

The motor position is a fundamental measurement of the robot joints. It is needed by the joint control and by the CCU to get information about the joint state. As described in Section 2.3.3, LOLA has two sensors for the position of every joint[1]: incremental encoder on the motor side, absolute position encoder on the joint side. In an infinitely stiff mechanical system, there is no difference between the two positions and the use of two sensors would be redundant. Nevertheless, LOLA is a real physical system which inevitably presents both structural compliance and joint compliance. Even small elasticity in the gear box can cause a loss of motion and positioning errors, especially at high joint speed and if the joint actuates a long segment (for example the robot legs). To reduce these effects two position sensors are used for LOLA. While the interface between the DSCB and the link encoders has been described in Section 2.4.2, in this chapter the incremental encoder interface is defined and its development is described.

To achieve high control performance, to improve the system analysis possibilities and to implement advanced control algorithms, the use of a motor velocity estimation method is a consequent step. Many different mathematical algorithm have been proposed during the years to solve this problem for servo systems with only position sensing. The use of microprocessor and micro-controllers has become the state of the art in motion control systems, leading to a growing interest in discrete time systems.

Different techniques have been proposed and they can be divided in two main categories:

- methods based on signal processing, and
- model based methods.

The first types use only the information of the available position sensor, while the latter requires a system model to be implemented and, therefore, a deeper system knowledge. Moreover, if the velocity estimation is used to improve the control of the system, the method must be computable in real-time. Therefore, it should not be too complex or require too much computation power.

In this chapter the implementation of the position sensor extension feature of the LOLA hardware is discussed. Different methods to compute an on-line estimate of the velocity of a mechanical system based on position measurements will be analyzed and compared. At first, a mathematical overview of the problem is given and then various approaches are compared to find an optimal algorithm, which takes advantage of the system architecture.

1 Exception are the three joints of the head of the robot.

3.2 Incremental Encoders and Sensor Augmented Functions

The DSC used in the DSCB custom embedded boards can be configured to read from up to two incremental encoders at once. Each interface uses two hardware counters (internal peripheral of the DSC) and four DSC pins. Recalling the DSCB interface requirement discussed in Section 2.3, the DSCB must be able to interface up to three motors. Therefore, the internal DSC resources don't satisfy this important requirement.

One possible solution would be to use the two incremental encoder interfaces of the DSC and implement the third as an external device. In this case the latter must still be connected to the DSC to send the position data of the third motor. This solution has the disadvantages of reserving hardware resources of the DSC that can be used for other purposes, e. g., as high precision internal timers or to measure timing performance of the software, and to use mixed interfaces to access the same kind of data. This case is highly undesirable because introduces a mixed methods to retrieve the position data increasing the complexity of the system.

Another possibility is to use three external incremental encoder counters, or one device that can interface all three position sensors, with a unified connection to the DSC. In a first implementation, the ELMOs were used as external encoder counter devices. The ELMOs need the position information for the motor alignment procedure and for computing the motor velocity. They have a common interface to the DSC, i. e., CAN bus. The motor position information can be mapped to a CANopen PDO message (see Section 2.4.2) which is automatically sent to the DSC every time a predefined event occurs, e. g., a new desired trajectory value has been sent to the AUs. In this configuration LOLA walked for the first time and performed more than 25 presentation at the *Hannover Messe* 2010.

The limitation of this solution is the time needed by all three ELMOs to send the new measured position value to the DSC. It must be recalled that the three AUs are connected to the DSC via the same CAN channel, reducing the communication bandwidth. Moreover, these communication tasks on the ELMO do not have the highest priority compared to the control tasks, meaning that the ELMOs send this data when the higher priority tasks have been completed. The time needed by all three ELMOs to send their position via CAN is about 3 ms. This leads to a limitation of the control cycle time to 3.5 ms, i. e., a control frequency of 286 Hz (also considering the time needed by the DSC to read the other sensor data, compute the joint control and manage the CCU communication tasks).

Another variant of this solution is to develop an external hardware module that can be simply integrated in the DSCB stacked board architecture to implement the needed functionality and that can be easily extended augmenting the system capabilities. To achieve this kind of flexibility a hardware programmable device must be used, e. g., an FPGA. The *Incremental Encoder Module* (see Section 2.4.1) has been developed to solve this problem. The board is integrated in the DSCB and connected to the DSC via the Multichannel Connectors. It uses the two incremental encoder connectors of the *Power Electronics Support Board* and mounts only one connector for the third motor encoder. Because the PCB must have a reduced dimension to fit between the *Control-Communication Board* and the *Power Electronics Support Board*, an FPGA with FLASH memory integrated in package is chose. This design decision has the advantage of saving space on the PCB which can be used for other components. Three differential drivers and three CAN connectors are

mounted on the board for a future parallel implementation of the CAN communication between DSC and AUs.

3.2.1 Incremental Encoder: Principle of Operation

An incremental encoder is a position transducer that generates a cyclical output only when the device rotates. It employs two *Quadrature Digital Signals* (normally called *A* and *B*) as output which are 90 Degrees out of phase. The two signals supply embedded information about the rotating direction of the joint and an encoded position counter. The information about the rotating direction depends on which of the signals has the leading phase, while each raising or falling transition of the signals means an increment (or decrement) of the joint position.

In Figure 3.1 the quadrature signals in the two possible configurations of positive and negative rotation direction are depicted. The signals follow two defined stream patterns and the rotation direction can be identified by comparing their current value with the previous ones. A movement of the encoder in the positive direction causes an increment of the position count, while a movement in the negative direction causes a decrement.

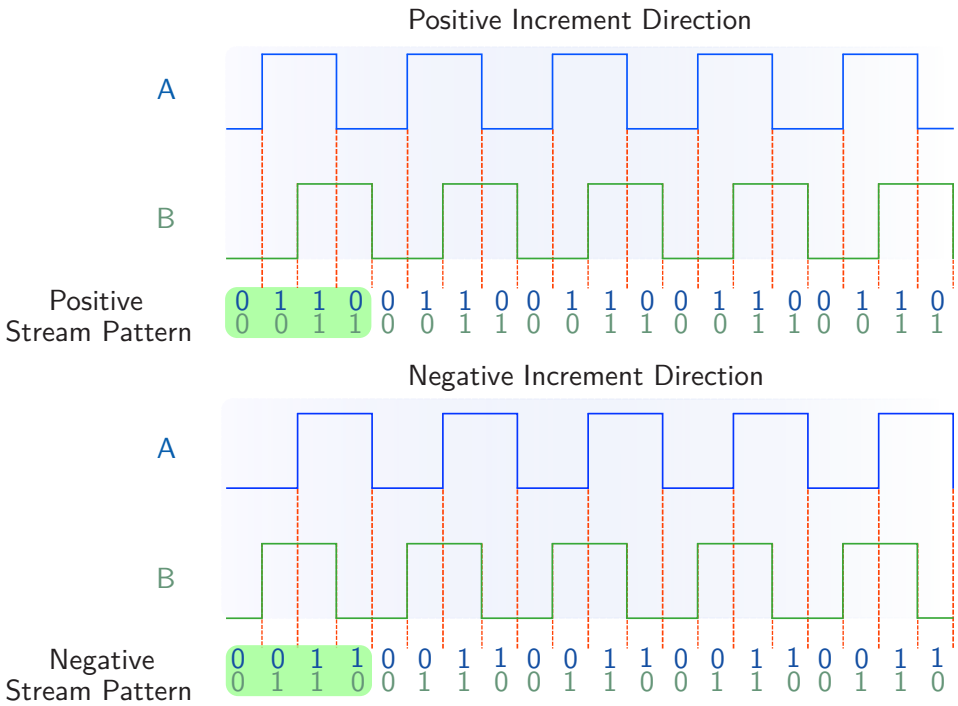


Figure 3.1: Incremental Encoder Quadrature Signals.

3.3 FPGA IP-Core Design

Developing an FPGA module means developing an IP-Core that satisfies the requirements needed for the application. The interface between the DSC and the FPGA must be defined as well as the interface with the incremental encoders. All necessary components of the IP-Core must be defined in order to *a*) decode the A and B signals, *b*) estimate the joint velocity *c*) and implement the CAN driver.

Referring to Figure 3.2 and Table 3.1, the input output interface of the IP-Core have been implemented as follows:

1. The communication with the DSC is realized via the EMI interface as for the other FPGA in the DSCB. A dual port RAM[2] is used to read from the DSC side (Port A) the sensors data and write parameters to the internal modules. On the other side of the RAM (Port B), the FPGA internal modules write the current motor positions and velocities.
2. The Data bus of the EMI has a data width of 16 bit while the position and velocity value are computed in 32 bit. A Multiplexer/Demultiplexer[3] buffer (Mux/Demux) is needed to let the communication work properly. The latter is used to control the communication on both sides.
3. The incremental encoder signals A and B are differential and must be converted to single-ended.
4. Four LEDs are used to highlight some basic and debug information: one LED is used to indicate if the device receives power, while the other three are connected to the 0 bit of each Encoder Counter and blink if the relative counter receives data from the sensors.
5. Three CAN interfaces for a parallel and efficient connection with the ELMOs.

The IP-Core is composed of different units having specific purposes: *a*) three incremental encoder decoders and counters, *b*) one joint velocity computation module, *c*) three CAN drivers and *d*) one Communication Control Logic unit (CCL). The latter is needed to arbitrate the internal and external communication with the dual port RAM. It must manage the read and write requests from the DSC, update the position and velocity data values and exchange data with the CAN drivers.

In the following subsections each unit of the IP-Core is introduced and its operation principles described.

3.3.1 IP-Core Interface and Communication Control Logic

The communication between DSC and FPGA uses the EMI interface of the DSC. The interface is composed of a 16 bit Data Bus, a 24 bit Address Bus, Read (RD), Write (WR) and Chip Select (CS) signals. The Incremental Encoder FPGA (IEF) has its own address

2 A particular kind of RAM with two identical communication ports that can be accessed independently from each side.

3 A Multiplexer is a device that selects one of several analog or digital input signals and forwards the selected input to a single output. A Demultiplexer is a device that takes a single input signal and selects one of many data-output lines.

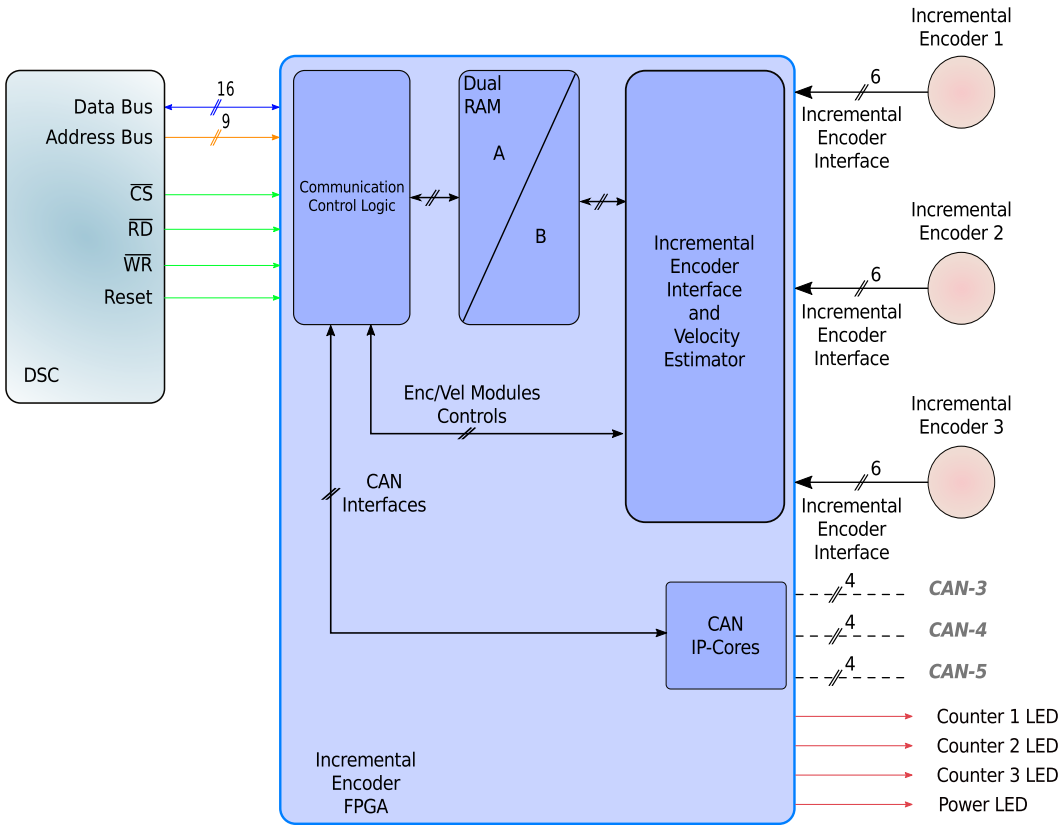


Figure 3.2: Incremental Encoder Module: FPGA Interfaces.

space that is mapped to a specific bit of the DSC peripheral memory definition. When the IEF address is set, the DSC automatically sets the IEF Chip Select signal, enabling it to receive the read or write request. After the CS signal is set, RD (or WR) is asserted and the data on the Data Bus is accessed. Every unit of the IEF also has its unique address space. The later are listed in Table D.7. Each Incremental Encoder has 16 registers that can be addressed (see Section 3.3.2 for a detailed register description) while each CAN module has 64 registers (see [149] for a definition of every available CAN register). Each register is 32 bit wide.

As already described in the previous section, a Mux/Demux buffer is needed to enable the external communication. The communication is controlled by the CCL. This unit coordinates the read/write requests from the DSC, controls the Mux/Demux buffer, enables the position data store operation in the dual port RAM, triggers the velocity estimation computation, saves the computed velocity value in RAM and manages the read/write tasks with the CAN modules. It is implemented as a Finite State Machine. While position values are constantly written to RAM using the polling technique,[4] velocity values are computed and stored in RAM only on request. To guarantee data consistency of the RAM, every time

4 The polling techniques refers to the data sampling of an external device by a controller. The external device sends his data only if the controller requests it.

Table 3.1: Interface signals specification of the Incremental Encoder Module FPGA.

Bus	Number of wires	Signal Name	Purpose
Data Bus	16	$D0$-$D15$	Data Transfer
Address Bus	9	$A1$-$A9$	Data Addressing
Control Signals	4		
		$\overline{Reset}$	Reset
		$\overline{RD}$	Read Request
		$\overline{WR}$	Write Request
		$\overline{CS}$	Chip Select
Incremental	6		
Encoder		A_p	Signal A (positive)
Interface(x3)		A_n	Signal A (negative)
		B_p	Signal B (positive)
		B_n	Signal B (negative)
		Vcc	5 V (Power Supply)
		GND	0 V (Power Supply)
CAN Interface(x3)	4		
		CAN_H	CAN-H (positive)
		CAN_L	CAN-L (negative)
		Vcc	5 V (Power Supply)
		GND	0 V (Power Supply)
LED	4		
		$Counter_1$	Position Counter 1 bit 0
		$Counter_2$	Position Counter 2 bit 0
		$Counter_3$	Position Counter 3 bit 0
		Power ON	FPGA Working

the DSC performs a data request, the CCL disables the RAM access to all the other units. A block diagram of the external interface and CCL is depicted in Figure 3.2. The *Enc/Vel Modules Controls bus* and *CAN Interfaces*, shown in Figure 3.3, groups the signals needed to synchronize all the operations of the internal units of the IP-Core. These control signals are: *Velocity Estimation Control group* (Figure 3.4), *Encoder Position Velocity Mux control group* (Figure 3.4), the RAM control signal on Port B and the *CAN Module Control group* (Figure 3.5).

3.3.2 Implementation of the Incremental Encoder Decoder

To decode the A and B signals from the encoders the following components are needed:

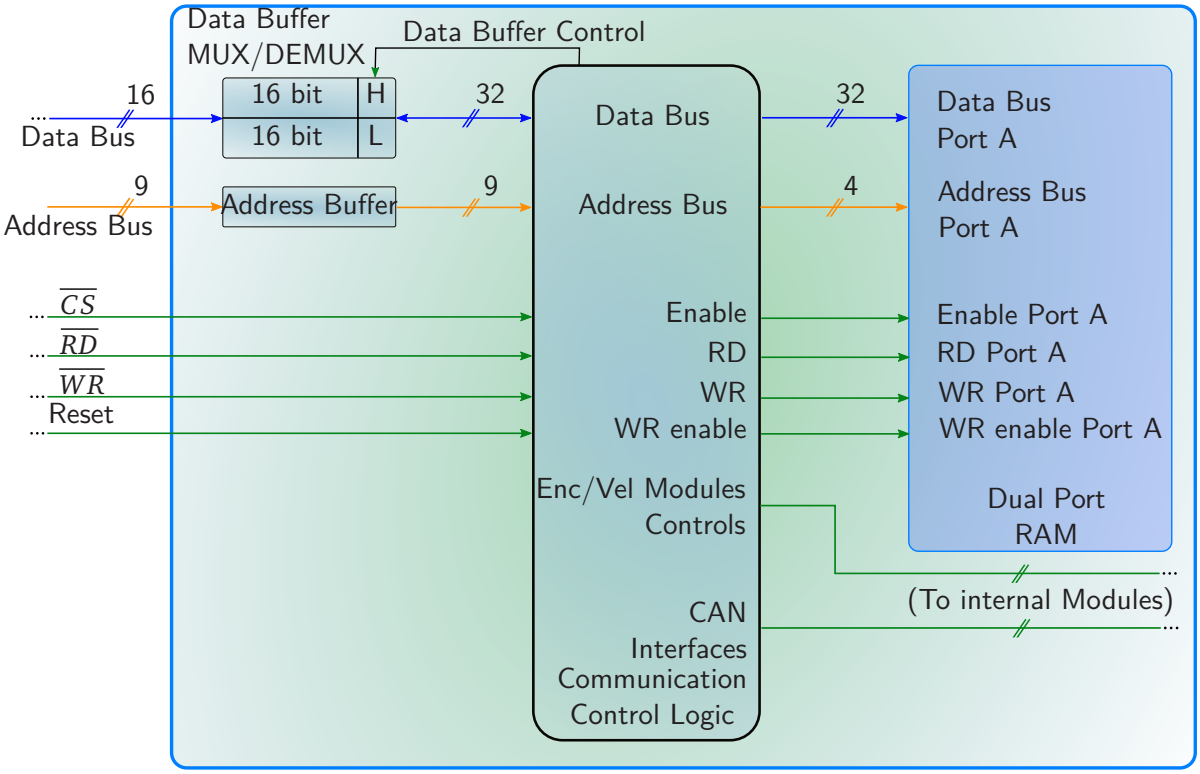


Figure 3.3: Incremental Encoder Module: FPGA Dual RAM interface with DSC.

- differential to single-end logic,
- debounce filter for the physical signals,
- signal decoder,
- position counter.

The encoder signals are transmitted as differential in order to reduce the effects of disturbances. They must be converted to single-end to be decoded and to extract the position information. The debounce filter is needed to avoid false signal transition to be considered in the position counting. This unit can be seen as low pass filter. When a signal has a transition, the unit waits a certain number of cycles before validating it and makes it available for the subsequent modules. An important parameter to avoid the loss of signal information is the maximum count transition frequency (MCF). This parameter indicates the maximum frequency of counts at the motor maximum speed. MCF depends on the encoder resolution and on the maximum motor revolution speed. It defines how many transitions per seconds an incremental encoder can generate. The decoding logic must be able to read every transition of the encoder signals, therefore, the clocking frequency of the module must be at least twice MCF. In Table A.1 the MCF parameter for every joint of LOLA is listed. Since the FPGA frequency $f_{FPGA} = 32\,\text{MHz}$ and considering the maximum value

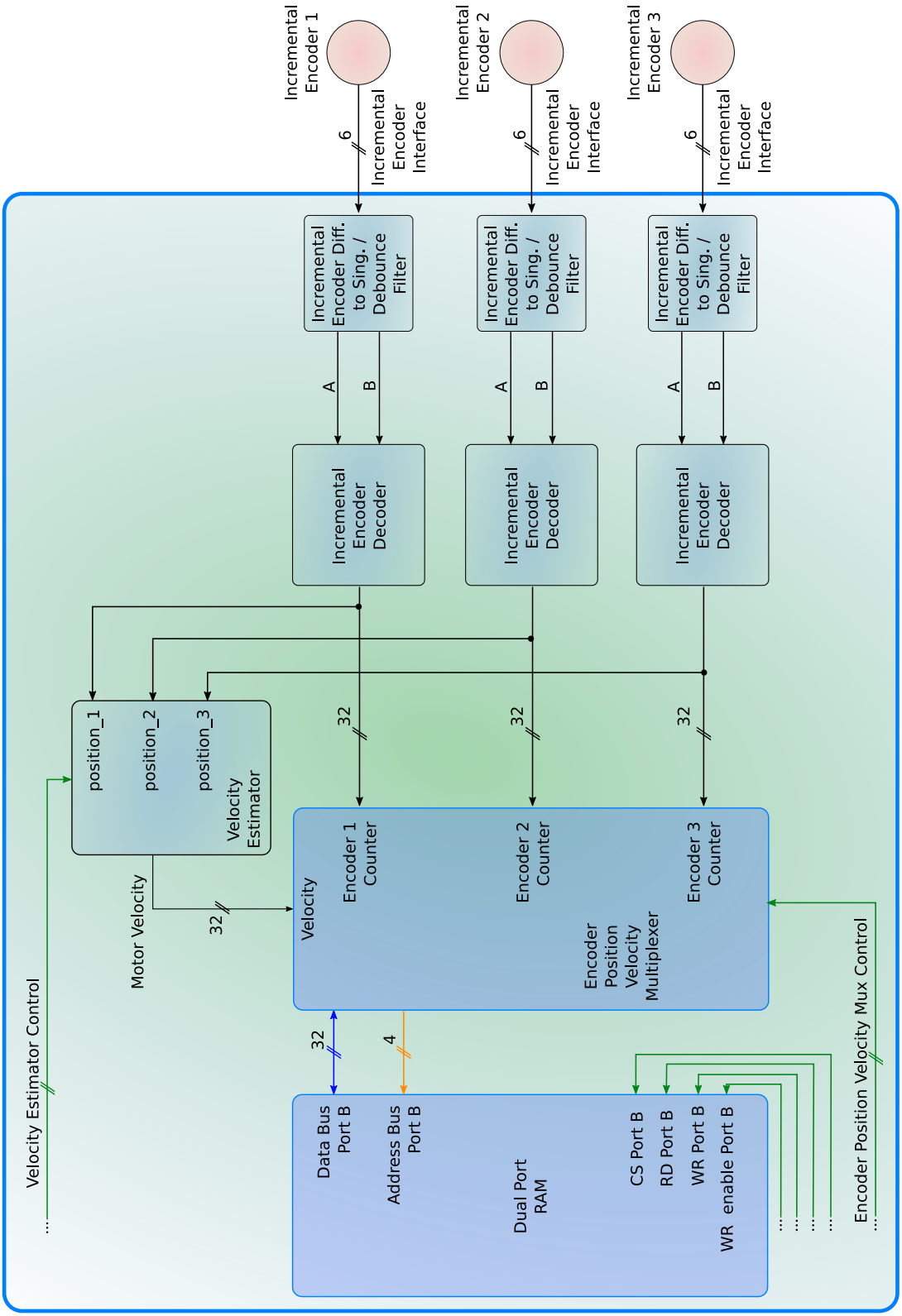


Figure 3.4: Incremental Encoder Module: Encoder decoder, Velocity estimator and Dual RAM interface.

of $MCF = 1.632\,\text{MHz}$, the maximum number of waiting cycles is computed as follows:

$$\text{Wait Cycle}_{MAX} = \frac{f_{FPGA}}{2MCF}\left[\frac{MHz}{MHz}\right] = \frac{32}{2 \cdot 1.632} \approx 9. \tag{3.1}$$

The default value of this parameter is set to 8, but it can be changed by the user.

Once the signals have been filtered, they are passed to the incremental encoder decoder. This module compares the current and previous state of the signals A and B detecting the rotation direction. If a valid transition has been recognized, the module commands the encoder counter to increment (or decrement) the current position value.

An important configuration parameter for the Incremental Encoder Decoder is the *Position Offset*. The encoder must be initialized to a proper value to avoid data inconsistency and control errors. The offset value is related to the initial position of the joint and to the calibration value of the absolute encoder and, therefore, set every time the robot is switched on. During the DSCB initialization (see Section 2.4.3), this value is automatically computed and sent to the FPGA.

The last parameter that can be set is the encoder counting direction. The positive rotation direction for each joint is defined in the multibody model of the robot. The latter must be identical also in the real robot otherwise the joint control will fail. Due to the mounting and sensor cabling it is possible that the positive encoder counting direction is opposite to the one defined in the model.

To access the encoder related data and set the parameters mentioned above, five register are defined:

1. Encoder Position: current position value.
2. Encoder Velocity: current velocity value.
3. Encoder Offset: current sensor offset.
4. Encoder Debounce Filter: number of wait cycles for the debounce filter.
5. Encoder Configuration Register: triggers the offset and wait cycle storing operation, sets the positive rotation direction parameters and performs a software reset of the decoder unit.

The value of each register can be read from the DSC, while write operations are permitted only on the last three. The block diagram for each unit is shown in Figure 3.4.

3.3.3 Encoder Velocity Estimation Module

The *Encoder Velocity Estimation Module* accepts as input the position of the three encoders, one at a time, and computes an estimation of the motor velocity for the last sampling period.

The module is controlled by the CCL. Every time the DSC requests the velocity estimation for one of the motors, the CCL stops the polling write operations for the *Incremental Encoder Decoder* and triggers the velocity computation via the Velocity Estimation Control group. Once the estimation has been computed, the value is copied to the Dual Port RAM and

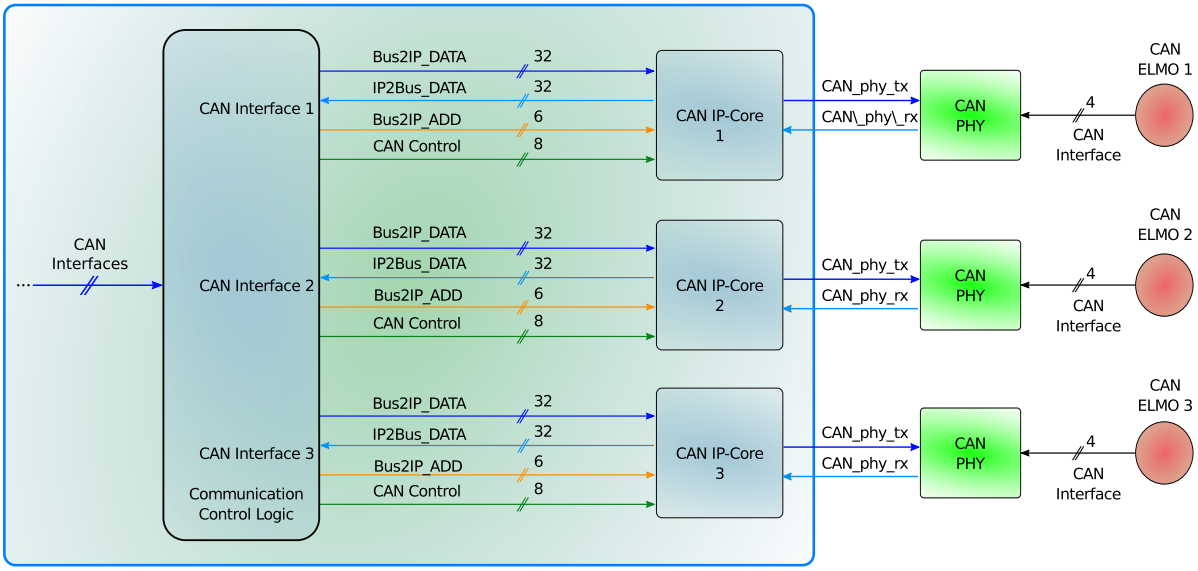


Figure 3.5: Incremental Encoder Module: CAN interface.

is transmitted to the DSC. After this operations is completed, the normal position storing operations are enabled again.

The algorithm used for the velocity computation will be described in the following section. A detailed description of the interface is given in Appendix D.

3.3.4 CAN Module Implementation

The *CAN Module* is implemented using the Xilinx CAN v3.2 IP-Core. Description of the core parameters, registers and functionalities can be found in [149][5].

The communication with the core is managed by the CCL. Figure 3.5 shows a diagram of the unit interface. A detailed description of the interface is given in Appendix D. Two independent Data buses are used: *Bus2IP_DATA* for the write operations and *IP2Bus_DATA* for the read operations. To address each register of the core the *Bus2IP_ADD* bus is used.

On the CAN bus side, the signals *can_phy_tx* and *can_phy_rx* are used to exchange data whit the PHY chip. The latter is used to convert the differential CAN bus signals CAN-H and CAN-L into single end.

At the initialization stage, the unit must be configured with the bus parameters depending on the CAN clock frequency and on the bus Baud rate. The unit needs two clocks, one is the FPGA system clock (32 MHz) used for the communication with the CCL and one for the CAN bus Communication (24 MHz). The latter is generated with a *Digital Clock Management unit* (DCM) from the system clock. The DCM is an embedded unit of the FPGA in use that manages the clock synthesis. For more information see [148].

5 Each of the three units can be parameterized and a basic CAN communication has been tested. A complete DSC API to control the core is not available at this time. Yet the communication between DSC and ELMO is still available using the DSC CAN interface.

3.4 Velocity Estimation Algorithm

3.4.1 Overview

The problem velocity computation from position data has been addressed many times over the last years. Many different approaches have been proposed and they can be divided in two categories: *a*) methods based on signal processing and *b*) model based methods.

The main difference between the two approaches is the quantity of system information needed by the estimation method. While the methods based on signal processing need only information from the position sensor, the others also need a physical model of the system to compute its velocity.

The signal processing category can be divided in two subcategories: *a*) discrete differentiation and *b*) differentiation filters.

Discrete differentiation is the oldest and simplest method and can be implemented in different forms. Hoffman [69], Kreyszig [81] and Hamming [64] give a good introduction to the classic mathematical methods of numeric differentiation. In Brown et al. [19] an overview of fixed-position, fixed-time and least square algorithms until third order is given. In Hamming [65] and Lyons [93] the problem of signal differentiation is appointed starting from the discrete differentiation to wide-band filters. Differential filters have recently gained importance especially in the field of communication engineering but they have also been used in automation and for motion control (see Carpenter et al. [31]). In Selesnick [120], a low pass FIR[6] digital filter for differentiation is presented and in Dutta and Kumar [41] a relation between minimax digital differentiation filter and Hilber transformation is exploited.

The second category of velocity estimation algorithms can also be divided in two groups: system observers and Kalman filters. System observers have been successfully used in many application as shown in [16, 75, 95, 100]. In Belanger [15] a Kalman filtering approach is presented and applied to estimate the velocity of a position controlled joint robot.

Different hardware and software implementations to estimate the velocity are discussed as in Habibullah et al. [62] where a digital speed transducer using a discrete electronics implementation and an approximated computation of the encoder impulse train frequency is presented. In Galvan et al. [53] an FPGA implementation for a wide range speed estimator combining a fixed-position and a fixed-time algorithm is discussed. The control unit on the FPGA use the best approximation of the two methods depending on the current speed range of the system. In [112, 113, 147] the authors present different microprocessor based approaches, with the support of some custom electronics. Kavanagh [78] proposes a hybrid software/hardware solution also using an FPGA and a microprocessor.

In the next subsections, an introduction to the different estimation approaches is given and the motivation for the final implementation of the LOLA velocity estimation method is discussed.

6 Finite Input Response filter.

3.4.2 Mathematical Background

The computation of the derivative of a function $f(t) : t \in \mathbb{R}$ is equivalent to the computation of its rate of change as the variable t changes of the quantity $\Delta t \to 0$:

$$\frac{\mathrm{d}\, f(t)}{\mathrm{d}\, t} = \lim_{\Delta t \to 0} \frac{f(t+\Delta t) - f(t)}{\Delta t}, \tag{3.2}$$

and it is equivalent to the computation of the line tangent to the function $f(t)$ at point t. Considering a quantization of the function $f(t)$ with a quantization step h, its derivative can be computed at every point t_n with $n = 0,1,\ldots N+1$ as:

$$\frac{\mathrm{d}\, f(t_n)}{\mathrm{d}\, t} \approx \frac{f(t_n+h) - f(t_n)}{h}, n = 0,1,\ldots N+1. \tag{3.3}$$

In this case, the line secant to the points t_n and $t_n + h$ will be computed, which is an approximation of the tangent line at point t_n.

Many numerical methods have been developed to compute the derivative of a discrete function of time. Basically, all of them use the function values and the time data available at the discrete time t_n to calculate the estimation of the function derivative.

Considering a discrete function $f(t)$ with N equally spaced sample points, and the sampling period h, the values of the function computed at each point can be calculated using the Taylor series expansion (see [69]):

$$\begin{aligned} f(t) = f(t_0) + f^{'}(t_0)(t-t_0) + \frac{f^{''}(t_0)}{2!}(t-t_0)^2 + \frac{f^{'''}(t_0)}{3!}(t-t_0)^3 + \ldots \\ + \frac{f^{N}(t_0)}{N!}(t-t_0)^N + O(h^{N+1}), \end{aligned} \tag{3.4}$$

where t_0 is a known point of the function, $h = t - t_0$ is the sampling period, $f^N(t_0)$ is the N^{th} derivative of the function $f(t)$ and $O(h^{N+1})$ is the residual error of the function expansion (proportional to h). Truncating (3.4) at the second term on the right hand side of the equation and solving for $f^{'}(t_0)$:

$$f^{'}(t_0) = \frac{f(t) - f(t_0)}{t - t_0} + O(h) = \frac{f(t) - f(t_0)}{h} + O(h), \tag{3.5}$$

an expression for the first derivative of the function $f(t)$ at the point t_0 is obtained. Using the Taylor function expansion it is possible to elaborate different types of approximation for the first and successive derivatives of the function $f(t)$ with different orders of precision.

Substituting the value of the function at time $t_0 = t_n$ and $t = t_n + h$ in (3.5), the *first forward difference approximation* is obtained:

$$f^{'}(t_n) = \frac{f(t_n+h) - f(t_n)}{h} + O(h), \tag{3.6}$$

which is equivalent to (3.3).

Substituting the value of the function at time $t_0 = t_n$ and $t = t_n - h$ in (3.5), the *first*

Table 3.2: First derivative backward difference approximation coefficients for different accuracy orders.

Accuracy Order	n	$n-1$	$n-2$	$n-3$	$n-4$	$n-5$	$n-6$
1	1	−1					
2	3/2	−2	1/2				
3	11/6	−3	3/2	−1/3			
4	25/12	−4	3	−4/3	1/4		
5	137/60	−5	5	−10/3	5/4	−1/5	
6	49/20	−6	15/2	−20/3	15/4	−6/5	1/2

Table 3.3: First derivative central difference approximation coefficients for different accuracy orders.

Accuracy Order	$n-4$	$n-3$	$n-2$	$n-1$	n	$n+1$	$n+2$	$n+3$	$n+4$
2				−1/2	0	1/2			
4			1/12	−2/3	0	2/3	−1/12		
6		−1/60	3/2	−3/4	0	3/4	−3/2	1/60	
8	1/280	−4/105	1/5	−4/5	0	4/5	−1/5	4/105	−1/280

backward difference approximation is obtained:

$$f'(t_n) = \frac{f(t_n) - f(t_n - h)}{h} + O(h). \tag{3.7}$$

Subtracting (3.6) and (3.7) the *first central difference approximation* algorithm can be computed:

$$f'(t_n) = \frac{f(t_n + h) - f(t_n - h)}{2h} + O(h^2). \tag{3.8}$$

In Figure 3.6 an example of the three methods applied to a function $f(t)$ is depicted. The derivative is computed at point t_n. As can be seen in Figure 3.6, the central difference method gives the best approximation of the tangent of the function at $f(t_n)$ compared to the other two methods.

From (3.4), if the successive derivatives are considered, higher order formulas can be computed. Those expressions yield a smaller residual error but need more sampling points to be calculated and tend to magnify measurement errors (see [56]). Some examples of coefficients for higher order differentiation approximation formulas are listed in Table 3.2 and Table 3.3 (see [52]).

From a real-time control point of view, in the case of the central difference and first forward difference approximations, the computation of the function derivative at point $f(t_n)$, must be performed with a delay of one sampling time because both methods need the function value at $f(t_{n+1})$. In many practical application this delay is not acceptable because it can lead to instability. For this reason, in many practical real-time control systems the first backward difference approximation is the preferred method for signal differentiation.

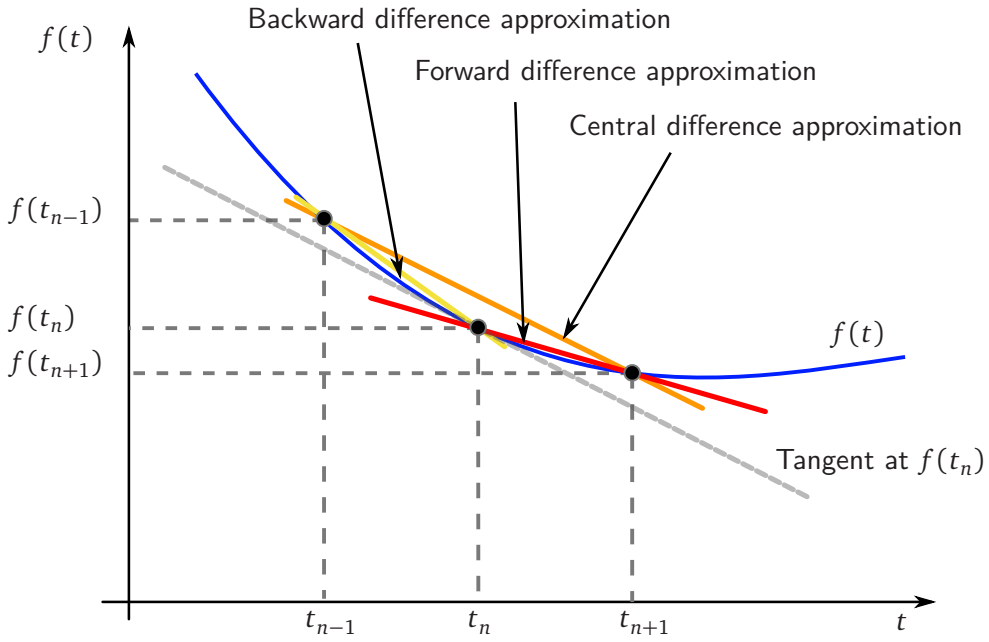


Figure 3.6: First derivative first order approximation.

3.4.3 Fixed Time and Fixed Position Increment

The backward difference approximation (3.7) is the most used and simple algorithm in control applications. It can be implemented in two different ways: *Fixed-Time Increment* (FTI) and *Fixed-Position Increment* (FPI). Defining:

$$\Delta\theta = \theta(t_n) - \theta(t_n - h), \tag{3.9}$$

$$\Delta t = t_{n-1} - t_n, \tag{3.10}$$

for FTI the position Θ is read at constant time intervals $\Delta t = h$. At each interval the velocity $\dot{\Theta}$ is then computed (3.11). In the FPI, the time Δt between constant position $\Delta\Theta = \Theta_c$ is measured. Every time that the position increment is equal Θ_c a new velocity value is computed (3.12).

$$\dot{\Theta} \approx \left.\frac{\Delta\Theta}{\Delta t}\right|_{\Delta t=\text{const.}} = (\Delta\Theta)\frac{1}{h}, \tag{3.11}$$

$$\dot{\Theta} \approx \left.\frac{\Delta\Theta}{\Delta t}\right|_{\Delta\Theta=\text{const.}} = \frac{\Theta_c}{\Delta t}. \tag{3.12}$$

Mathematically the two methods are equivalent. Nevertheless, it is shown in [19] that FTI delivers the best results at higher velocities when the number of position increments are very large. On the other side, FPI performs better at lower speed when the time between two successive position increments is long. For medium velocities the approximation performance of the two methods are similar. These effects are due to both the position information delivered from the encoder and the measured time (available as clock counts), which are integer values. Normally neither on micro-controllers nor on FPGAs floating point arithmetic[7] is implemented.

3.4.4 Narrow and Wide-Band Differentiators

The wideband differentiations have their roots in the frequency domain. The main idea of their development is the approximation of the ideal derivative of a sinusoidal signal. Defining a sinusoidal signal of amplitude A and frequency f (or angular velocity $\omega = 2\pi f$):

$$\theta(t) = A\sin(\omega t), \tag{3.13}$$

its derivative with respect to time t is:

$$\dot{\theta}(t) = \omega A\cos(\omega t). \tag{3.14}$$

If the signal (3.13) is sampled with a sampling frequency f_s, the response of (3.14) in the normalized discrete frequency domain[8] $H_{Ideal}(\omega)$ is a straight line (see Figure 3.7) proportional to the signal angular velocity ω times the signal amplitude A.

The normalized discrete frequency response of the central difference differentiator (3.8) H_{cd} and the first difference backward differentiator (3.7) H_{fd} can also be computed. The results are shown in Figure 3.7. The main difference between the two filters is that the latter acts like a high pass filter (HPS), while the first does attenuate high frequency components of the input signal. Because real signals are affected by noise, i. e., high frequency components, the central difference algorithm presents better attenuation performance. Nevertheless, its response is delayed by one sampling period.

To obtain better approximations of $H_{Ideal}(\omega)$, other methods to determine filter coefficients have been developed and are mainly used in the field of communication technology. They are normally implemented as FIR filters. Once defined the number N of the filter coefficients, the algorithm delivers their value in the normalized frequency domain. The general form of a FIR filter is as follows:

$$\dot{\theta}(n) = \sum_{k=0}^{N-1} h(k)\theta(n-k), \tag{3.15}$$

where:

- θ is the input variable,

7 Floating point arithmetic cores can be implemented in FPGA, but the number of logical gates needed is very large, especially for older FPGA models. Thus, if this units are implemented, they will use many of the logic gates available on the chip leaving less resources for other functionalities.

8 Refer to Hamming [64] and Lyons [93] for the definition of normalized discrete frequency domain.

- $\dot{\theta}$ is the filter output,
- $h(k)$ is the filter coefficient at index k,
- N is the number of filter coefficients.

Hamming [65] proposes the so called *Low-noise Lanczos filter*. The filter presents a strong high frequency attenuation and is defined as a *Narrow-Band Differentiation Filter*. Its coefficients are computed as follows:

$$h_L(k) = \frac{-3k}{M(M+1)(2M+1)}, \quad \text{with} \quad M = \frac{(N-1)}{2}. \tag{3.16}$$

If $M = 1$, i. e., FIR with 3 coefficients, the central difference differentiator are obtained. Assigning $M = 2$ the filter coefficients are:

$$h_l = [0.2; 0.1; 0; -0.1; -0.2]^T. \tag{3.17}$$

The frequency response of the latter is shown in Figure 3.7.

Another category of FIR filters proposed in the literature are *Wide-Band Differentiation Filters*. These filters give a better approximation of H_{Ideal}. Nevertheless, they also present an amplitude oscillation in the pass band and a very sharp high frequency attenuation in the stop band. The value of the filter cut-off frequency ω_c can be normally defined as input of the algorithm. In order to obtain a small oscillation in the pass band, the filter must have a high number of coefficients. An expression of wide-band differentiator coefficients computation proposed in [93] is:

$$h_{wb} = \omega_c \frac{\cos(\omega_c(k-M))}{\pi(k-M)} - \frac{\sin(\omega_c(k-M)))}{\pi(k-M)^2}. \tag{3.18}$$

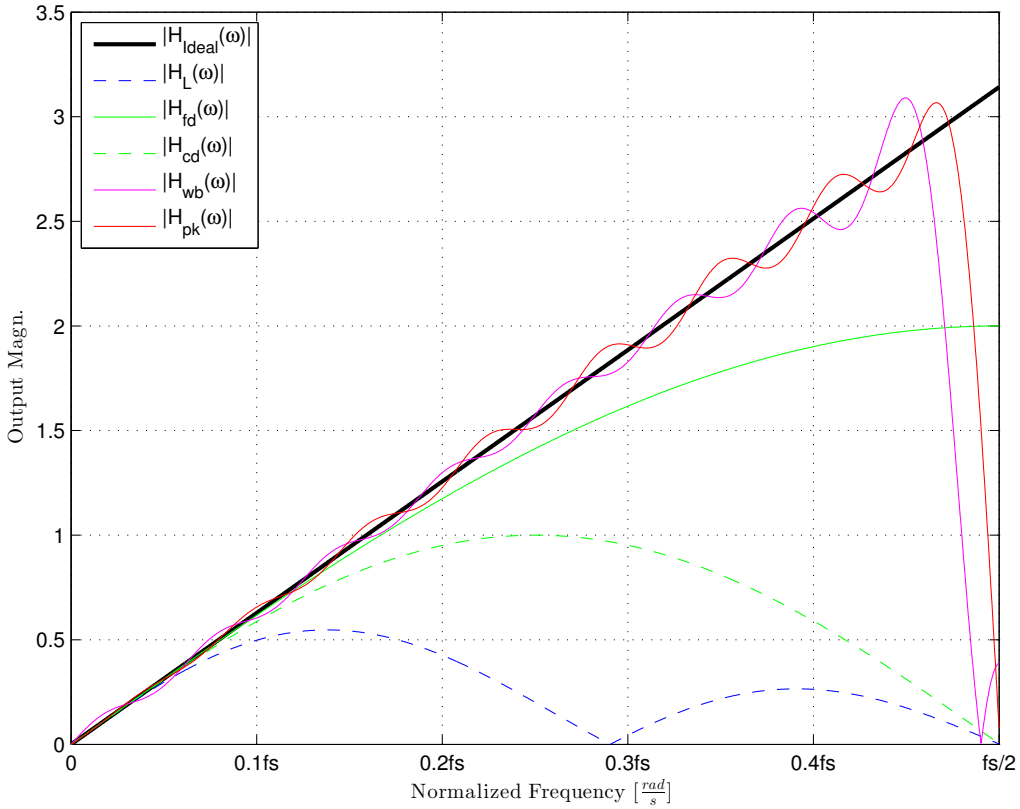
An improvement of the latter expression is the *Parks-McClellan algorithm*, proposed in [109].

In Figure 3.7, the normalized frequency response of a 32 coefficients FIR Differentiation Wide-Band filter and Parks-McClellan filter is depicted.

The Narrow and Wide-Band differentiation FIR filters have clear advantages in comparison with the first difference algorithms. They approximate the ideal differentiation frequency response in a better way and give the user the possibility to choose the cut-off frequency to reject the signal high frequency components. On the other side they also have the disadvantage of having many coefficients to reduce the amplitude oscillation in the pass band. This also causes an initial delay before the filter can deliver a valid differentiation value. Moreover, the dependency of the latest filter output from previous input values can cause undesired oscillations of the filter output in the time domain.

3.4.5 Constant Elapsed Time

The *Constant Elapsed Time* (CET) velocity estimation method was introduced in [112]. It combines the FTI and FPI methods. The main idea is to measure the occurrence time



$H_{Ideal}(\omega)$	Ideal Differentiator
$H_L(\omega)$	Low-noise Lanczos ($M = 2$)
$H_{fd}(\omega)$	Backward Difference
$H_{cd}(\omega)$	Central Difference
$H_{wb}(\omega)$	Wide Band Differentiator
$H_{pk}(\omega)$	Parks-McClellan

Figure 3.7: Digital Differentiation filters, frequency response comparison.

$t_{current}$ of each incoming increments from the position sensor $\theta_{current}$. The velocity is then computed on request using the last position and time information and the same values from the last computation interval, i. e., t_{old} and θ_{old}:

$$\dot{\theta}_{current} = \frac{\theta_{current} - \theta_{old}}{t_{curent} - t_{old}}. \tag{3.19}$$

The algorithm is an improved version of the first difference method previously discussed. The latter computes the velocity with a constant frequency and does not consider the time jitter between the encoder increment incoming time and the time of the actual algorithm computation. The precise measurement of the occurrence moment of the last encoder increment in (3.19) improves the method with a sort of sampling frequency adaptation. An essential requirement for the CET to correctly work is to have access to a high resolution timing resource which can precisely measure the occurrence time of every encoder increment. In [112], the author proposes a combined hardware and software solution to implement the CET algorithm. The hardware consists of a custom

electronic buffer system that can save the last and current position and time information of the encoder increments, while a microprocessor handles the algorithm computation and requests the information to the custom hardware. In [113] the CET algorithm is extended to the lower speed range enabling the estimation of velocities below the limit of one increment occurrence per sampling frequency cycle. The major requirement for this method to correctly work is that the clock frequency f_{ck} of the custom hardware must be fast enough to identify with sufficient precision the incoming time of each encoder increment. The latter depends on the encoder resolution R and the maximum moving speed of the system $\dot{\theta}_{MAX}$ measured in $[Hz]$. The maximum frequency of incoming encoder counts f_{ic} can be computed as:

$$f_{ic} = (R\dot{\theta}_{MAX}). \tag{3.20}$$

(3.20) indicates the minimum value for f_{ck} to correctly identify the input encoder counts. In order to increase the reliability of the system and to reduce the error in the time measurements, it is convenient to increase f_{ic} by a factor β_{ck}:

$$f_{ck} \geq \beta_{ck} f_{ic}. \tag{3.21}$$

Increasing the hardware clock frequency as described in (3.21) reduces the time measurement error by a factor of $1/\beta_{ck}$.

3.4.6 State Observer

The *State Observer* is a method to estimate the state vector of a dynamic system. It is used in control theory to retrieve information about those system states that are not directly available as measured outputs. This method of system state estimation was first discussed by Kalman [74] for statistical systems and revised for linear invariant system by Luenberger [92]. State observers have been widely used to stabilize real systems. The main requirement of this method is the availability of a model of the system under investigation. Considering a *Linear Time Invariant* (LTI) system:

$$\dot{\boldsymbol{x}}(t) = \boldsymbol{A}\boldsymbol{x}(t) + \boldsymbol{B}\boldsymbol{u}(t), \tag{3.22}$$
$$\boldsymbol{y}(t) = \boldsymbol{C}\boldsymbol{x}(t), \tag{3.23}$$

where:

- $\boldsymbol{x}(t)$ is the state vector,
- $\boldsymbol{u}(t)$ is the input vector,
- $\boldsymbol{y}(t)$ is the measured output vector,
- $\boldsymbol{A}$ is the system matrix,
- $\boldsymbol{B}$ is the input matrix,
- $\boldsymbol{C}$ is the output matrix.

The estimation of the state vector $\hat{\boldsymbol{x}}(t)$ of the system (3.22)-(3.23) can be computed as follows:

$$\dot{\hat{\boldsymbol{x}}}(t) = \boldsymbol{A}\hat{\boldsymbol{x}}(t) + \boldsymbol{B}\boldsymbol{u}(t) + \boldsymbol{L}(\boldsymbol{y}(t) - \boldsymbol{C}\hat{\boldsymbol{x}}(t)), \tag{3.24}$$

$$\dot{\boldsymbol{e}}(t) = (\boldsymbol{A} - \boldsymbol{L}\boldsymbol{C})\boldsymbol{e}(t), \tag{3.25}$$

where:

- $\boldsymbol{e}(t)$ is the observer error vector,
- $\boldsymbol{L}$ is the observer gain matrix,
- $\hat{\boldsymbol{x}}(t)$ is the estimated state vector.

The time needed by the error $\boldsymbol{e}(t)$ to tend to zero, depends on the proper choice of the gain matrix $\boldsymbol{L}$ which places the eigenvalues of the error dynamics matrix $(\boldsymbol{A} - \boldsymbol{L}\boldsymbol{C})$. The value of the gain matrix $\boldsymbol{L}$ can be set arbitrarily high to obtain a fast error dynamic, i. e., a small error settle time. Nevertheless, the stability of the error dynamic matrix, i. e., all eigenvalues must have negative real part, must be guaranteed. This fact introduces a practical limit for real world systems where sensor noise in the measured system output can endanger the observer stability. Equations (3.24)-(3.25) can also be reduced to estimate only a part of the state vector $\boldsymbol{x}$.

A well designed state observer can deliver good performance and a smooth estimation of the state vector $\boldsymbol{x}(t)$. On the other side, it needs a system model and it is particularly sensible to system parameter errors. Also, a small error of a few percent in the system dynamic matrix $\boldsymbol{A}$ can lead to a non-negligible error in the observer output $\hat{\boldsymbol{x}}(t)$.

3.4.7 Velocity Estimation Methods Comparison

As a benchmark for the comparison of the mentioned velocity estimation methods, the velocity profile shown in Figure 3.8 is chosen. It is the typical velocity trajectory of LOLA's hip join when the robot walks forward at 3.6 km/h. The related position trajectory is shown in Figure 3.9. This velocity profile is chosen to directly test the different algorithms with a real joint movement. It also has a large dynamic containing slow as well as fast movements of the joint.

All the aforementioned estimation methods have been simulated using Matlab and Simulink[9]. The wide band differentiator is a 32 coefficient Parks-McClellan filter. The state observer has been simulated using a continuous time model of LOLA hip joint with a relative error in the dynamic matrix $\boldsymbol{A}$ of 1 %. The finite difference and Constant Elapsed Time algorithms have been simulated as described in Section 3.4.2 and Section 3.4.5. The position signal used for the simulations has a superposed random error of 0.005 rad (1 encoder increment).

Absolute error $\dot{\theta}_{Ae}$ and relative error $\dot{\theta}_{Re}$ between the estimated velocity and the real joint velocity have been computed es follows:

$$\dot{\theta}_{Ae}(t) = \dot{\theta}_{estim}(t) - \dot{\theta}(t), \tag{3.26}$$

9 MATLAB and Simulink are commercial numerical computing environments, developed by THE MATHWORKS, INC. (http://www.mathworks.com/).

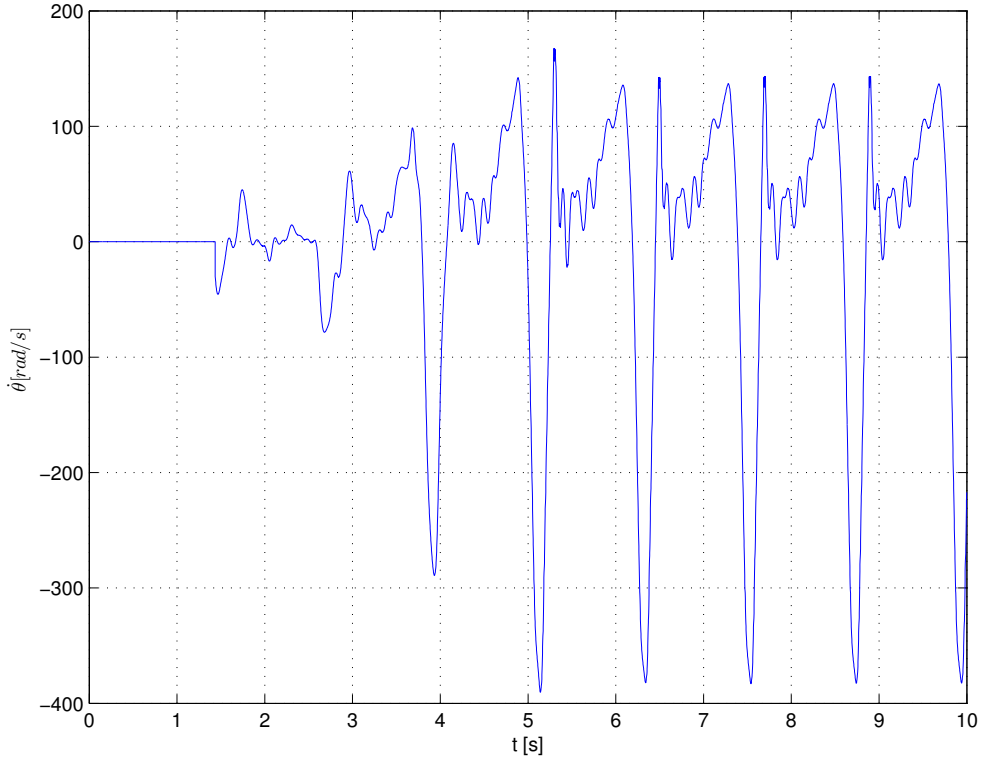


Figure 3.8: Reference velocity profile for velocity comparison.

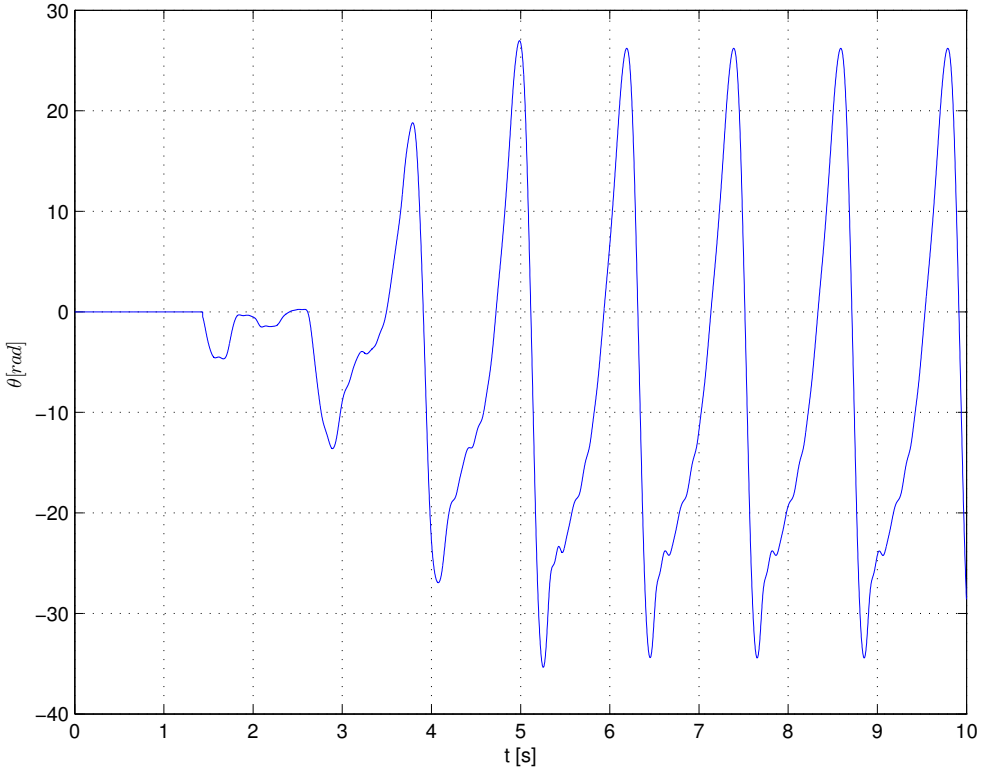


Figure 3.9: Reference position profile for velocity comparison.

$$\dot{\theta}_{Re}(t) = \frac{\dot{\theta}_{Ae}(t)}{\dot{\theta}(t)} 100, \tag{3.27}$$

where:

- $\dot{\theta}(t)$ is the real joint velocity,
- $\dot{\theta}_{estim}(t)$ is the estimated joint velocity using the investigated estimation algorithm.

To compare the algorithms, different statistical values of the absolute and relative error have been computed: root mean square (RMS), maximum value (Max), mean value (Mean) and standard deviation (Std). The results are listed in Table 3.4 and Table 3.5 for the absolute and percentage relative error respectively. As can be seen from the simulation results, the CET algorithm is the one which outperforms in all the considered statistics. It presents the lowest absolute and relative error in terms of root mean square value, peak error, mean value and standard deviation in comparison with the other methods. For this reasons it is the preferred algorithm to be implemented as joint velocity estimation method on LOLA.

Table 3.4: Velocity methods absolute estimation error statistics computed in $[rad/s]$.

Estimation Method	RMS	Max	Mean	Std
Finite Difference	3.96	308	0.76	3.89
Wide Band Differentiator	5.52	231	2.06	5.52
State Observer	7.86	175	3.06	9.97
Constant Elapsed Time	**1.57**	**82**	**0.72**	**1.40**

Table 3.5: Velocity methods percentage relative estimation error statistics.

Estimation Method	RMS	Max	Mean	Std
Finite Difference	9.05	69	3.35	**10.61**
Wide Band Differentiator	17.56	88	6.98	16.11
State Observer	9.89	48	5.48	11.16
Constant Elapsed Time	**3.60**	**43**	**3.33**	10.81

3.5 Extended Constant Elapsed Time Algorithm

In the previous section it is shown that the algorithm which delivers the best performance for velocity estimation is the CET. It is a signal processing method, i. e., independent from the system model, and requires only few parameters. On the other side it requires a microcontroller external hardware component with a high resolution clock to be implemented.

In this work, some extensions to the original algorithm to further improve the performance for the low velocity range estimation and computation time are proposed. This

improvement take advantage of the hardware topology of LOLA, but can easily be extend to any multi-axis servo system.

By exploiting the capabilities of the FPGA programmable logic, an integer division unit can be implement in hardware without the micro-controller support. In this way, the external hardware can directly deliver the velocity information when the micro-controller requests it without further computations. Because the velocity requests for each joint are always sequential, only one division unit in the FPGA can be shared between all the controlled axis. This improves the usage of the limited FPGA programmable electronics, leaving resources available for the implementation of other features. The latter is a very desirable feature because the hardware division does require many logic gates on the FPGA chip. The FPGA chip is also equipped with configurable hardware multipliers which have been used to speed up the computation and to further save logic resources.

A further improvement is the low velocity range extension. This is especially useful for feedback controlled system because it helps improving the behavior during the transition between the no-motion (or very slow motion) state to the beginning of motion. It avoids jumps in the control command of the motor. Considering a classic finite difference method, which is the basic algorithm for the CET implementation, the lower detectable velocity $\dot{\theta}_{low}$ depends on the sampling frequency of the estimation system t_{samp}. If the position of the encoder changes of one increment, positive or negative, within a sampling cycle, the lowest detectable velocity is:

$$\dot{\theta}_{low} = \frac{1}{t_{samp}}. \tag{3.28}$$

If the time detection counter used to measure the incoming time of each increment of the position sensor is not strictly limited by the sampling period t_{samp}. It can measure also longer periods of time between successive encoder increments. The limit set by (3.28) can be arbitrarily reduced enabling the counter to measure longer periods of time between successive encoder inputs. Parameter t_{limit} defines the upper saturation value of the counter and, therefore, the minimum velocity that can be measured. The smallest recognizable velocity is reduced to the following lower value:

$$\dot{\theta}_{Llow} = \frac{1}{t_{limit}}, \quad \text{with} \quad t_{limit} > t_{samp} \tag{3.29}$$

$$\dot{\theta}_{Llow} < \dot{\theta}_{low}. \tag{3.30}$$

Another feature that can be implemented to improve the transition between the slow motion to no-motion, is the inverse velocity decrement. Whenever a joint is slowing down to the no-motion state, a jump from $\dot{\theta}_{low}$ to 0 rad/s occurs. To reduce this hard transition an inverse polynomial decrement of the joint velocity can be implemented. The current value of the velocity estimation $\dot{\theta}_{current}$ is computed by dividing the previous value $\dot{\theta}_{old}$ by a suitable reduction constant factor β_r:

$$\dot{\theta}_{current} = \frac{\dot{\theta}_{old}}{\beta_r}. \tag{3.31}$$

The major effect of this feature is the improvement of the smoothness of the joint control

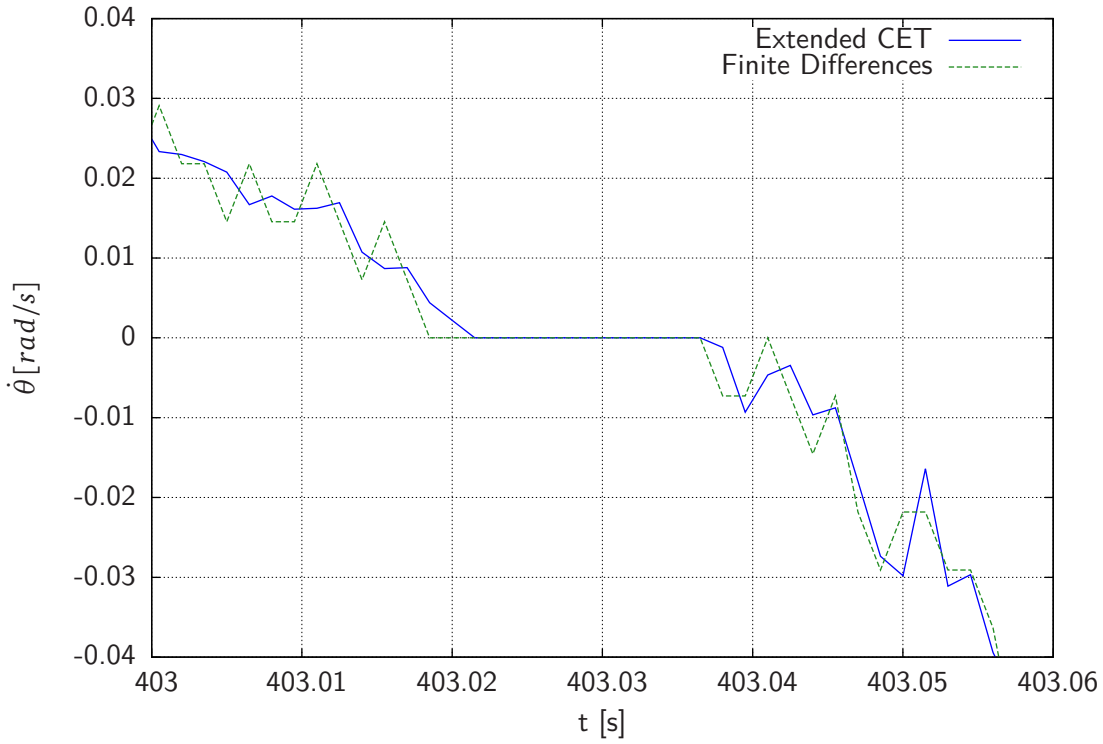


Figure 3.10: Comparison between the Extended CET and finite differences algorithms.

trajectory.

The velocity estimation unit of LOLA has all the aforementioned features. A 32 Bit integer division IP-Core from the Xilinx IP-Core Generator[10] is implemented. The lower limit of the unit time counter has been set to $t_{limit} = 4.5$ ms, i. e., three times the control sampling frequency of the robot. The reduction constant factor is set to $\beta_r = 2$.

In Figure 3.10 a comparison between the extended CET and the backward finite differences algorithms for low velocities is shown. The picture is generated from real measurement on LOLA's hip joint. The velocity computed with the extended CET presents a finer resolution of the joint velocity, a smoother transition between the no motion and start of motion region and a smoother velocity decrement to 0 rad/s.

3.6 Programmable Hardware Implementation

The velocity estimation module is implemented in the Incremental Encoder FPGA. The unit is composed of two parallel processes and two FPGA computation modules:

- one input and control process (ICP),
- one velocity algorithm process (VAP),
- one 32 bit hardware multiplier,

10 The IP-Core Generator is a software program supplied with the Xilinx ISE design development suite.

- one 32 bit division module.

To maintain a modular architecture, the input and algorithm process are kept separated. In this way, it is easier to implement improvement or different algorithms in the future.

The ICP receives the position counter information from each incremental encoder decoder. The incoming time of every position count is saved for each encoder. The time source is a process local counter running at the FPGA clock frequency. Once a velocity request is detected, ICP triggers VAP to start the velocity estimation for a specific encoder. The passed information to VAP are the current encoder count $\theta_{current}$ and its detection time $t_{current}$.

The VAP analyses the input data and, depending on the position values of the last computation θ_{old}, performs one of the following actions:

1. if $\theta_{current} \neq \theta_{old}$: The velocity is computed using the CET algorithm. For a more precise time computation, the remaining time between the end of the sampling period t_{samp} and $t_{current}$ is computed and saved to be used in the next computation.
2. if $\theta_{current} = \theta_{old}$:
   - if $t_{current} < t_{limit}$: The value t_{old} is incremented with $t_{current}$ and $\dot{\theta}_{current}$ is decremented using (3.31).
   - if $t_{current} > t_{limit}$: The value t_{old} is saturated to value t_{limit} until a new position increment is detected.

VAP uses the hardware multiplier and the division module to compute the new velocity value. The hardware multiplier is a silicon embedded component of the FPGA chip[11] and it needs only one clock cycle per operation. The division module is implemented in the FPGA logic [150] and does have constant latency of 36 clock cycles. This value depends on the number of bits of dividend and quotient.

3.7 Chapter Summary

This chapter introduces the sensor features extensions for the robot LOLA. The problem of receiving the motor position information for a reliable control of each joint of the system is solved using a dedicated electronic module. The latter drastically reduces the transmission time of the position information to the DSC, and consequently to the CCU. In the first implementation, the transmission of the position over CAN bus caused a limitation of the control cycle period. For a research platform, this kind of performance limit is highly undesirable. Taking advantage of the Incremental Encoder FPGA chip, this limitation are drastically reduced.

A robust and efficient velocity estimation is a very desirable feature in servo controlled systems. The information delivered by the velocity signal can be used for analysis purposes and to improve the dynamic control of the system. Different velocity estimation algorithms have been presented and analyzed in this chapter. Their advantages and disadvantages are discussed and their performance is compared using real measurement data of joint trajectories of the robot. The best performing algorithm in terms of maximum, mean and

11 More recent Xilinx FPGAs also provide more advanced hardware units called DSP Slice. This component can implement advanced mathematical operations.

RMS value of the absolute and relative error, i. e., the Constant Elapsed Time, is chosen. This estimation method presents advantages in terms of parameterization simplicity, requiring only the nominal sampling period of the servo system. Furthermore, improvement for the low velocity range are proposed as an extension of the aforementioned method, i. e., the extended CET algorithm.

4 Joint Model Simulation and Parameter Identification

4.1 Introduction

A simulation program for a mechatronic system is a valuable instrument for the design of the system, to choose the right sensors for the system specifications and to design the system control. The system behavior and performance can be analyzed in ideal conditions and in the presence of disturbances and noises.

In [21, 22, 23] different aspects and improvements of the simulation of JOHNNIE and LOLA are presented. In [20] a complete overview of a simulation library for electrically actuated humanoid robots is discussed. Every aspect of the dynamic simulation, methods for direct and inverse kinematics computation, path and gait generation as well as control aspect is discussed. In the simulation different models for each hardware component of the system are available: rigid and elastic joints, Harmonic Drive gears, rotational to linear movement transformers (ball screws drivers for JOHNNIE and roller screw drives for LOLA), electrical actuators, sensors, environment and obstacles models. The simulation and control program is written in the C++ programming language and, thanks to its modular structure, can be easily extended with new software modules. In Schwienbacher [118] advanced algorithms for system dynamics solving, direct and inverse kinematics, self-collision avoidance and angular momentum control are proposed. These methods are part of the system simulation for JOHNNIE and LOLA.

Using this software framework, different models of LOLA and JOHNNIE can be simulated. Depending on the purpose, simulation programs with different levels of complexity, and consequently different execution time can be built. A *reduced model simulation* assumes perfect joint tracking control and neglects effects like gears elasticity, drive dynamics and uses a simplified model of the feet contact forces. This simulation is mainly used to test high level tasks like robot trajectory planning, step sequence generation, inertial stabilization, etc. A family of *complete model simulations* considers the previously neglected effects and the complete joint tracking control. These simulations can be used for system design, joint control design and to simulate fast walking scenarios.

In this chapter, the *Low level control* aspects of the simulation software are discussed, i. e., the joint control, motor and power electronics. The implementation of a model for the joint actuation train for LOLA is introduced and the estimation of joint parameters, with the final goal of improving the system control using model based methods, is discussed.

4.2 Control System Overview

A humanoid robot is a complex mechatronic system. Its control system must take into account many different physical effects to guarantee the stability of the overall system and its performance. Especially for biped walking robots the reliability of the control and gait generation must be well organized to achieve the desired system performance. It is useful and convenient to subdivide the control of the robot in dedicated parts, each with its specific purpose, in order to keep a good overview and maintenance of the complete system. Every component has a specific interface for input and output values. This modular approach allows the developers to change and improve every component of the control system and reducing the risks of failures as long as the module interfaces compatibility is guaranteed.

For these reasons, the control system of LOLA can be divided in different functional parts that have different purposes. With reference to Figure 4.1, the modules can be identified as:

1. Input system: The interface of the robot user. Walking direction and velocity are given as input to the robot. The inputs can be generated by the user using a graphical user interface, a joystick or can be autonomously generated by the robot if the vision system is p. In the latter case the vision system does generate the required inputs depending on the environment in which the robot is moving avoiding obstacles that can be encountered along the walking path.

2. Finite state machine: Depending on the desired input, this module triggers and coordinates the gait generation and step sequence planning. If the robot should not move, no gait nor step sequence is generated.

3. Step sequence planning: this module generates the step sequence that the robot must accomplish to move in the direction and velocity requested by the input.

4. Trajectory planning: Once the step sequence has been planned, this module calculates the desired task space trajectories and the ideal contact forces to achieve the desired movement.

5. Inertial stabilization: The ideal planned contact forces must be modified depending on the state of the robot.

6. Hybrid position/force control: The stabilized trajectories in task space and contact forces must be mapped to each joint of the robot, i. e., to the joint space through inverse kinematics computation, to accomplish the desired movement. The output of this module are the desired position and velocity of every joint of the robot. These trajectories and contact force are planned to achieve the desired movement.

7. Direct Kinematics: Depending on the current state of the robot, this module computes the task space state of the system. Its output is used by the inertial stabilization and hybrid position/force control.

8. Position Control: the joint space trajectories must be controlled by local controllers. This module represents the DSCB joint controllers.

9. Velocity control, Current control and Actuation: This module represents the joint actuation train. The DSCBs output is the desired velocity. The module controls motor velocities, motor currents and, through the AUs, gears and motors, generates the desired torque.

10. Robot Joints and Sensors: This last module represents the robot joints and the sensors that read the robot state, i. e., motor positions, velocities, current, joint positions, contact forces and upper body orientation.

The complete system can be seen as composed of two interacting parts: *High Level Control* (HLC), composed by the first seven parts and *Low Level Control* (LLC) with the last three units. All the tasks are executed in real-time on the CCU and on the DSCBs.

4.2.1 High Level Control

The HLC monitors the state of the overall robot system. Major tasks of the HLC of LOLA are: the walk pattern generation (see [22]) and the stabilization of the robot (see Buschmann et al. [23]). Another important task is to ensure safety for the humans that work with the robot and for the robot itself. In order to extend the low level error check routines available on the ELMO servo drive and on the DSCBs, the following safety functions are implemented:

1. watch dog: monitoring all the HLC running processes ([20]),

2. safety check: inspection of the sensor data to detect data inconsistency ([20]),

3. self-collision: avoidance of self-collisions based on the current robot pose and shut-down in the case of an imminent collision ([118, 119]).

LOLA can be controlled in different ways (see Figure 4.1). The input data needed by the Finite State Machine (FSM) are the desired walking direction and speed. This information can be sent from a human user by a graphical user interface, a joystick or from the vision system.

The FSM triggers the gait generation and the step sequence planning. After that, the ideal contact force ($\boldsymbol{\lambda}_{id}$) and task space trajectories ($\boldsymbol{w}_{id}$) are computed. These trajectories must be stabilized against model imperfections and external disturbances ($\boldsymbol{\lambda}_d$ and $\boldsymbol{w}_d$). The desired joint-space positions ($\boldsymbol{\Theta}_d$) and velocities ($\dot{\boldsymbol{\Theta}}_d$) are computed and transmitted to the LLC.

4.2.2 Low Level Control

In Section 2.4.3 the main software features of the DSCB boards are presented. In control function mode the DSCB starts the motor control and tracks the desired trajectories received from the CCU. The implemented control algorithm uses cascaded PID controllers for motor position, velocity and current.

The current and velocity loops run on the ELMO servo drives. The first has a cycle time of 70 μs and the second of 140 μs. The position loop is implemented as proportional controller with a velocity feed-forward. The cycle time of the position controller is 1.5 ms,

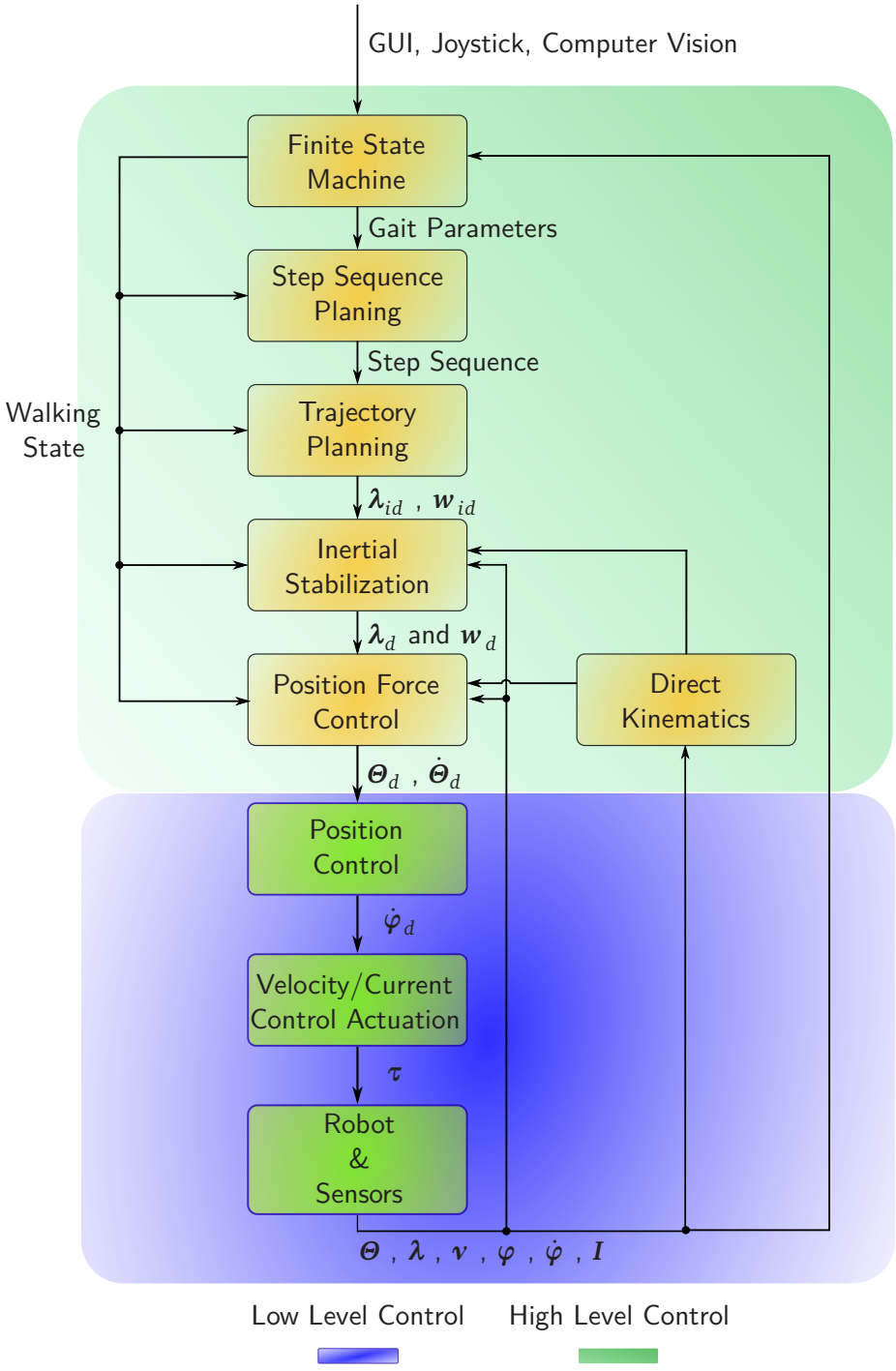


Figure 4.1: Overview of the walking control system. Where: λ: are the contact forces. w: are the trajectories in task-space. Θ_d , $\dot{\Theta}_d$: are the joint trajectories in joint-space. ν: is the vector of the upper body positions and orientations. ϕ_d , $\dot{\phi}_d$ are the desired motor positions and velocities. τ are the joints torques. Θ , φ , $\dot{\varphi}$, I are the measured joint positions, motor positions, velocities and currents respectively.

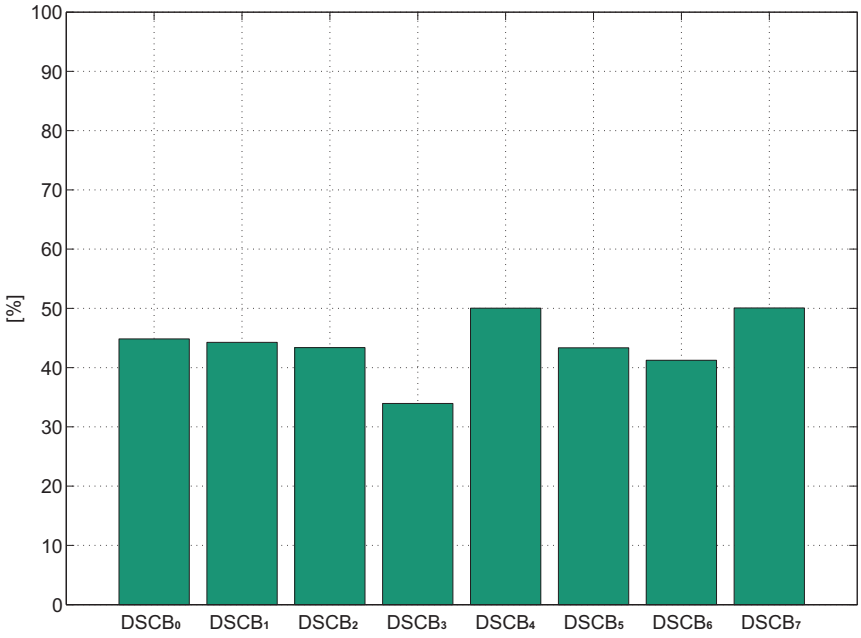


Figure 4.2: Percentage of Maximum DSCBs Control Loop time in relation with the Sercos-III cycle time.

i. e., one cycle of the SERCOS-III bus. It can run directly on the DSCB or on the CCU. In the first case the DSCBs control the motors position, while in the second case they are used as I/O devices which forward the desired velocity to the ELMO and sends the measured position to the CCU. When the position control runs on the CCU, the complete state of the robot is available. Advantages of this control structure is the availability on the CCU of a floating point unit (FPU) and that, in the case of modifications of the control law, the update must be compiled and programmed for only one device instead of eight. The FPU allows a slightly increase in the precision performance of the position controller, while the easier programmability is a more practical aspect that still must be considered for a research platform where the experimentation and consequently the changes in the software, are very frequent. The disadvantage of this implementation is a latency of one SERCOS-III cycle (t_s =1.5 ms) in the position control loop.

In Table 4.1 the mean $t_{cl\,mean}$ and maximum $t_{cl\,max}$ duration of a complete control loop on the DSCBs are reported. The values are computed over 168960 cycles (253.44 s). The measurements consider the time consumed by a DSCB to read the new desired trajectories t_{tr}, sample the sensors t_{ss}, control the actuators t_{ca} and send the feedback data to the CCU t_{fb}:

$$t_{cl} = t_{tr} + t_{ss} + t_{ca} + t_{fb} \tag{4.1}$$

As can be seen, the mean and maximum values depend on the considered DSCBs. That can be explained recalling that the number of sensors and actuators connected to a DSCB can vary, see Section 2.4. Considering $DSCB_4$ and $DSCB_7$, which interface the same sensors

Table 4.1: DSCBs Control Loop time.

DSCB Name	$t_{\text{cl mean}}\,[\mu s]$	$t_{\text{cl max}}\,[\mu s]$
DSCB_0	566	672
DSCB_1	558	663
DSCB_2	547	650
DSCB_3	423	509
DSCB_4	644	750
DSCB_5	550	650
DSCB_6	495	618
DSCB_7	644	750

and actuators (see Figure 2.7, *Distributed Sensor Control Board i*), they need the same time to complete a control cycle.

An interesting comparison is the percent of Sercos cycle time consumed by every DSCB in the worst case, i. e., $t_{\text{cl max}}$ (Figure 4.2). As can be seen none of the DSCBs overcame the 50% of the SERCOS-III cycle. This means that the DSCBs have enough available computation resources and, from the LLC point of view, it would be possible to further reduce the SERCOS-III cycle time.

4.3 Lola Multibody Simulation

To design the control of a mechatronic system a dynamic model is required. Buschmann [20] has developed a multibody simulation to simulate both LOLA and JOHNNIE. Methods to develop the dynamic equation of a mechanical systems can be found in [110, 142]. Specific reads about industrial robotic systems are [33, 125] and [73] describe the model of legged robots. The Lagrangian dynamic model of a humanoid robot, actuated by electrical motors with unilateral ground contacts, and its *Equation of Motion* (EoM) can be written as follows[1]:

$$\boldsymbol{M}(\boldsymbol{q})\ddot{\boldsymbol{q}} + \boldsymbol{C}(\boldsymbol{q},\dot{\boldsymbol{q}})\dot{\boldsymbol{q}} + \boldsymbol{G}(\boldsymbol{q}) = \boldsymbol{Q}_{drive} + \boldsymbol{Q}_{\boldsymbol{ext}}, \tag{4.2}$$

where:

- $\boldsymbol{q} = [\boldsymbol{\nu}, \boldsymbol{\Theta}]^T$: is the vector of the generalized coordinates of the robot, or configuration space. $\boldsymbol{\nu}$ is the vector of upper body positions and orientations and $\boldsymbol{\Theta}$ is the vector of the joint space coordinates,
- $\boldsymbol{M}(\boldsymbol{q})$: is the mass matrix of the system,
- $\boldsymbol{C}(\boldsymbol{q},\dot{\boldsymbol{q}})$: is the matrix of centrifugal and Coriolis forces,
- $\boldsymbol{G}(\boldsymbol{q})$: is the generalized gravity force vector.

1 To reduce the complexity of the equation the time dependency of the variable is omitted.

- $\boldsymbol{Q}_{drive} = [0, \boldsymbol{\tau}]^T$: is the joint actuation torque vector,
- $\boldsymbol{Q_{ext}}$: is the vector of the forces generated by ground contacts[2].

The main objective of this work is the control of the joints, for these reason from now on only the joint space part of the model is considered. Considering only the joint space coordinates $\boldsymbol{\Theta}$ and the dynamic model of motors and rigid gears, the following dynamic equations can be written:

$$\boldsymbol{M}_J(\boldsymbol{\Theta})\ddot{\boldsymbol{\Theta}} + \boldsymbol{C}_J(\boldsymbol{\Theta}, \dot{\boldsymbol{\Theta}})\dot{\boldsymbol{\Theta}} + \boldsymbol{G}_J(\boldsymbol{\Theta}) = \boldsymbol{\tau} + \boldsymbol{Q}_{Jext} + \boldsymbol{Q}_{\boldsymbol{\nu}}, \tag{4.3}$$

$$\boldsymbol{L}\dot{\boldsymbol{I}} + \boldsymbol{R}\boldsymbol{I} + \boldsymbol{k}_M\dot{\boldsymbol{\varphi}} = \boldsymbol{U}, \tag{4.4}$$

$$\boldsymbol{k}_\tau \boldsymbol{I} = \boldsymbol{\tau}_m, \tag{4.5}$$

$$\boldsymbol{\tau} = \boldsymbol{N}(\boldsymbol{\Theta})\boldsymbol{\tau}_m, \tag{4.6}$$

where:

- $\boldsymbol{M}_J(\boldsymbol{\Theta})$: is the joint space mass matrix.
- $\boldsymbol{C}_J(\boldsymbol{\Theta}, \dot{\boldsymbol{\Theta}})$: is the joint space matrix of Coriolis and centrifugal forces.
- $\boldsymbol{G_J}(\boldsymbol{\Theta})$: is the vector of gravitational forces acting on the joints.
- $\boldsymbol{\tau}$: is the vector of the joint actuation torques.
- $\boldsymbol{Q}_{\boldsymbol{Jext}}$: is the vector of the external contact forces projected into the joints coordinates.
- $\boldsymbol{Q}_{\boldsymbol{\nu}}$: is the vector of the forces related to the upper body coordinates $\boldsymbol{\nu}$ acting on the joints.
- $\boldsymbol{\Theta}, \dot{\boldsymbol{\Theta}}, \ddot{\boldsymbol{\Theta}}$: are the joint positions, velocities and accelerations respectively.
- $\boldsymbol{U}$: is the vector of the voltage applied to the motors.
- $\boldsymbol{\tau}_m$: is the vector of motor torques.
- $\boldsymbol{L}$: is the diagonal matrix of motor inductance.
- $\boldsymbol{R}$: is the diagonal matrix of motor resistance.
- $\boldsymbol{k}_M$: is the diagonal matrix of motor back EMF[3] constants.
- $\boldsymbol{k}_\tau$: is the diagonal matrix of motor constants. This value models the properties of the motor of transform the current I into the actuating torque τ_m[4].
- $\boldsymbol{N}(\boldsymbol{\Theta})$: is the diagonal matrix of joint gears reduction ratio. In the case of LOLA this matrix is variable with the joint configuration $\boldsymbol{\Theta}$ due to the non-linear kinematic of the knee and ankle joints.

2 $\boldsymbol{Q_{ext}}$ contains the resulting generalized forces due to ground contact. Unilaterality conditions are not shown for simplicity.

3 Electromotive Force.

4 For many motors used in servo applications $\boldsymbol{k}_\tau \simeq \boldsymbol{k}_M$.

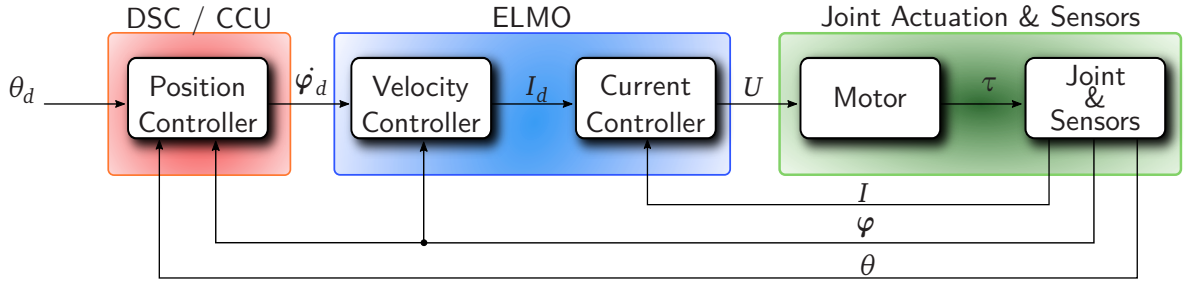


Figure 4.3: Structure of the joint control of Lola.

- $\varphi, \dot{\varphi}, \ddot{\varphi}$: are the vectors of motor positions, velocities and accelerations respectively.

The system (4.3) represents a very general model of a multibody system and permits to easily consider non-linear effects, disturbances and friction forces. Equations (4.4)-(4.5) are the dynamic equations of a DC motor. Nevertheless, they can still be used for PMSM motors if the Clarke [34] and Park [108] transformation and a *Field Oriented Control* (FOC) (see [17, 98]) algorithm are used. In Park's coordinates, the current has two components: the magnetizing current I_d and current producing torque I_q. In the FOC control algorithm the first is regulated to 0, while the latter is controlled to track the desired current and, consequently, to produce the required torque profile. Under these conditions, the PMSM motor can be considered equivalent to a DC machine. Equation (4.6) considers the amplification of the motor torque through the joint gears.

For further details about the development of the dynamic model (4.3)-(4.6) and kinematic model of Lola (and Johnnie) see [20] and [118].

4.4 Extension of Lola Multibody Simulation: Actuation System

The multibody simulation of Lola is a comprehensive software that considers every aspect of the robotic system. Nevertheless, the joint control can be extended considering the characteristic of the actuation train. This includes the implementation of: models for the AUs, the use of the real cycle time of the control loops (considering also the latency of the control signals) and the use of integer arithmetic in the LLC and sensor data.

As it is shown in Chapter 2 and previously in this chapter, the LLC implemented for Lola has a distributed structure of independent local joint controllers. In Figure 4.3 the structure of one of the joint controller of the robot is depicted. The current and velocity controls take place in the AUs, i. e., the ELMOs, while the position control can be implemented either on the DSCBs or directly on the CCU.

This kind of joint control is one of the simplest but still very robust technique to achieve satisfying results in terms of tracking errors, disturbance rejection and effort for controller

tuning. Considering the reduced joint space system of the actuated joint DoFs (4.3), the approach proposed in [125] can be used. Left-multiplying by the constant diagonal matrix[5] $\boldsymbol{N}^{-1}$, (4.3) can be rewritten on the motor side of the joint:

$$\begin{aligned} \boldsymbol{N}^{-1}\boldsymbol{M}_J(\boldsymbol{\Theta})\boldsymbol{N}^{-1}\ddot{\boldsymbol{\varphi}} + \boldsymbol{N}^{-1}\boldsymbol{C}_J(\boldsymbol{\Theta},\dot{\boldsymbol{\Theta}})\boldsymbol{N}^{-1}\dot{\boldsymbol{\varphi}} + \boldsymbol{N}^{-1}\boldsymbol{G}_J(\boldsymbol{\Theta}) &= \boldsymbol{\tau}_m \\ &\quad + \boldsymbol{N}^{-1}\boldsymbol{Q}_{Jext} + \boldsymbol{N}^{-1}\boldsymbol{Q}_v, \end{aligned} \tag{4.7}$$

$$\boldsymbol{N}^{-1}\boldsymbol{\tau} = \boldsymbol{\tau}_m. \tag{4.8}$$

For high values of gear transformation ratio $\boldsymbol{N}$, as it is the case for LOLA, and for medium or slow velocities, the effects due to the robot position configuration vectors $\boldsymbol{C}_J(\boldsymbol{\Theta},\dot{\boldsymbol{\Theta}})$ and $\boldsymbol{G}_J(\boldsymbol{\Theta})$ can be considered as disturbances. Neglecting the joint coupling effect of off-diagonal elements of the mass matrix $\boldsymbol{M}_{aj}(\boldsymbol{\Theta})$ and observing that the diagonal elements can be seen as sum of the mean values of joint inertia $\overline{\boldsymbol{M}}_{aj}$ and a configuration dependent term $\boldsymbol{\Delta M}_{aj}(\boldsymbol{\Theta})$, the following equation can be written as:

$$\boldsymbol{M}_{aj}(\boldsymbol{\Theta}) = \overline{\boldsymbol{M}}_{aj} + \boldsymbol{\Delta M}_{aj}(\boldsymbol{\Theta}). \tag{4.9}$$

Consequently, (4.7) can be rewritten as:

$$\boldsymbol{\tau}_m = \boldsymbol{N}^{-1}\overline{\boldsymbol{M}}_{aj}\boldsymbol{N}^{-1}\ddot{\boldsymbol{\varphi}} + \boldsymbol{d}, \tag{4.10}$$

$$\begin{aligned} \boldsymbol{d} = {} & \boldsymbol{N}^{-1}\boldsymbol{\Delta M}_{aj}(\boldsymbol{\Theta})\boldsymbol{N}^{-1}\ddot{\boldsymbol{\varphi}} + \boldsymbol{N}^{-1}\boldsymbol{C}_J(\boldsymbol{\Theta},\dot{\boldsymbol{\Theta}})\boldsymbol{N}^{-1}\dot{\boldsymbol{\varphi}} + \boldsymbol{N}^{-1}\boldsymbol{G}_J(\boldsymbol{\Theta}) \\ & - \boldsymbol{N}^{-1}\boldsymbol{Q}_{Jext} + \boldsymbol{N}^{-1}\boldsymbol{Q}_v, \end{aligned} \tag{4.11}$$

where $\boldsymbol{d}$ represents all the configuration dependent terms, the external forces applied to the robot and the joint friction. Equation (4.10) is a linear decoupled system of equations, while (4.11) is a non-linear coupled system.

For high gear reduction ration $\boldsymbol{N}$ and moderate joint velocities, (4.10) can be used to design control schemes for decoupled single input single output systems. Vector $\boldsymbol{d}$ can be considered as the vector of system disturbances to be rejected. This control scheme has been demonstrated to be successful for LOLA, letting the robot reach a walking velocity of 2.7 km/h (Favot et al. [48]), 3.34 km/h (Buschmann et al. [24]) and also 3.6 km/h.

4.4.1 Lola Control Scheme

Current Control Loop

The current control loop is implemented in the AUs. It is a proportional integral (PI) tracking controller. Defining the tracking error as $e_I = I_d - I_m$, the control law is defined as follows:

$$C_I = K_{pi}e_I + K_{ii}e_{Ii}, \tag{4.12}$$

$$e_{Ii} = \int e_I dt, \tag{4.13}$$

5 This assumption is explained in detail in Section 4.5.1.

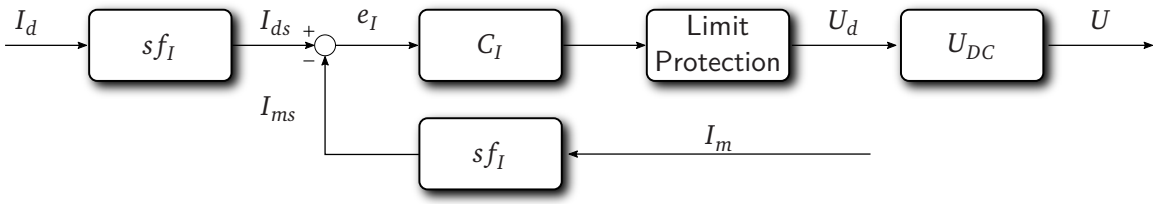


Figure 4.4: Block diagram of the motor current controller.

where K_{pi} is the proportional gain, K_{ii} is the integral gain, I_d is the desired motor current, I_m is the measured motor current and e_{Ii} is its integral of the tracking error. Considering a single actuated joint of the motor dynamics equations (4.4)-(4.5) and applying (4.12), the motor current error equation can be written as:

$$\dot{I} = -\frac{R}{L}I + \frac{1}{L}U - \frac{k_\tau \dot{\varphi}}{L}, \tag{4.14}$$

$$\ddot{e}_{Ii} + \left(\frac{R}{L} + \frac{K_{pi}}{L}\right)\dot{e}_{Ii} + \frac{K_{ii}}{L}e_{Ii} = \frac{1}{L}\left(k_\tau \dot{\varphi} + R\dot{e}_{Ii}\right). \tag{4.15}$$

The current control is shown as a block diagram in Figure 4.4. The current scaling factor sf_I converts the current signals (I_d is the desired current and I_m is the measured current) from $[A]$ into the integer value used by the AU current controller (I_{ds} is the scaled desired current and I_{ms} is the scaled measured current). The *Limit Protection* block at the controller output protect the power electronics from too high input signals which could cause over-voltages and loss of stability of the control loop. The power electronics is a first approximation model of an impressed voltage inverter in Parks coordinates. Its input U_d is $-1 < U_d < 1$ and $U_{DC} = 80\,\mathrm{V}$ is the maximum available voltage for the motors. The output voltage U is then $-U_{DC} < U < U_{DC}$.

The tuning of the PI controller, i. e., the search of the optimal values for K_{pi} and K_{ii}, is an automatic procedure implemented in the ELMO (see [45], [43] and [46]). This procedure must be executed directly on the system to be controlled when the unit is initialized for the first time. It can also be repeated for further improvements of the system or in case the system is changed. The values can also be manually set by the user.

In Figure 4.5, an example of the current tracking performance of the left knee AU is shown. The measured current I_m perfectly follows the desired current trajectory I_d. This profile was measured while the robot was stepping on the spot. Because of the good performance of the current control loop the electrical dynamic can be neglected for the design of the position and velocity loop. With this assumption each motor can be considered as a torque generator and modeled as:

$$k_\tau I = \tau_m. \tag{4.16}$$

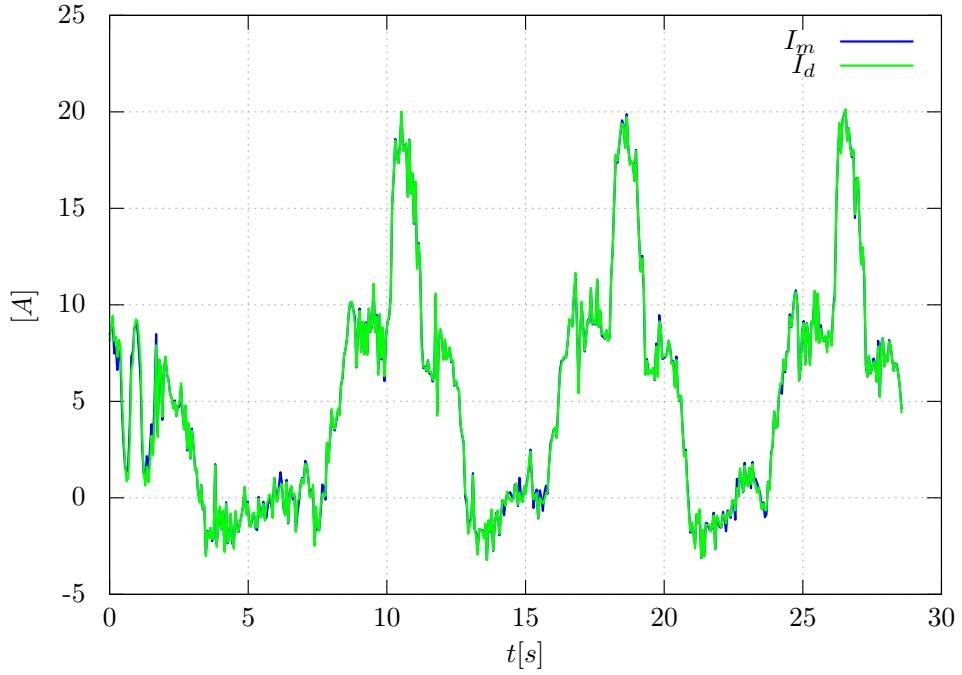


Figure 4.5: Measured and Desired current for the left knee joint.

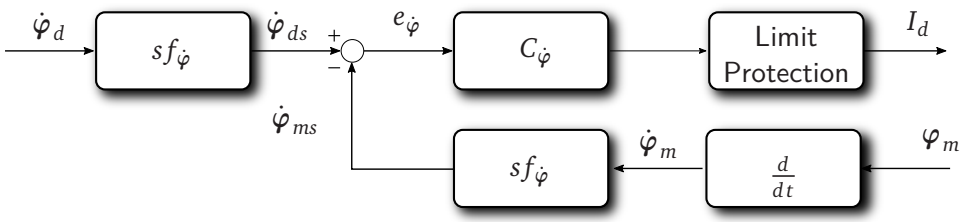


Figure 4.6: Block diagram of the motor velocity controller.

Velocity Control Loop

The velocity controller is also implemented in the AUs. One of the major advantages is the fast execution time of this control loop. It offers a valuable smoothness and adds a desirable velocity dependent damping to the joint control loop. It is implemented as a proportional (P) controller:

$$C_{\dot{\varphi}} = K_{pv} e_{\dot{\varphi}}, \tag{4.17}$$

$$e_{\dot{\varphi}} = \dot{\varphi}_d - \dot{\varphi}_m. \tag{4.18}$$

The velocity control is shown as a block diagram in Figure 4.6. The current velocity of the motor is computed by differentiating the position signals φ_m every sampling cycle. The velocity scaling factor $sf_{\dot{\varphi}}$ converts the velocity signals ($\dot{\varphi}_d$, desired motor velocity and $\dot{\varphi}_m$ measured motor velocity) in $[\frac{counts}{s}]$ ($\dot{\varphi}_{ds}$ scales desired motor velocity and $\dot{\varphi}_{ms}$ scales

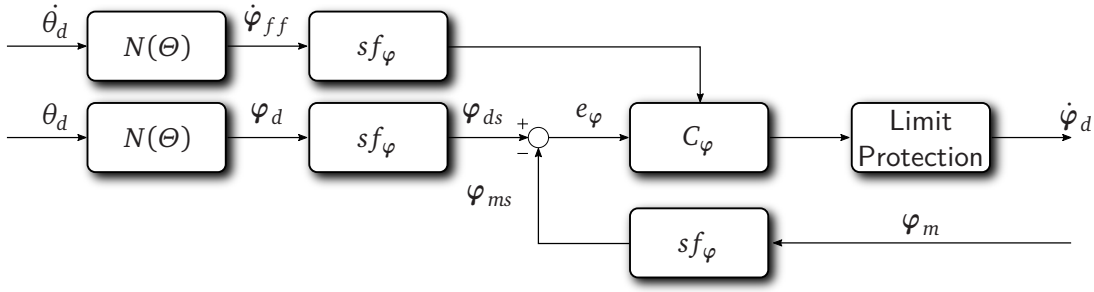


Figure 4.7: Block diagram of the motor position controller.

measured motor velocity).

Position Control Loop

The position control loop is implemented as a proportional tracking controller with velocity feed-forward signal. It can be executed on the DSCB or on the CCU:

$$C_\varphi = K_{pp} e_\varphi + \dot{\varphi}_{ff}, \tag{4.19}$$

$$e_\varphi = \varphi_d - \varphi_m. \tag{4.20}$$

The position control is shown as a block diagram in Figure 4.7. The desired joint position Θ_d and velocity $\dot{\Theta}_d$ are converted to motor position φ_d and velocity $\dot{\varphi}_d$ using the joint gear transformation function $N(\Theta)$.

The position scaling factor sf_φ converts the position signals (φ_d desired motor position and φ_m measured motor position) from $[rad]$ into a suitable scaled integer value (φ_{ds} is the scaled desired motor position and φ_{ms} is the scaled measured motor position). The latter is then converted on the DSBC into encoder counts. sf_φ is computed in order to take advantage of all the range of a 32 Bit integer value on the DSC. The joints have different encoder resolution (see Table A.1) and with the goal of using only one value for each all joints, the *maximum common multiplier* between all the encoder resolutions is computed. Having only one scaling factor for all joints improves the transparency of the hardware to the programmer and simplifies the control algorithm. The use of the velocity feed-forward signal improves the trajectory tracking of the joint position reducing the phase lag between the desired and measured position.

Control Loops in the Robot Simulation

The previously discussed control loop is implemented in the *full model simulation* of LOLA. Each loop is executed with it is own cycle time as described in Section 4.2.2. Implementing the complete control structure of the robot gives the advantages of a powerful detailed analysis tool. It allows to analyze the system performance at every control level, to observe and understand its limits and to understand the causes of eventual problems and failures that may occur during the experiments. On the other side, the implementation of such a detailed model, causes longer simulation times and the generation of a bigger amount of

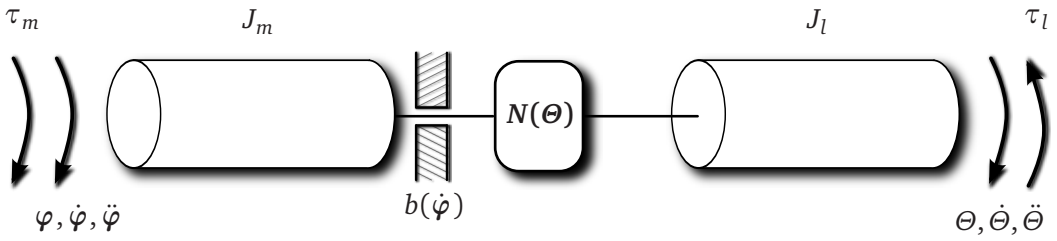


Figure 4.8: Model for a stiff joint.

data. The program must be executed with an integration time step constant well below the sampling time of the current control loop ($70\,\mu s$) and is typically set to $7\,\mu s$.

With these improvements, the full multibody simulation of LOLA uses the same control loop gains as for the real robot.

4.5 Lola Joint Model

To improve the joint control of the robot, a suitable model is required. Effects like friction and stiffness must be taken into account to reduce the tracking error. Especially in the case of fast walking speeds, i. e., fast trajectory tracking, those effects play a major role.

The transmission component of the actuation train of LOLA are roller screw drives for the knee and ankle and Harmonic Drive gears for the other actuated DoFs. Both elements present high but limited stiffness which can lead to deformation in the gears. This deformation induces a time varying displacement between the position of the motors and the position of the joint after the transmission. This effect can lead to instabilities of the control chain and of the entire robot system. For joints with a long actuation link as the hip (0.984 m) or knee (0.544 m), an error of a few $mrad$ in the position tracking result in positioning errors of the end effector, i. e., the foot, in the order of cm. The latter also leads to stronger impact to the ground, higher impulse disturbances in the actuation train and consequently to bigger deformation in the joints. All these effects can be classified as disturbances that the joint control must handle and limit their impact on the system stability. In this section, two kinds of model are considered: *reduced model* of the robot joints which considers an ideal transmission elements with infinite stiffness, an *extended model* implements joints with finite stiffness. To simplify the analysis and the implementation of a control scheme, a joint local approach based on the reduced actuation system described in (4.7) is chosen. Contrary to the system global approach in Section 4.3, the latter considers a model in the joint local coordinates with concentrated parameters.

4.5.1 Reduced Model

If the electrically actuated joints are considered ideally stiff, they can be modeled using three elements:

- the motor inertia J_m,
- the transmission ratio $N(\Theta)$,

- the link inertia J_l.

Figure 4.8 shows a block diagram of this model. The torque τ_m acts on the motor inertia generating the motion described by the motor position, velocity and acceleration $\varphi, \dot{\varphi}$ and $\ddot{\varphi}$ respectively. $b(\dot{\varphi})$ is the motor side friction as a function of the motor velocity and is part of the transmission model. The link side motion described by position, velocity and acceleration $\Theta, \dot{\Theta}$ and $\ddot{\Theta}$ respectively, is linked to the motor states by the transmission ratio $N(\Theta)$, τ_l is the load torque acting on the joint link. For the joints of the system which use Harmonic Drive gears the transmission ratio is constant and independent from the joint position. Consequently, the relation between the motor and link side variables become a linear relations:

$$N = \frac{\tau}{\tau_m} = \frac{\varphi}{\Theta} = \frac{\dot{\varphi}}{\dot{\Theta}} = \frac{\ddot{\varphi}}{\ddot{\Theta}}, \tag{4.21}$$

where τ is the actuation torque available on the link side after the joint transmission.

For the knee and ankle joints the transmission ratio is a non-linear function of the position of the joints Θ. The relation between motor side and link side variables can be written calculating the first and second time derivative as follows:

$$\varphi = \Lambda_i(\Theta), \tag{4.22}$$

$$\dot{\varphi} = \nabla_\Theta \Lambda_i(\Theta)\dot{\Theta}, \tag{4.23}$$

$$\ddot{\varphi} = \nabla^2_\Theta \Lambda_i(\Theta)\dot{\Theta}^2 + \nabla_\Theta \Lambda_i(\Theta)\ddot{\Theta}. \tag{4.24}$$

$\nabla_\Theta \Lambda_i$ is the first function derivative with respect the variable Θ, while $\nabla^2_\Theta \Lambda_i$ is the second function derivative. Figure 4.9 and Figure 4.10 illustrate the relation (4.22) for the right and left knee joints. In [20] and [118] the computation of the previous expressions is described. Since (4.22)-(4.24) are kinematic relations, their values can be precomputed offline and stored in look-up tables. In this way a fast availability of the values is guaranteed for the real-time control of the robot.

Applying the same approximation approach as in Section 4.4, the non-linear transmission ratio in (4.22) can be considered as the sum of a constant $\overline{N}_{im}$ and a configuration dependent term $\Delta\Lambda_i(\Theta)$:

$$N_i(\Theta) = \overline{N}_{im} + \Delta\Lambda_i(\Theta). \tag{4.25}$$

Neglecting the second term of the equation for limited variation of the angle Θ, N_i can be considered constant. In this case (4.21) can be also used to model the knee and ankle joint transmissions. Consequently, the matrix of the gear reduction ratios $\boldsymbol{N}$ becomes a constant diagonal matrix.

Considering, (4.21) the generalized joint model dynamic equations can then be written on the motor side of the joint[6] as a function of the variable φ:

$$(J_m + J_{lm})\ddot{\varphi} + b(\dot{\varphi}) = \tau_m + \tau_{lm}, \tag{4.26}$$

6 Note that the equations could have equivalently been rewritten on the link side of the joint as function of the variable Θ.

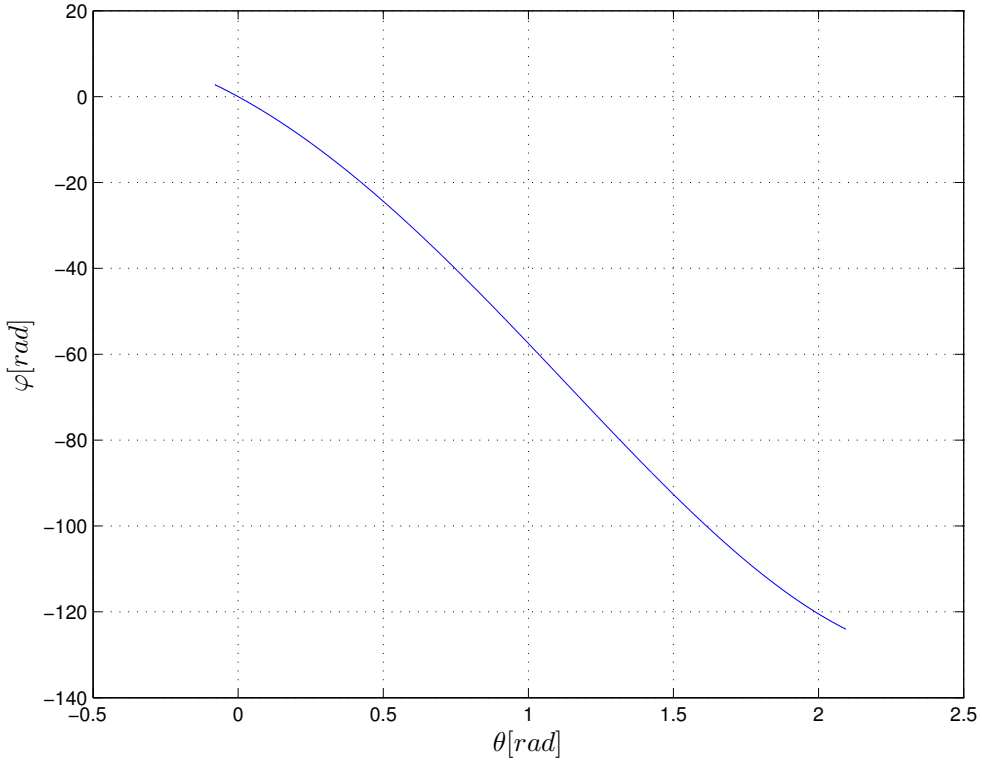


Figure 4.9: Relation between motor and link side angles of the right knee.

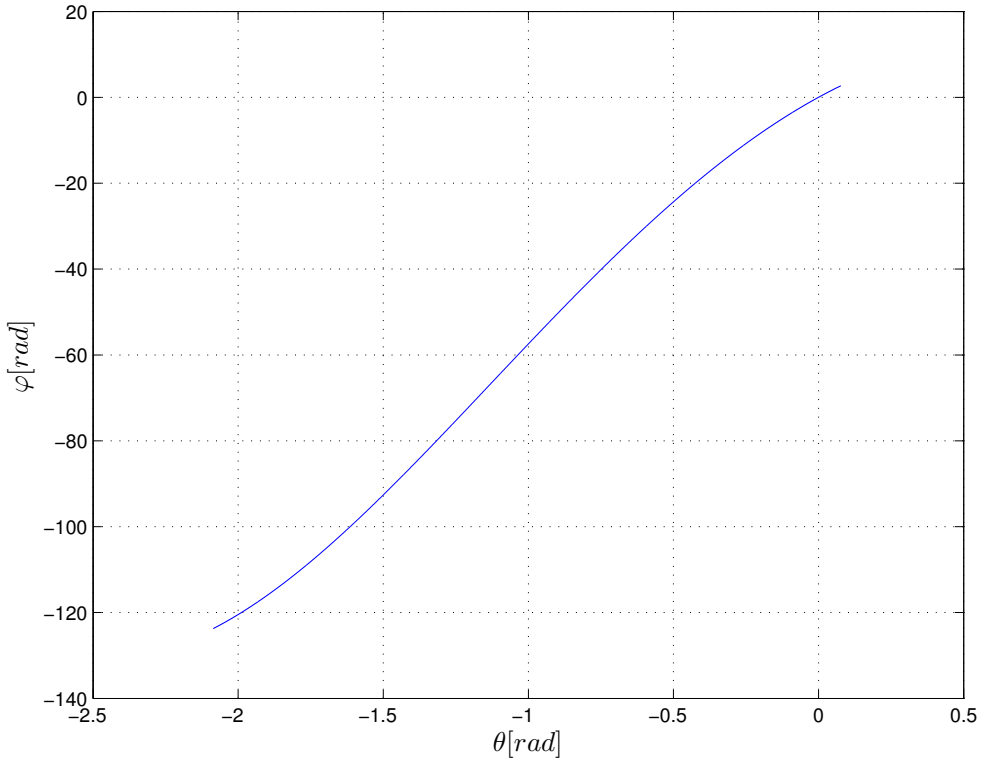


Figure 4.10: Relation between motor and link side angles of the left knee.

$$J_{lm} = \frac{J_l}{N^2}, \tag{4.27}$$

$$\tau_{lm} = \frac{\tau_l}{N}, \tag{4.28}$$

where J_{lm} is the load inertia reported on the motor side of the gear and τ_{lm} is the load torque also reported on the motor side of the joint.

4.5.2 Extended Model

In [20] the mechanical model of each joint of LOLA is described. The author proposes a model that defines the effects of friction and stiffness as concentrated parameters in the joint transmissions. This model is based on the catalog data available for those components and has been proved to deliver good results for the dynamic simulation of LOLA in comparison with the data collected during the experiment with the robot.

For the joint friction of the Harmonic Drive gears, Buschmann [20] proposes a modeling function dependent on both the motor shaft velocity and the load torque:

$$\tau_f(\dot{\varphi}, \tau_l) = -\operatorname{sgn}(\dot{\varphi})(\tau_{f0} + \mu \|\tau_l\|) - (b_v + \gamma \|\tau_l\|)\dot{\varphi}. \tag{4.29}$$

The model parameters τ_{f0}, no-load starting torque/no-load back driving torque, μ and γ were determined by least square fitting. τ_l is the load torque acting on the link side of the gear divided by the transmission ration N. This model has proven to fit well with the catalog data of the Harmonic Drive gears, improving the model proposed by Rossmann [114] and Löffler [86].

For the knee and ankle joints, the only available catalog data for the roller screws are the direct η_D and indirect η_I efficiency. Buschmann [20] uses the model (4.29) computing the equivalent friction torque parameters. The result is a function dependent from the motor torque and the efficiency η:

$$\tau_f(\dot{\varphi}, \tau_m) = -\operatorname{sgn}(\dot{\varphi}) \|(1-\eta)\tau_m)\|. \tag{4.30}$$

In the previous equation $\eta = \eta_D$ when the joint is actuated, i. e., $\varphi\tau_m > 0$, and $\eta = \eta_I$ when the roller screw is moved by an external force, i. e., $\varphi\tau_m < 0$.

To correctly model the joint behavior, another important parameter is its elasticity. Every real mechanical system presents some deformation when subject to forces or torques, which can be external forces or forces caused by the normal work of the system. The elasticity characteristics of a robot arm can be modeled as concentrated in the joint or distributed along the link. In [38] the authors discuss modeling methods and control techniques for the regulation and the tracking control of robotic arms.

The presence of flexible elements in the actuation train causes small but high frequency oscillations on the link side of the joint. This can lead to errors in the trajectory tracking and, therefore, errors in the positioning of the end effector. Having elastic elements in the actuation train leads to a mechanical resonance which limits the bandwidth of the control loop and consequently the value of the control gains. On the other side, taking into account the elasticity, leads to a more complex system model and a more complex control problem.

Table 4.2: Lola joint deformation while walking at 2.3 km/h.

Joint Name	$\Delta\Theta_{mean}$ [mrad]	$\Delta\Theta_{MAX}$ [mrad]
Pelvis rotation	0.085	2.069
Pelvis adduction	0.013	1.743
Hip rotation right	0.082	0.529
Hip adduction right	0.680	0.681
Hip flexion right	0.717	3.174
Knee flexion right	3.028	12.488
Ankle adduction right	-	-
Ankle flexion right	-	-
Toe flexion right	0.452	1.996
Hip rotation left	0.101	0.870
Hip adduction left	0.723	1.085
Hip flexion left	0.762	3.391
Knee flexion left	5.000	14.735
Ankle adduction left	-	-
Ankle flexion left	-	-
Toe flexion left	0.637	1.529
Arm flexion right	0.097	0.747
Arm adduction right	0.075	0.971
Elbow flexion right	0.129	0.756
Arm flexion left	0.081	1.664
Arm adduction left	0.074	0.429
Elbow flexion left	0.187	2.069

The number of state variables necessary to model a flexible joint is twice that for the rigid model if the flexibility is considered concentrated in the joint transmission. For the case of distributed flexibility along the link a continuous system must be considered. The latter is even more complex than in the previous case. Since LOLA has relative short and stiff links between the joints, the elasticity is considered concentrated in the transmission. Position, velocity and acceleration on the motor side are decoupled from the one on the link side and the relation (4.21) can not be used directly.

The possibility of measuring the position on the motor and link side of the transmission enables the quantification of the relative deformation of the joints. Table 4.2 shows the mean and maximum value of the joint deformation $\Delta\Theta = (\Theta - \varphi_l)$ (where φ_l is the motor position reported on the link side of the joint) for each joint of the robot equipped with a link side encoder[7]. The data was measured while the robot walks at a velocity of 2.3 km/h.

7 For the ankle joints no measurement is available. The link side encoder is sensitive to mechanical

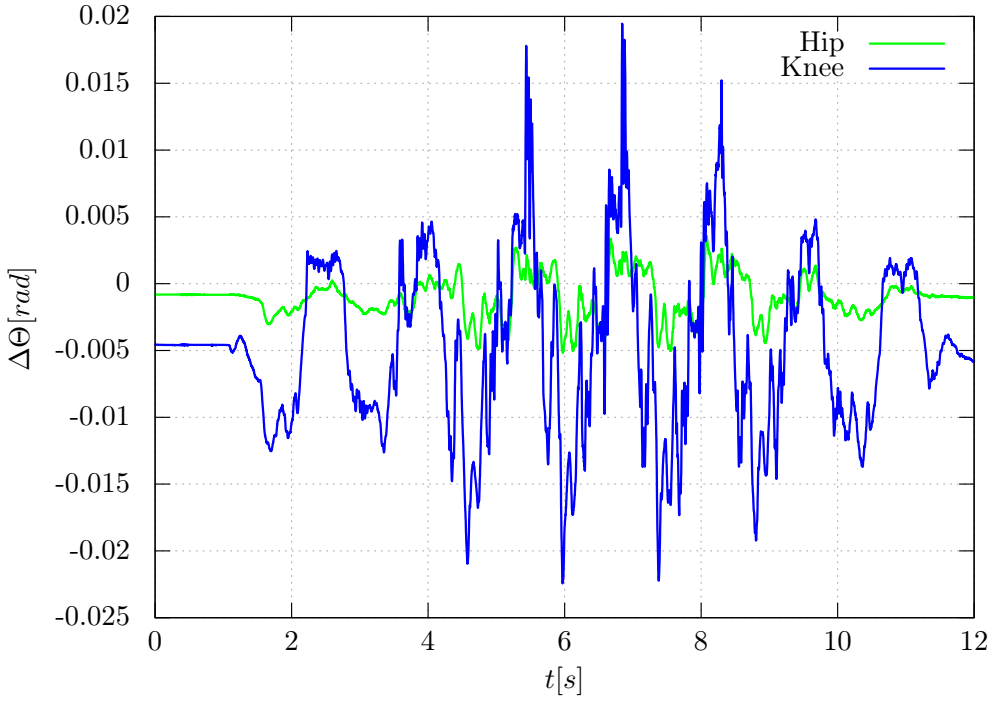


Figure 4.11: Deformation on knee and hip flexion joints (walking velocity 3.4 km/h).

As it can be seen, the maximum value of the deformation is around 1 mrad for almost all joints, except for the hip flexion and knee joints. In those cases the error between the motor and link side positions reaches values over 3 mrad for the hip flexion and 12 mrad for the knee. These indicates that this joints are more subject to deformation effects.

Figure 4.11 shows the deformation of the right hip and right knee joints when the robot is walking at 3.6 km/s. As can be seen the deformation increases for both the joint in comparison with the previous experiment. The maximum difference between motor and link side position is 4 mrad for the hip and 20 mrad for the knee.

The joint stiffness model proposed by Buschmann [20] for the Harmonic Drives gears is described as a progressive and piecewise linear torque/torsion function [85]:

$$\tau_K = \begin{cases} K\Delta\Theta & \text{for } \tau_k \leq \tau_{e1} \\ \tau_{e1} + K_1\Delta\Theta & \text{for } \tau_{e1} \geq \tau_k \leq \tau_{e2} \\ \tau_{e2} + K_2\Delta\Theta & \text{for } \tau_k > \tau_{e2} \end{cases} \tag{4.31}$$

τ_k is the elasticity torque, $\Delta\Theta$ is the gear deformation, τ_{e1} and τ_{e2} are the limit torques of the low to middle and middle to high torque range for the Harmonic Drive gears, K, K_1 and K_2 are the equivalent stiffness parameters also for the low, middle and high torque range. The values of the limit torques and equivalent stiffness for the Harmonic Drive gears are listed in Table E.1, Appendix E. Figure 4.12 shows the schematic diagram of the elastic joint, where the joint stiffness K, modeled as a torsional spring, is added on the motor side

deflection of the inductive measuring disk. For these joints the deformation of the support structure of the disk causes the sensor to fail.

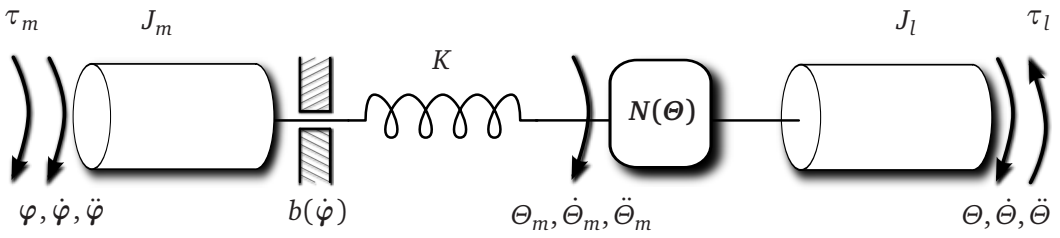


Figure 4.12: Model for an elastic joint.

of the joint. Θ_m, $\dot{\Theta}_m$ and $\ddot{\Theta}_m$ are the link position, velocity and acceleration respectively, reported on the motor side. The dynamic model of the elastic link can then be written on the motor side of the gear as:

$$J_{lm}\ddot{\Theta}_m + K(\Theta_m - \varphi) = \tau_{lm} \tag{4.32}$$

$$J_m\ddot{\varphi} + b(\dot{\varphi}) + K(\varphi - \Theta_m) = \tau_m, \tag{4.33}$$

$$J_{lm} = \frac{J_l}{N^2}, \tag{4.34}$$

$$\tau_{lm} = \frac{\tau_l}{N}, \tag{4.35}$$

$$N = \frac{\Theta_m}{\Theta} = \frac{\dot{\Theta}_m}{\dot{\Theta}} = \frac{\ddot{\Theta}_m}{\ddot{\Theta}}. \tag{4.36}$$

Here, J_{lm} is the link inertia reported on the motor side and τ_{lm} is the disturbance torque also considered on the motor side of the joint. Instead of the model (4.31) the joint elasticity is modeled as a torsional spring with constant stiffness K.

For the knee joint, no information is available on the stiffness of the spindles. As it is shown in Figure 4.11 this component shows the largest deformation in comparison with the other joints. To correctly model the behavior of the knee joint, its stiffness function is needed. Without any catalog data, the parameters must be determined by an estimation using experimental measurements.

In the next section the joint parameter estimation for the knee is discussed. The stiffness and friction parameters for the other joints are also computed to confirm the values from the catalog data.

4.6 Joint Parameter Identification

The challenge of the parameter estimation in dynamic systems has been addressed many times. The determination of real system parameters is very important for the validation of system models and the design of control laws.

The classical problem is the identification of the inertial parameters of a stiff robots. The procedure is generally based on *Least Square Estimation* (LSE) using measurement data from torque, position, velocity and acceleration as described in Siciliano et al. [125] and Hollerbach et al. [70], while in Christensen and Hager [32] the sensor fusion and estimation methods are considered in a more extended view of the robot perception.

The estimation of other joint parameter like friction and stiffness has also been widely discussed in the literature. In Pham et al. [111] the authors describe a parameter identification method based on LSE with emphasis on correct data filtering before computing the estimation algorithm to avoid bias in the results due to noisy measurement. Albu-Schäffer and Hirzinger [12] obtain good results in the identification of the friction and elastic elements of a 7 DoF robot using motor and link side position measurement and link torque sensors. The inertia parameters of the robot are computed using the CAD data of the robot. Gautier et al. [55] present a method to estimate the friction and stiffness parameters of a single DoF robot using only motor torque measurement and then minimizing the error between measurement and simulated data. In Flacco et al. [51] the authors use motor position and velocity data to estimate nonlinear stiffness characteristics using a residual model based estimation and a black-box model of the stiffness. Flacco et al. [50] a method based on a combination of a residual-based stiffness estimator and Kalman filter using motor side position information is presented. In Tuttle [141], Taghirad et al. [133] and Kircanski et al. [80] detailed identification of stiffness and friction for Harmonic Drive gears with satisfactory experimental results are presented. In Carbone et al. [30] the authors analyze the stiffness property of the humanoid Robot WABIAN-RIV from a purely theoretical point of view to estimate the stiffness property of the robot's joints.

Detailed methods to specifically estimate the friction characteristic of an industrial robots addressing also the control problem have been published. In Le Tien et al. [82] the authors propose a friction observer based on the measurements of a torque sensor on the link side of the joint gear. In Grotjahn and Heimann [59] and Grotjahn et al. [60] a two step method with weighted least squares estimation to determine the friction parameters for an industrial robot is defined. In Thuemmel and Rossner [139] and Thuemmel et al. [140], the authors describe an identification method for joint friction for a generic mechanisms based on iterative parameter estimation using simulation and measurements.

Tuttle [141], Swevers et al. [132] and Otani and Kakizaki [104] among others, address the problem of how to properly excite the robot joint. They use optimized trajectories to properly estimate the joint parameter in presence of measurement noise and actuation disturbances.

In the case of LOLA, experiments have shown that the inertial parameters are sufficiently well know from the CAD model of the robot. Also, the kinematic and dynamic model have been proved to deliver good performance in comparison with the experiments as shown in Buschmann [20]. Nevertheless, to improve the precision of the full multibody simulation and to enable a model based design of the robot joint control, an experimental estimation of the friction and stiffness parameters of the joints is desirable, especially for the hip flexion and knee joints of the robot, which are shown to have the biggest tracking error on the link side of the joint.

4.6.1 Parameter Identification — Method

In Crassidis and Junkins [35] and Simon [126] a good introduction to estimation methods, from the classic linear LSE to more advanced non-linear algorithms, is presented. In this work, the recursive LSE with forgetting factor method is used. The LSE problem can be stated as follows: having a known regression matrix $\boldsymbol{H}$, with the system output measurement $\boldsymbol{y}$, the estimation $\hat{\boldsymbol{\chi}}$ of the constant parameter vector $\boldsymbol{\chi}$ must be determined.

The relation between the system output and its model can be stated as:

$$\boldsymbol{y} = \boldsymbol{H}\boldsymbol{\chi}, \tag{4.37}$$

$$\boldsymbol{e} = \boldsymbol{H}\hat{\boldsymbol{\chi}} - \boldsymbol{y}. \tag{4.38}$$

In (4.37) it is assumed that the parameters in the vector $\boldsymbol{\chi}$ have a linear relation to the output vector $\boldsymbol{y}$ which is described by the regression matrix $\boldsymbol{H}$.

Defining the residual error, due to measurement noise and modeling errors, between the measurements and the system model estimation as in (4.38) the optimal solution to the problem, originally proposed by Gauss [54], is the minimization of the following function of the residual error:

$$\boldsymbol{J} = \frac{1}{2}\boldsymbol{e}^T\boldsymbol{e}. \tag{4.39}$$

Computing the derivative of (4.39) with respect the estimated parameter vector $\hat{\boldsymbol{\chi}}$, a necessary and a sufficient condition for the minimization problem solution can be found:

$$\nabla_{\hat{\chi}}\boldsymbol{J} = \frac{\partial \boldsymbol{J}}{\partial \hat{\boldsymbol{\chi}}} = \boldsymbol{H}^T\boldsymbol{H}\hat{\boldsymbol{\chi}} - \boldsymbol{H}^T\bar{\boldsymbol{y}}, \tag{4.40}$$

$$\nabla^2_{\hat{\chi}}\boldsymbol{J} = \frac{\partial^2 \boldsymbol{J}}{\partial \hat{\boldsymbol{\chi}}\,\partial \hat{\boldsymbol{\chi}}} = \boldsymbol{H}^T\boldsymbol{H} \succeq 0. \tag{4.41}$$

Setting the Jacobian $\nabla_{\hat{\chi}}\boldsymbol{J}$ of (4.39) equal to zero (necessary condition) and if its Hessian (4.41) is positive semi-defined (sufficient condition) the minimization problem can be solved from (4.40) and the estimation vector $\hat{\boldsymbol{\chi}}$ can be directly computed as:

$$\hat{\boldsymbol{\chi}} = (\boldsymbol{H}^T\boldsymbol{H})^{-1}\boldsymbol{H}^T\boldsymbol{y}. \tag{4.42}$$

The assumption for the previous method is that all the elements of the measurement vector $\boldsymbol{y}$ are available at the moment of the computation. This is true if the estimation is performed off-line after the conclusion of the experiment. If the estimation must be performed during the experiment, i. e., on-line, the previous LSE method can be rewritten in recursive form (RLSE). In this case only one set of measurements is needed at each computation step k.

The RLSE algorithm can be stated as follows[8]:

$$\boldsymbol{K}_k = \boldsymbol{P}_{k-1}\boldsymbol{H}^T{}_k[\boldsymbol{H}_k\boldsymbol{P}_{k-1}\boldsymbol{H}^T{}_k + \boldsymbol{R}]^{-1}, \tag{4.43}$$

$$\boldsymbol{P}_k = [\boldsymbol{I} - \boldsymbol{K}_k\boldsymbol{H}_k]\boldsymbol{P}_{k-1}, \tag{4.44}$$

$$\hat{\boldsymbol{\chi}}_k = \hat{\boldsymbol{\chi}}_{k-1} + \boldsymbol{K}_k(\boldsymbol{y}_k - \boldsymbol{H}_k\hat{\boldsymbol{\chi}}_{k-1}). \tag{4.45}$$

Where:

- $\hat{\boldsymbol{\chi}}_{k-1}$: is the previous step estimation of $\boldsymbol{\chi}$.
- $\hat{\boldsymbol{\chi}}_k$: is the current step estimation of $\boldsymbol{\chi}$.

8 A complete derivation of the RLSE algorithm from LSE can be found in Crassidis and Junkins [35] and Simon [126].

- $\boldsymbol{y}_k$: is the current step measurement vector.
- $\boldsymbol{H}_k$: is the current step system parameter matrix.
- $\boldsymbol{K}_k$: is the estimator gain.
- $\boldsymbol{P}_{k-1}$: is the previous step covariance matrix.
- $\boldsymbol{P}_k$: is the current step covariance matrix.
- $\boldsymbol{R}$: is the measurements noise covariance matrix.
- $\boldsymbol{I}$: is the identity matrix.

To initialize (4.43)-(4.45) at the first step, Crassidis and Junkins [35] suggest to use an a priori estimate of the vector $\hat{\boldsymbol{\chi}}$ and the covariance matrix or to compute them using the first measurement ($k = 1$) vector as follows:

$$\boldsymbol{P}_1 = [\frac{1}{\alpha^2}\boldsymbol{I} + \boldsymbol{H}^T{}_1\boldsymbol{R}_1\boldsymbol{H}_1]^{-1}, \tag{4.46}$$

$$\hat{\boldsymbol{\chi}}_1 = \boldsymbol{P}_1[\frac{1}{\alpha}\beta + \boldsymbol{H}^T{}_1\bar{\boldsymbol{y}}], \tag{4.47}$$

$$\alpha \gg 1, \tag{4.48}$$

$$\beta \ll 1. \tag{4.49}$$

The RLSE algorithm is also a suitable choice for off-line implementation. The evolution of the error, covariance and estimator gain can be saved for post computation analysis purposes.

In both LSE and RLSE algorithms, it is assumed that the vector $\boldsymbol{\chi}$ to be estimated is constant. In case of a slowly changing vector, this assumption can be relaxed introducing a suitable *Forgetting Factor* γ in (4.44) (see [71, 116]):

$$\boldsymbol{P}_k = \gamma^{-1}[\boldsymbol{I} - \boldsymbol{K}_k\boldsymbol{H}_k]\boldsymbol{P}_{k-1}, \tag{4.50}$$

$$0 < \gamma \leq 1. \tag{4.51}$$

If $\gamma = 1$ the normal RLSE as stated in (4.43)-(4.45) is obtained. When $\gamma < 1$ the effect of older data is limited for the current computation of the covariance matrix P_k reducing the averaging trough the complete measurement set. Normally, the value of γ used in practice are between 0.99 and 0.95 for slow or very fast variation of the estimation of $\boldsymbol{\chi}$. The RLSE algorithm can be also used for a non-linear model as long as the parameters to be estimated are linear with the functions used in the regression matrix $\boldsymbol{H}$.

4.6.2 Parameter Identification — Implementation

To estimate the required joint parameters for the extended joint model of LOLA applying the RLSE method with forgetting factor, the model (4.32)-(4.33) must be rewritten in regression matrix form as in (4.37). The model parameters that will be estimated for the two hip and two knee joints are the friction function $b(\dot{\varphi})$ and the joint stiffness k. The identification of the stiffness is the major goal. The inertia parameters of the motors and

link side of the gear are assumed to be known from the CAD data of the robot. For the hip joints the estimation is carried out to prove the correctness of the model based on the catalog data, while for the knee those parameters are unknown.

The measurements available for the estimation are: motor and link position, motor velocity and motor current. The motor constant value k_τ is considered as known from the motor catalog data. With this assumption, the motor torque can be directly computed from (4.5). The sampling time of the data is equal to the position control loop $T_s = 1.5ms$. The link velocity, motor and link acceleration are also needed. They can be computed from link position and motor and link velocity respectively, using the central difference derivative (3.8) for each sampling point k of the measurement:

$$\dot{\Theta}_k = \frac{\Theta_{k+1} - \Theta_{k-1}}{2T_s}, \tag{4.52}$$

$$\ddot{\Theta}_k = \frac{\dot{\Theta}_{k+1} - \dot{\Theta}_{k-1}}{2T_s}, \tag{4.53}$$

$$\ddot{\varphi}_k = \frac{\dot{\varphi}_{k+1} - \dot{\varphi}_{k-1}}{2T_s}, \tag{4.54}$$

where the indexes $k-1$, k and $k+1$ indicate the previous, current and next measured position and velocity respectively.

All the input data are filtered with a first order Butterworth low pass filter with a cut-off frequency of 30 Hz, which has two positive effects: it limits the sensor noises amplification effect of the discrete differentiation, i. e., for the velocity and acceleration signal and set the same phase lag for the complete input vector of the estimation system ensuring its time coherence. Because of the need of discrete differentiation, the availability of the velocity and acceleration data are shifted of one and two sampling period with respect to the position measured data. To maintain the signals time synchronization, the estimation is then started after two sampling periods.

Considering the measurement data available and the signals that can be computed from them, the joint model presented in the previous section must be adapted accordingly. The friction function (4.29) has two components which depend on the load torque of the link. Having no possibility to directly measure this quantity those term are neglected in the regression matrix. With this simplification (4.29) reduces to:

$$b(\dot{\varphi}) = \mathrm{sgn}(\dot{\varphi})\tau_{f0} + b_v\dot{\varphi}. \tag{4.55}$$

The parameters that must be identified are then the viscous friction coefficient b_v and the static friction coefficient τ_{f0}. This model is used for both the hip and knee joints.

Regarding the excitation trajectories of the joints, some consideration must be made. The estimation must be performed directly on the robot while it is walking. The normal approach used in the literature for industrial robots is to move each joint of the robot one at a time using trajectories optimized for parameter excitation (see [12, 59, 66]). Nevertheless, the industrial robots do normally have an integrated mechanical brakes for every joint. This is not the case for Lola for the reasons explained in Chapter 2. Therefore, the excitation trajectories used for the parameter identification are the normal walk trajectories of the robot. In Figure 4.13 the discrete Fourier transformation (DFT) of

the current signal for the knee motor is shown. The signal has 5 distinct major frequency components, which are sufficient to excite the system for the parameter identification[9]. Having a direct measure of the input signals, the effect of the closed loop feedback control can be neglected.

Another consideration to be made regards whether the model should be written on the motor or link side of the joint. The first choice seems to be the more promising for the fact that all friction and elastic components are concentrated only at one side of the joint and the link side is considered as a pure inertial load. This is particularly useful in case the estimation is performed on-line during the experiments. For the joints with variable gear transformation ratio, only the link inertia must be reported on the motor side of the joint.

To model the effect of the load torque τ_{lm} and of the unmodeled non-linear effects of the disturbance d, the parameter vector $\boldsymbol{\chi}$ is augmented with an element d_e.

Finally, with all the previous assumption the joint model (4.32)-(4.33) can be rewritten in regression form:

$$\begin{bmatrix} \tau_m - J_m\ddot{\varphi} \\ -J_{lm}\ddot{\Theta}_m \end{bmatrix} = \begin{bmatrix} -\Theta & \varphi & \dot{\varphi} & \operatorname{sgn}(\dot{\varphi}) & 0 \\ \Theta & -\varphi & 0 & 0 & c_d \end{bmatrix} \begin{bmatrix} K \\ K \\ b_v \\ \tau_{f0} \\ d_e \end{bmatrix}, \tag{4.56}$$

$$\boldsymbol{y} = \boldsymbol{H}\boldsymbol{\chi}. \tag{4.57}$$

All known quantities are grouped in the input vector $\boldsymbol{y}$: the input motor torque τ_m and torques due to inertial effects $J_m\ddot{\varphi}$ and $J_{lm}\ddot{\Theta}$. The parameters that must be identified are grouped in the vector $\boldsymbol{\chi}$: the stiffness K, the viscous friction b_v, the static friction τ_{f0} and an estimation of the system disturbance torque d. The components of the regression matrix $\boldsymbol{H}$ are the measured values of the motor position and velocity φ, $\dot{\varphi}$ and $\operatorname{sgn}(\dot{\varphi})$, the joint position Θ_m reported to the motor side and the disturbance constant $c_d = 1$.

4.6.3 Parameter Identification — Results

To test the previously introduced estimation method, a set of experiment at different walking velocities are performed. The robot walks in forward direction for 5 m. Each experiment is characterized by different combinations of maximum and minimum step length and step time. The step length defines how long is each step taken by the robot. The step time defines how fast the step must be performed.

The experiment is divided in three phases: acceleration, walk at maximum speed and deceleration. In the first phase, the robot starts walking from the steady state with the desired minimum step length and step time and increases the step length until the desired maximum is reached. In the second phase, the robot performs one or two steps at the maximum step length depending on the maximum defined speed. In the third phase, the robot decelerate to steady state. In Table 4.3 the parameters set for each experiment are shown.

For each run of the RLSE algorithm, the parameter estimation converged to stable values. The results are listed in Table 4.4 for the joint stiffness, Table 4.5 for the viscous friction

9 See [71, 116] for the concept of sufficiently rich input signal.

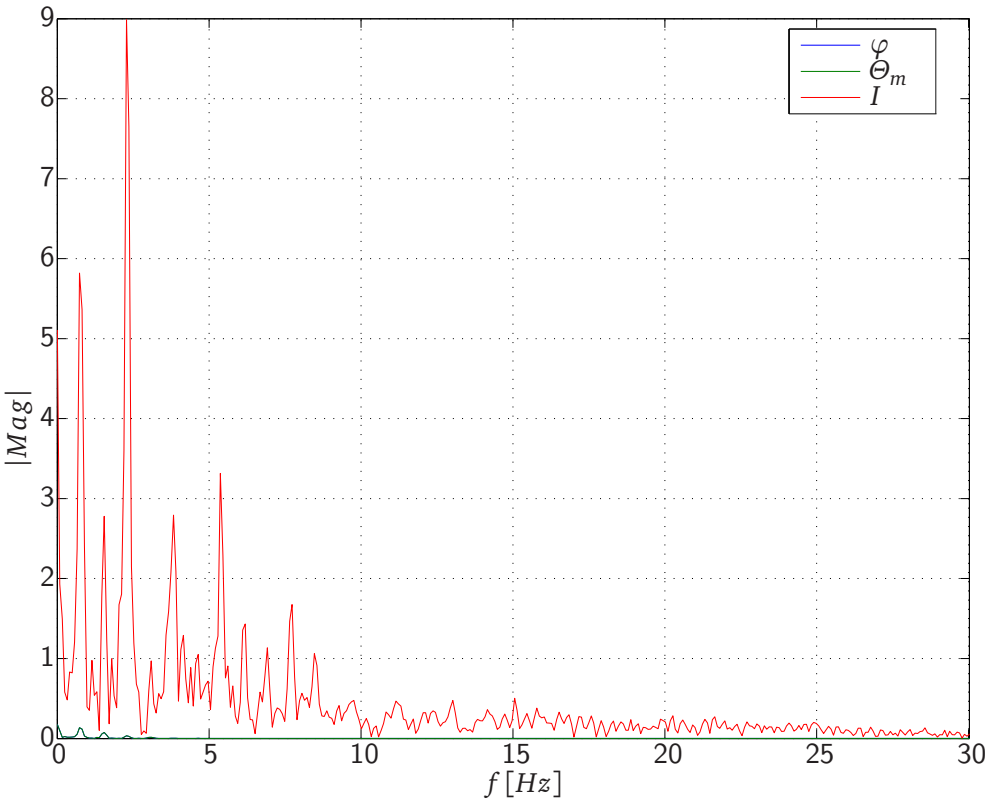


Figure 4.13: Discrete Fourier Transformation of the knee motor current (I), joint (Θ_m) and motor (φ) position (walking velocity 3.6 km/h).

Table 4.3: Joint model estimation experiment parameters.

Experiment	Minimum Step Length [m]	Maximum Step Length [m]	Step Time [s]	Maximum Speed [km/h]
Exp. 1	0.3	0.5	0.8	2.25
Exp. 2	0.4	0.6	0.8	2.7
Exp. 3	0.4	0.6	0.7	3.09
Exp. 4	0.45	0.65	0.7	3.34
Exp. 5	0.45	0.65	0.65	3.6

and Table 4.6 for the static friction. The mean value over the five experiment is computed in order to compare the experimental values with the model parameters based on catalog data. As can be seen in Table 4.7, the results show that the hip joints have a lower stiffness parameter value and the friction parameter have a higher value with respect to the catalog data.

The higher compliance of the joint are due to the mechanic construction, bearings and assembling of the joint. Moreover, a direct relation between the stiffness variation and the joint deformation could not be established, suggesting that the stiffness of the joint is also

Table 4.4: Estimated stiffness.

Experiment	Right Hip [Nm/rad]	Right Knee [Nm/rad]	Left Hip [Nm/rad]	Left Knee [Nm/rad]
Exp. 1	14.932	3.138	15.457	2.808
Exp. 2	18.097	3.316	16.140	3.176
Exp. 3	19.892	3.890	17.316	3.274
Exp. 4	20.794	5.307	17.600	3.779
Exp. 5	20.184	3.697	17.059	3.623
Mean	18.780	3.860	16.714	3.332

Table 4.5: Estimated viscous friction.

Experiment	Right Hip [Nms/rad]	Right Knee [Nms/rad]	Left Hip [Nms/rad]	Left Knee [Nms/rad]
Exp. 1	$9.294 \cdot 10^{-3}$	$0.253 \cdot 10^{-3}$	$6.255 \cdot 10^{-3}$	$1.266 \cdot 10^{-3}$
Exp. 2	$7.809 \cdot 10^{-3}$	$1.452 \cdot 10^{-3}$	$5.936 \cdot 10^{-3}$	$1.422 \cdot 10^{-3}$
Exp. 3	$8.356 \cdot 10^{-3}$	$1.011 \cdot 10^{-3}$	$5.371 \cdot 10^{-3}$	$1.771 \cdot 10^{-3}$
Exp. 4	$7.886 \cdot 10^{-3}$	$1.970 \cdot 10^{-3}$	$5.156 \cdot 10^{-3}$	$1.643 \cdot 10^{-3}$
Exp. 5	$7.469 \cdot 10^{-3}$	$1.389 \cdot 10^{-3}$	$4.373 \cdot 10^{-3}$	$2.353 \cdot 10^{-3}$
Mean	$8.163 \cdot 10^{-3}$	$1.215 \cdot 10^{-3}$	$5.418 \cdot 10^{-3}$	$1.691 \cdot 10^{-3}$

highly influenced by the load torque on the output of the gear.

Regarding the friction parameters, the higher values can be explained considering that the estimated model does not consider the friction terms related to the load torque. Therefore, the effect of the neglected components merge into the remaining two terms of the estimation model.

To verify the validity of the estimated parameters, the multibody simulation of LOLA is used. The walking experiments are reproduced in the simulation using the estimated parameter values instead of the catalog data. Comparing the measured joint positions and velocities at each walking speed showed a close match between the experiment and the simulated data. Figure 4.14-Figure 4.15 and Figure 4.16-Figure 4.17 show the joint deformation of the right hip and knee joint at the walking velocity of 3.0 km/h and 3.6 km/h respectively. As can be seen the deformation in the simulation shows a good match with the experiment especially for the knee joint while the robot is walking.

Table 4.6: Estimated static friction.

Experiment	Right Hip [Nm]	Right Knee [Nm]	Left Hip [Nm]	Left Knee [Nm]
Exp. 1	0.264	0.377	0.366	0.290
Exp. 2	0.378	0.393	0.348	0.310
Exp. 3	0.291	0.360	0.452	0.331
Exp. 4	0.682	0.379	0.762	0.405
Exp. 5	0.452	0.416	0.523	0.324
Mean	0.413	0.385	0.490	0.332

Table 4.7: Estimated joint parameters and catalog data joint parameters.

Joint	K		b_v		τ_{f0}	
	Estimated	Model	Estimated	Model	Estimated	Model
Hip Right	18.780	22	$8.163 \cdot 10^{-3}$	$0.152 \cdot 10^{-3}$	0.413	0.246
Hip Left	16.714	22	$5.418 \cdot 10^{-3}$	$0.152 \cdot 10^{-3}$	0.490	0.246
Knee Right	3.860	–	$1.215 \cdot 10^{-3}$	–	0.385	–
Knee Left	3.332	–	$1.691 \cdot 10^{-3}$	–	0.332	–

4.7 Chapter summary

In this chapter the structure of the control system of LOLA is presented. The HLC and LLC are introduced and their role for the motion generation, stabilization of the robot and trajectory tracking are discussed. The full multibody simulation of LOLA is augmented with a model of the actuation train with the result of using the same control gain implemented in the real robot also in the simulation program.

A model for the robot joint is presented for the stiff and elastic case. The latter is a concentrated parameter model of the joint stiffness and friction. Based on experimental measurements, the model parameters are estimated using the RLSE algorithm. A comparison between the measured and simulated deformation for the validation of the estimated parameters is also shown.

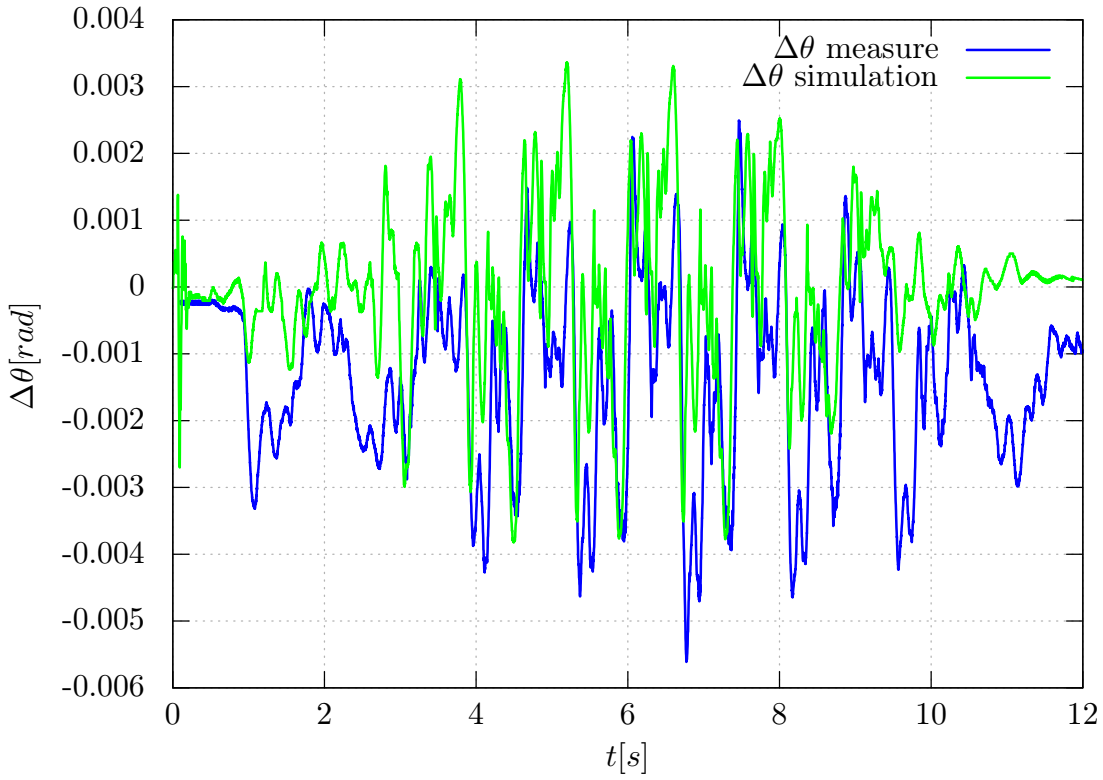


Figure 4.14: Comparison between measured and simulated joint deformation for the right hip at 3.0 km/h.

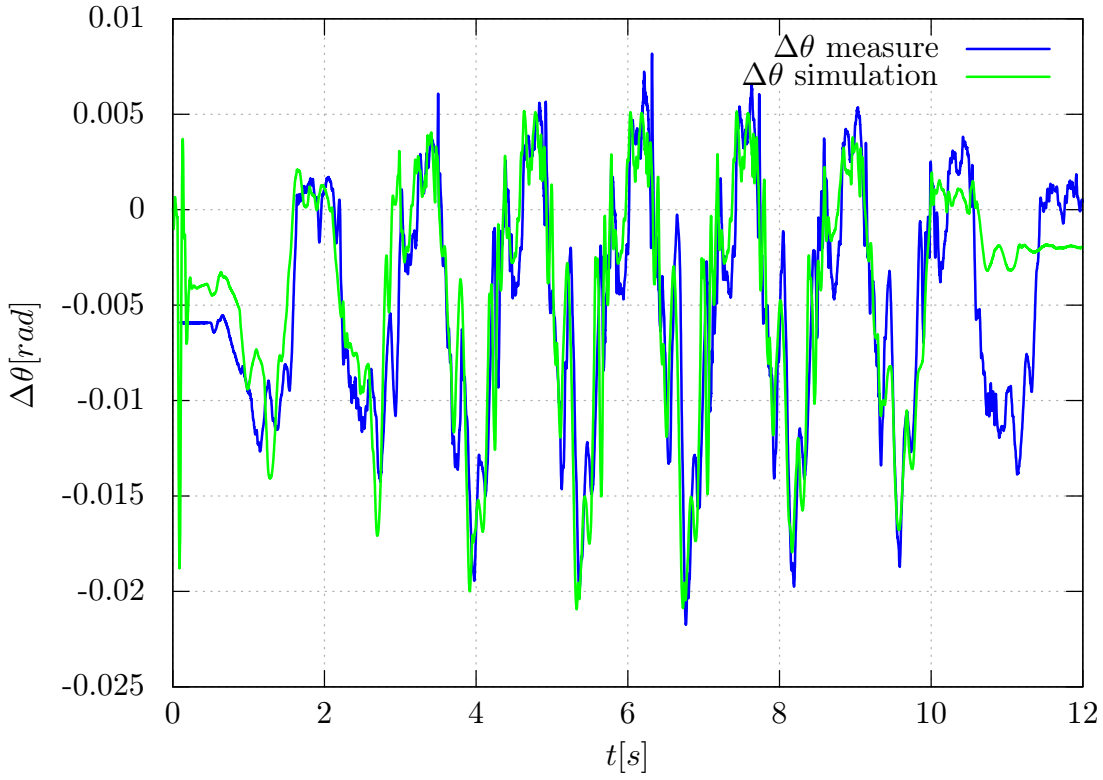


Figure 4.15: Comparison between measured and simulated joint deformation for the right knee at 3.0 km/h.

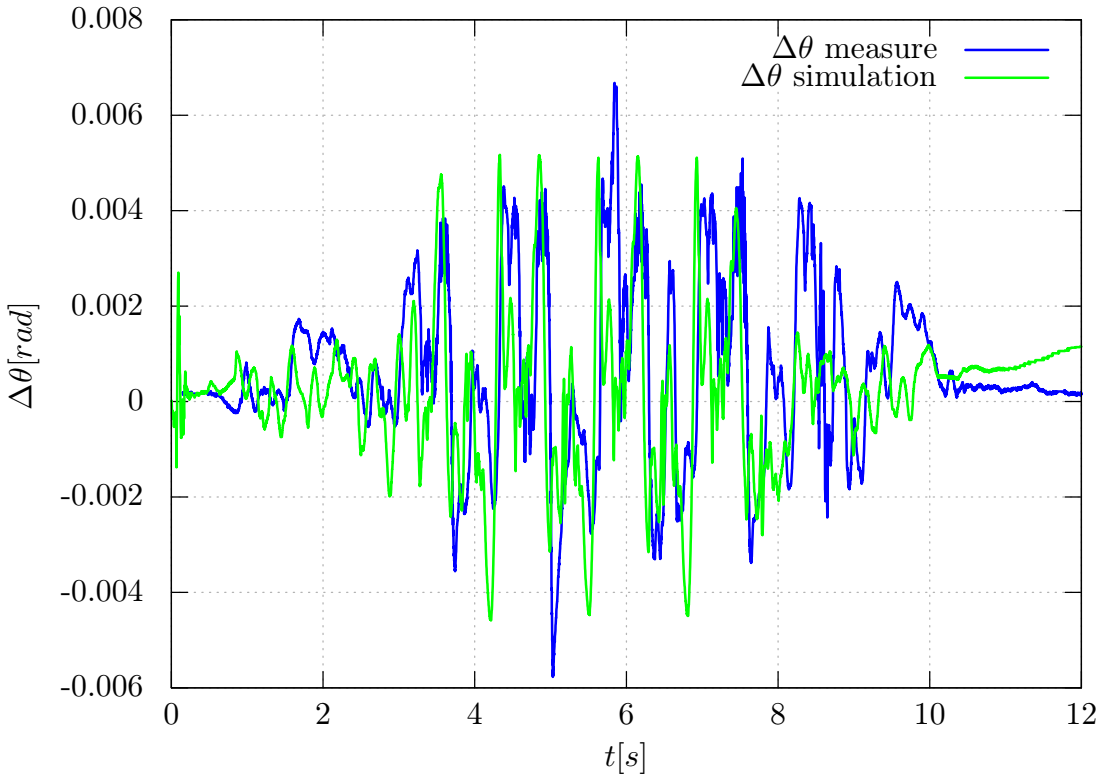


Figure 4.16: Comparison between measured and simulated joint deformation for the left hip at 3.6 km/h.

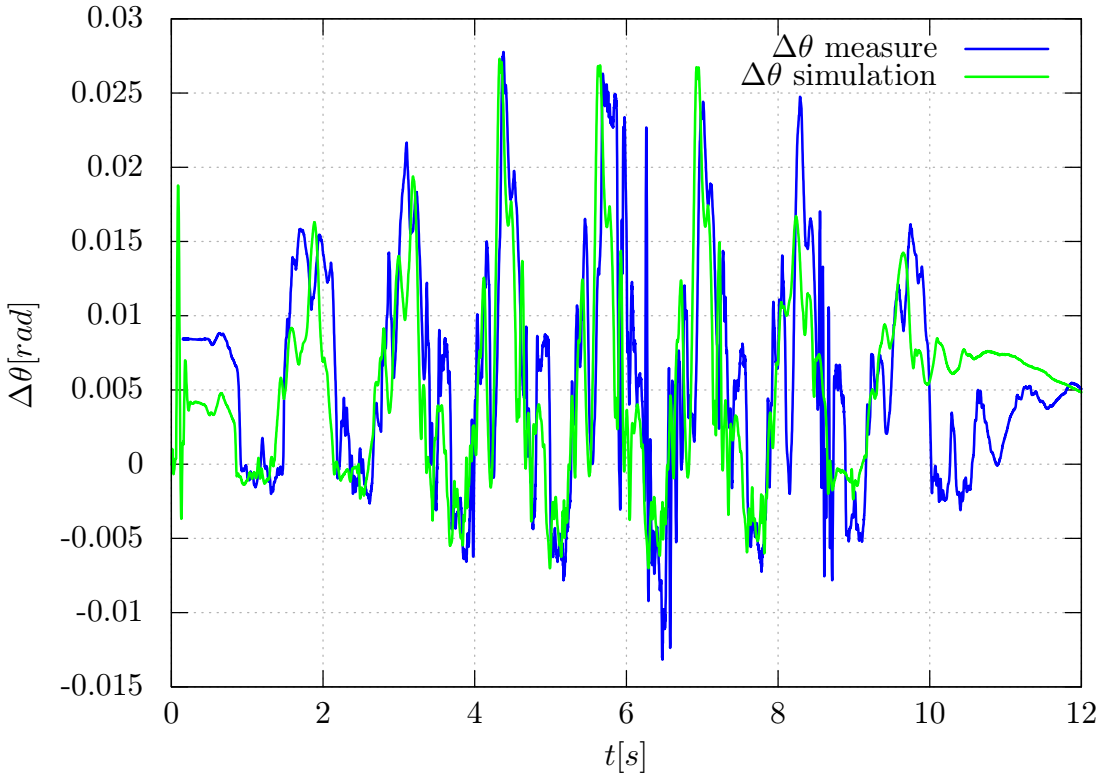


Figure 4.17: Comparison between measured and simulated joint deformation for the left knee at 3.6 km/h.

5 Advanced Joint Control

5.1 Introduction

The main focus of this chapter is the discussion and presentation of different joint control methods for humanoid robots in the presence of gear elasticity. This can be led back to the problem of controlling a two mass oscillating system with only one control input. The control of flexible robotic joints for industrial robots has been addressed by researchers since the early 1980's.

In [18, 107] a good survey on the modeling approaches and control techniques for flexible robotic joint is presented. Some of the early works on flexibility in robot joints are a series of papers addressing non-linear approaches to the solution of the problem (see [94, 129]) and the application of singular perturbation theory (see [130]). The latter approach has been developed over the years. The main result is the separation of the flexible system in two parts with different time scales, a *fast dynamics* part and a *slow dynamics* one. They can be in some way connected to the motor and link side dynamic. Different approaches have been proposed to stabilize these two time scale models. The majority of them tend to use a PD controller to asymptotically stabilize the fast time scale part of the system. To stabilize the slow time scale dynamic feedback linearization, Lyapunov based methods (see Al Ashoor et al. [11]), adaptive methods (see Ghorbel et al. [57]) and neural network (see Zeman et al. [152]) methods have been proposed.

Slotine and Li [127] present an adaptive tracking controller showing good experimental results also for high speed operations. Lozano and Brogliato [91] also use a stable adaptive controller without any assumption on the a priori knowledge of the joint flexibility. A comparison of different control laws for flexible joint of the humanoid robot RH-2 can be found in Villagra and Balaguer [145]. In Lee et al. [83] the authors discuss methods to identify the compliance in the joints of humanoid robots and on how to integrate these parameters in the simulation of a robotic system.

In Moberg [97] the identification and control of industrial robots with flexible joints is presented with extensive discussion on different control methods from the classic PD controller to computed torque and feedback linearization. Feedback linearization has been proposed also in De Luca [37], De Luca and Lucibello [39] and De Luca and Book [38]. In Albu-Schäffer and Hirzinger [12] and Le Tien et al. [82] the authors present a control method based on the passivity principle for a lightweight industrial robot with torque measurements on the link side.

The control of robots considering joints flexibility is a large research field and in the literature many different approaches have been proposed as can be seen from the previous brief discussion. While the vast majority of the articles regards industrial robot because of their importance in the industrial automation, their results can partially be extended also for humanoid robots.

Almost all the humanoid robots which have been developed over the years use different implementations of decentralized joint controllers (see [151], [77], [13], [105] and [40]).

Table 5.1: Cascaded PID gain stability conditions.

Measure		K_{pp}	K_{pv}	**Stable**
Link position and velocity	$(\Theta\ \dot{\Theta})$	/	/	No
Link position and Motor velocity	$(\Theta\ \dot{\varphi})$	$< K$	> 0	Limited
Motor position and velocity	$(\varphi\ \dot{\varphi})$	> 0	> 0	Yes

For the joint trajectories tracking the most used control scheme is the PID for its simplicity of tuning and the good performance if high gear ratios are used in the joints to limit the non-linear coupling effects and load torque disturbances.

In his chapter different joint control algorithms are discussed, based on the model and results of Chapter 4. Keeping in mind that the control laws must be implemented on the DSCBs or on the CCU, some of the basic methods are investigated in this thesis.

5.2 Cascaded PID Controller

The first discussed control law is the cascaded PID controller introduced in Chapter 4. This classic general method is extensively discussed in Siciliano et al. [125] for the rigid robot case. Considering the system (4.32)-(4.33), with constant stiffness K and neglecting the static friction τ_{f0}, the joint model can be written as follows:

$$J_{lm}\ddot{\Theta}_m + K(\Theta_m - \varphi) = \tau_{lm} \tag{5.1}$$

$$J_m\ddot{\varphi} + b_v\dot{\varphi} + K(\varphi - \Theta_m) = \tau_m. \tag{5.2}$$

Under these assumptions, the system model is linear and can be rewritten in the frequency domain using the Laplace transformation. Defining $s = \jmath\omega$, the open loop input/output transfer functions of the system are:

$$\frac{\Theta_m}{\tau_m} = \frac{1}{J_m J_{lm}} \frac{K}{(s^3 + \frac{b_v}{J_m}s^2 + K\frac{J_m+J_{lm}}{J_m J_{lm}}s + \frac{b_v K}{J_m J_{lm}})s}, \tag{5.3}$$

$$\frac{\varphi}{\tau_m} = \frac{1}{J_m J_{lm}} \frac{(J_{lm}s^2 + K)}{(s^3 + \frac{b_v}{J_m}s^2 + K\frac{J_m+J_{lm}}{J_m J_{lm}}s + \frac{b_v K}{J_m J_{lm}})s}. \tag{5.4}$$

Considering the limit case of a pure elastic joint without any dissipation effects (which is anyway the worst case scenario) the transfer functions can be written as:

$$\frac{\Theta_m}{\tau_m} = \frac{1}{J_m J_{lm}} \frac{K}{(s^2 + K\frac{J_m+J_{lm}}{J_m J_{lm}})s^2}, \tag{5.5}$$

$$\frac{\varphi}{\tau_m} = \frac{1}{J_m J_{lm}} \frac{(J_{lm}s^2 + K)}{(s^2 + K\frac{J_m+J_{lm}}{J_m J_{lm}})s^2}. \tag{5.6}$$

In Figure 5.1 and Figure 5.2, the typical transfer functions of the input torque to motor and link position respectively is shown.

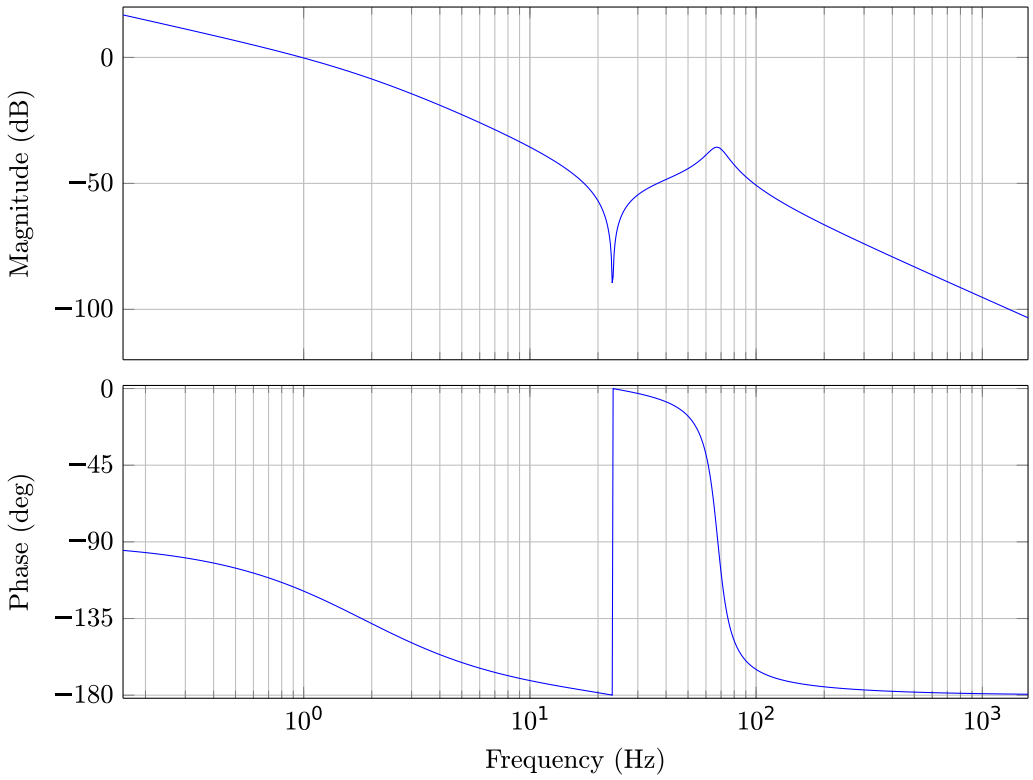


Figure 5.1: Transfer function of the hip flexion joint motor position.

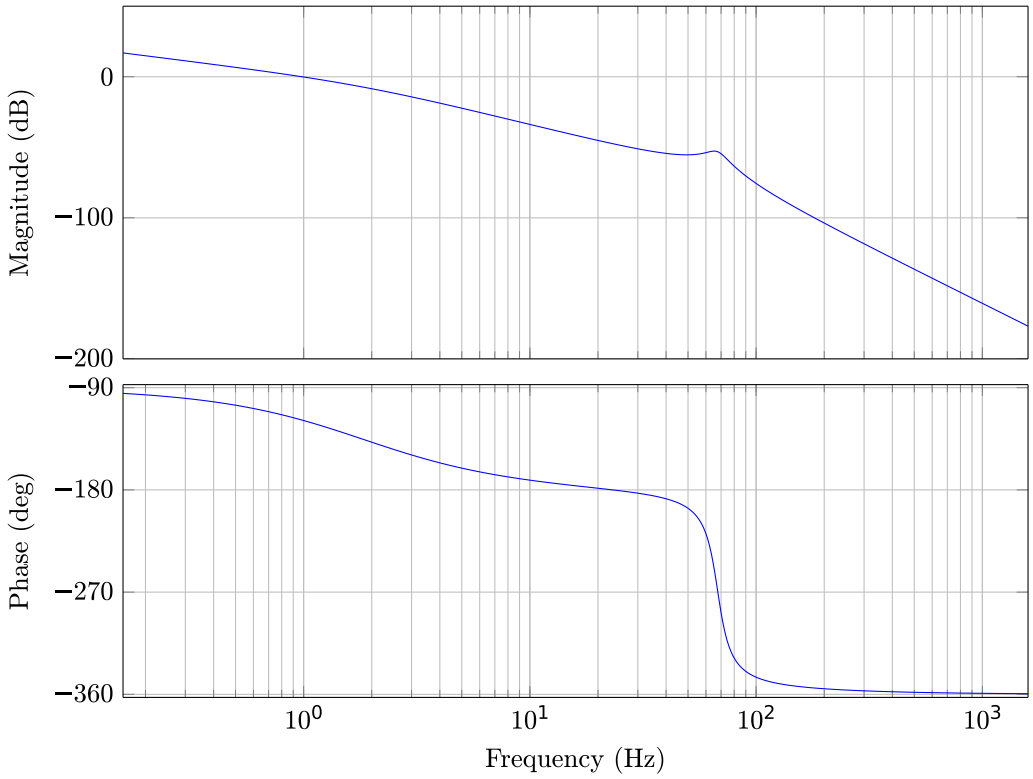


Figure 5.2: Transfer function of the hip flexion joint link position.

Equation (5.5) has a double pole in the origin, two imaginary poles and two imaginary zeros. The zero pair occurs at the frequency $\omega_l = \sqrt{K/J_{lm}}$ which characterizes the oscillation of the joint when the motor is locked. The presence of ω_l imposes a limit on the bandwidth and, consequently, on the gain of the motor feedback control loop. To avoid oscillations in the execution of the required trajectory, the bandwidth of the controller should be below ω_l. Equation (5.6) has a double pole in the origin, two imaginary poles and no zeros. For a detail discussion on the characteristics of the transfer functions (5.6) and (5.6) see Lohmeier [88] and De Luca and Book [38].

In Buschmann [20], De Luca and Book [38], the authors discuss the stability of the system (5.1)-(5.2). They consider different combinations of motor and joint position and velocity measurements in the feedback control loop and analyze each situation using Routh's criterion and Lyapunov stability theory respectively.

The use of link measurements only leads always to instability in the control loop independently of the values of the control gains (see Buschmann [20], De Luca and Book [38]). A combination of motor velocity and link position imposes limits on the gain values due to the elasticity: $K_{pp} < K$ and $K_{pv} > 0$. The less restrictive measurement combination is the use of motor side position and velocity measurements. The latter imposes that both the position and velocity feedback gains must be positive: $K_{pp} > 0$ and $K_{pv} > 0$. The results are summarized in Table 5.1.

Considering (4.16), (4.17) and (4.19) the control law for the joints of Lola can be written as follows:

$$\tau_m = k_\tau [K_{pv} K_{pp} (\varphi_d - \varphi) - K_{pv} \dot{\varphi}]. \tag{5.7}$$

By reorganizing the terms, the previous equation can be rewritten as a PD controller with velocity feed-forward compensation:

$$\tau_m = k_{\tau p} (\varphi_d - \varphi) - k_{\tau v} \dot{\varphi}, \tag{5.8}$$

$$k_{\tau p} = k_\tau K_{pv} K_{pp}, \tag{5.9}$$

$$k_{\tau v} = k_\tau K_{pv}. \tag{5.10}$$

Applying (5.8) to the model (4.32)-(4.33), De Luca and Book [38] show that (5.8) can globally stabilize the robotic joint system. Using motor side measurement only, the joint controller (5.8) is very robust and for high values of the gains $k_{\tau p}$ and $k_{\tau v}$ a very small tracking error on the motor side can be achieved. Nevertheless, due to the joint elasticity, the deformation of the gear leads to an increased error on the trajectory tracking on the link side.

Figure 5.3 shows tracking errors on the link ($e_\Theta = \Theta_d - \Theta$) and motor side ($e_\varphi = \varphi_d - \varphi$) for the hip flexion joint while the robot is walking forward at a speed of 3.4 km/h. The effect of the joint deformation can be seen comparing the two signals. The link side tracking error e_Θ is clearly larger in the static case at the beginning of the experiment when the robot stays and also while it is walking.

Figure 5.4 shows e_Θ and e_φ in the same experiment also for the knee joint. The static deformation of the joint is bigger in comparison with the hip flexion. While the robot is walking also a larger peak to peak errors for both the motor and link side tracking error can be seen.

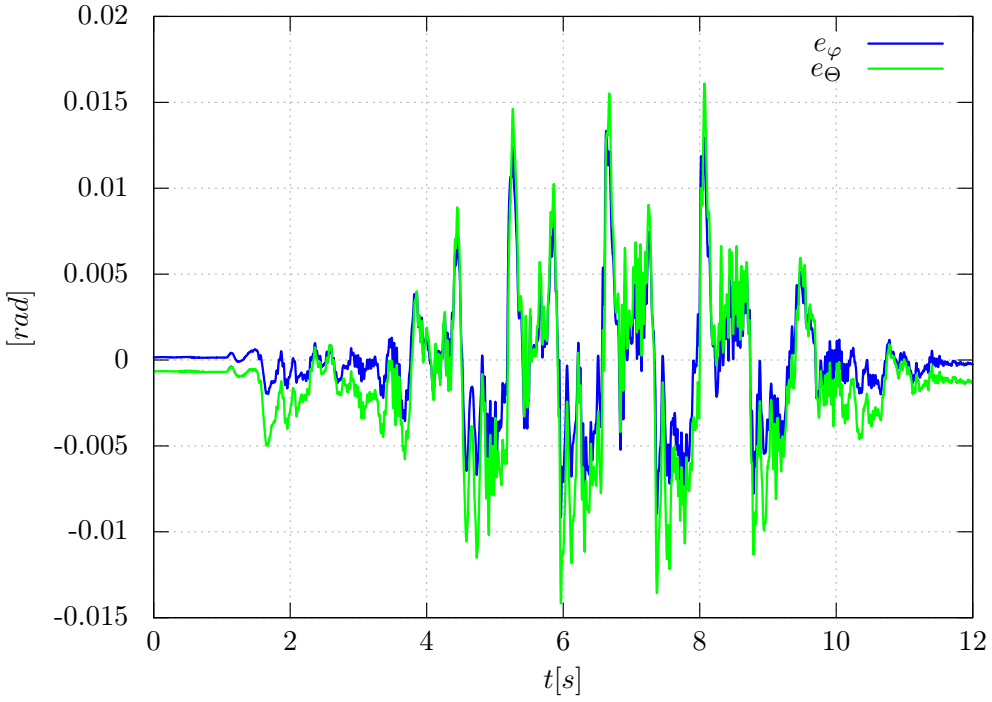


Figure 5.3: Tracking errors on the motor e_φ and link side e_Θ for the right hip joint (walking velocity 3.4 km/h). The motor tracking error has been projected to link side of the joint.

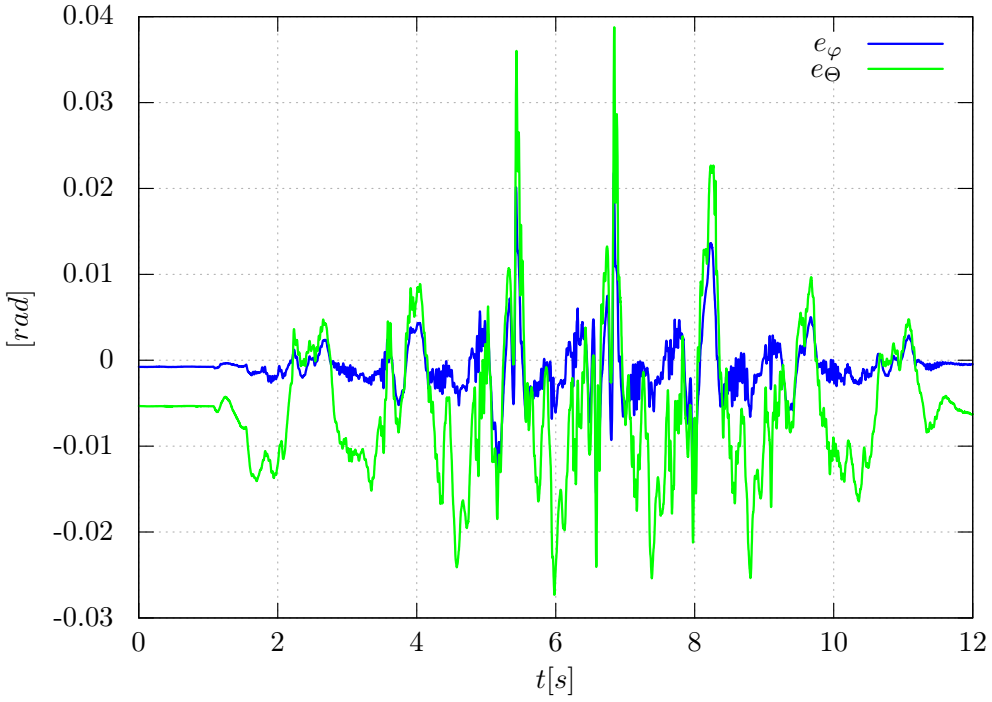


Figure 5.4: Tracking errors on the motor e_φ and link side e_Θ for the right knee joint (walking velocity 3.4 km/h). The motor tracking error has been projected to link side of the joint.

An interesting question is in which walking phase do the hip and knee joints present their biggest tracking error. To answer this question the measurement of the force torque sensor can be used. Plotting the normal contact force (F_z) of the right leg of the robot, the walking step can be divided into two phases: *a*) when the leg has contact with the floor, the sensor measures the total weight of the robot $\approx 600\,\mathrm{N}$. *b*) when the leg is moving forward to perform the step, the sensor measures a normal force of $\approx 0\,\mathrm{N}$.

Figure 5.5 and Figure 5.6 show a comparison between the leg normal force and the tracking error for the two considered joints. From the pictures two causes can be recognized for the decrease of tracking quality. The first one is when the joints experience the largest torque load for each phase *a*. This case can be clearly seen in the middle of the experiment between 6 s and 8 s, when the robot walks at the highest speed. The other source of error is the phase change to perform the walking step. The joints are commanded to execute a highly demanding trajectory, i. e., they are subject to large accelerations. For the hip joint this case can be seen between phase *b* and *a* when the leg must be moved forward. For the knee joint a similar situation occurs in the transition between phase *a* and *b*. In these transition, the robot is concluding the walking step and the leg must be correctly positioned on the ground. In this situation the tracking error presents its peak for both joints.

5.3 Integral Deformation Compensation

It has been shown that the presence of elasticity in the robot joints causes an increase of the trajectory tracking error on the link side in comparison with that on the motor side. Especially in the case of the knee the deformation of the joint mechanism causes a large error in comparison with all the other joints of the robot. Increasing the walk velocity, the loss of tracking performance causes instability in the control loop and, in the worst case, can lead to experiment fails and also the robot to fall.

In the proposed cascaded control loop (5.8) only feedback measurement from the motor side is used. To improve the tracking performance on the link side, the link position feedback must be introduced in the control law. Using the information delivered from the sensor, the goal is to reduce the static deflection caused by the gear elasticity and to improve the tracking performance on the link side of the joint.

A measurement of the gear deformation can be combined to (5.8) in order to compensate for the mentioned effects. The computation of the integral of the measured deformation $(\Theta_m - \varphi)$ can be added to the control law. The potential energy introduced by the joint stiffness is physically limited. Too large gear deformation would irremediably damage the joint (see [61]). For this reason the cascaded control law (5.8) can be improved as follows:

$$\tau_m = k_{\tau p}(\varphi_d - \varphi) - k_{\tau v}\dot{\varphi} + K_{\tau i}\int (\Theta_m - \varphi)\,\mathrm{d}t. \tag{5.11}$$

Under static control conditions, i. e., when the robot does not move but just stands, the static deformation of the joints is reduced and, consequently, leads the positioning error to zero. While the robot walks, the integral term reduces the mean value of the tracking error. The value of the integral gain $K_{\tau i}$ determines the time constant of the deformation zeroing. The drawback of a too high value of the integral gain is a memory effect that can lead to

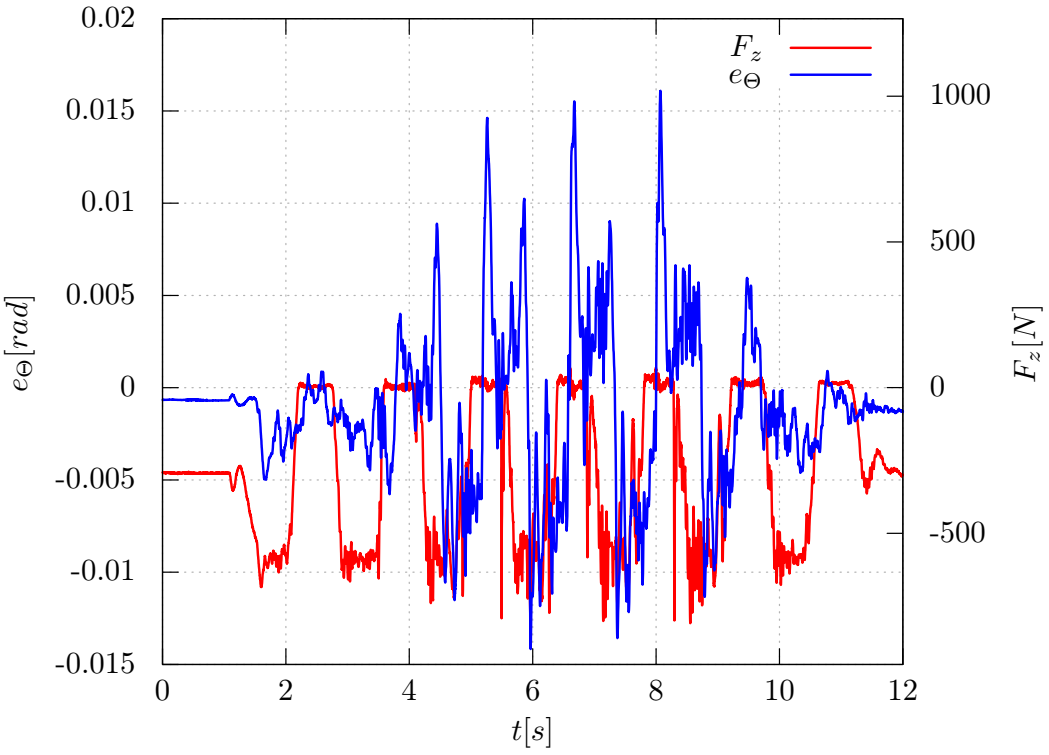


Figure 5.5: Hip link tracking error compared with normal contact force.

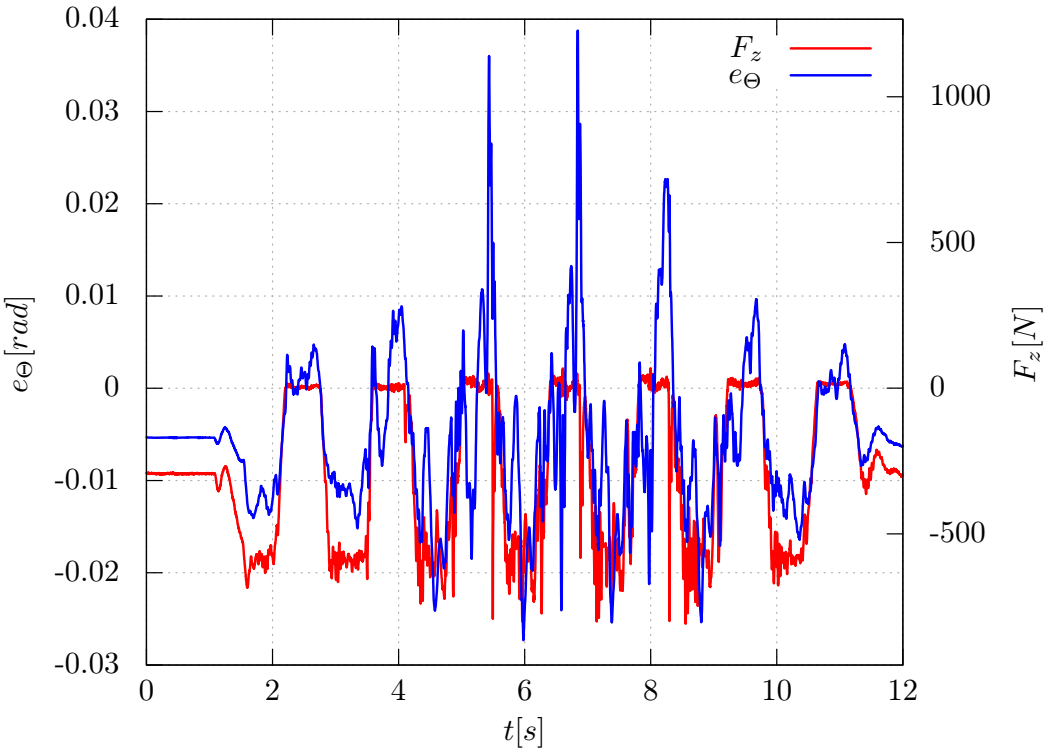


Figure 5.6: Knee link tracking error compared with normal contact force

Table 5.2: Knee and hip joint tracking error statistics for the cascaded *PD* and cascaded *PD+I* control methods.

Knee	*PD+I* $[mrad]$	*PD* $[mrad]$	Hip	*PD+I* $[mrad]$	*PD* $[mrad]$
Mean	**−0.060**	−5.557		**0.238**	−1.168
Peak	87.420	**38.765**		33.396	**16.091**
RMS	**222.505**	257.187		152.452	**115.746**

delays in the reaction of the feedback control and to an increase of the peak tracking error. This can happen if the desired trajectory requires high dynamic, i. e., for high speed and accelerations.

In Figure 5.7 and Figure 5.8 a comparison is shown between the link side tracking error when the *PD* control law (5.8) and *PD+I* control law (5.11) are used. The cascaded control law with integral deformation compensation removes the error offset caused by the gear deformation. In Table 5.2 the relevant statistics (mean, peck and RMS values) of the tracking error for the *PD* and *PD+I* are reported. For the hip joint the controller shows no further improvements in terms of tracking errors and reduction of the peak error. In the case of the knee joint the situation is different. The static offset has been successfully removed and the tracking error is lower than for cascaded controller. Nevertheless, it shows higher peaks.

5.4 Feed-Forward Compensation

The control laws presented in the previous section can stabilize the joint system and have been shown to deliver good tracking performance of the desired position trajectories for moderate walking velocities. Nevertheless, to further improve the control of the joints it is possible to take advantage of the system model and integrate more information in the control structure.

Siciliano et al. [125] propose the use of a model based feed-forward compensation for improved trajectory tracking in the case of a rigid industrial robots and for different control laws. In De Luca [36] and De Luca and Book [38] the authors discus the implementation of feed-forward compensation and feedback linearization techniques for the case of industrial robots with elastic joints.

The use of a feed-forward signal helps to, roughly speaking, leading the controlled system in to the "*right direction*", i. e., to follow the desired trajectory. In the ideal case of perfect modeling the latter is sufficient to let the system follow the imposed system trajectory. In any case, the use of feedback controllers is necessary for real systems. The feedback control helps to compensate for modeling errors and noise adding robustness. The feedback linearization is a control method that leads to a decoupling and linearization of the system, dynamically compensating its non-linear effects. The linearized system can then be controlled using a linear controller.

In Rouchon et al. [115] the authors discus the implementation of both feed-forward compensation and feedback linearization methods for industrial robots with elastic joints.

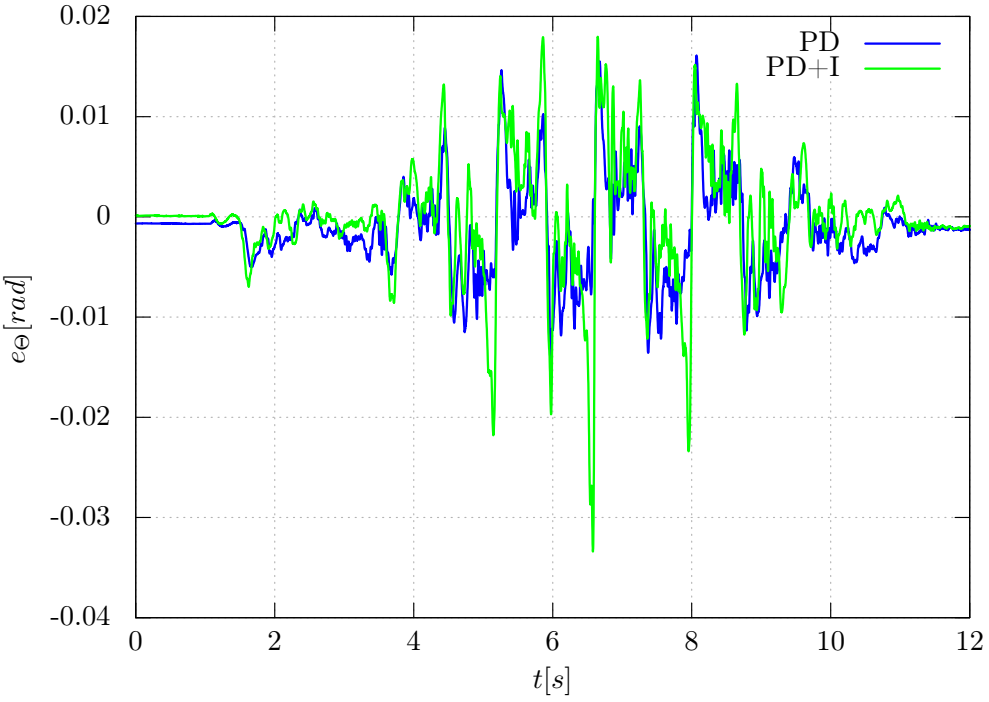


Figure 5.7: Comparison between the error on the link side for the right Hip joint (walking velocity 3.4 km/h) for the cascaded *PD* and cascaded *PD+I* with deformation compensation controllers.

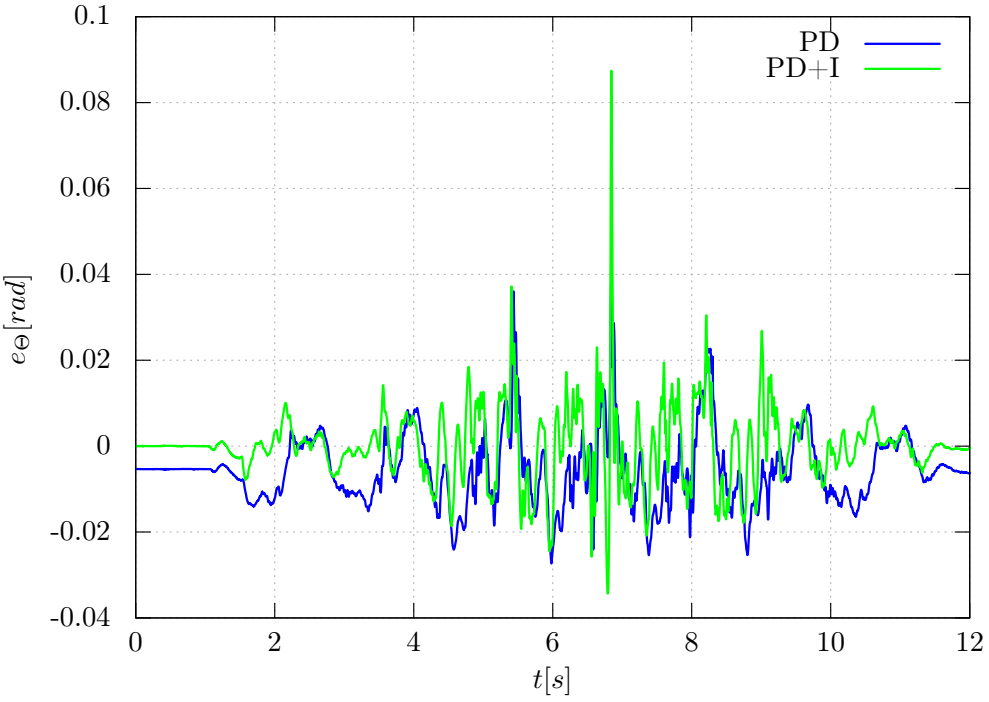


Figure 5.8: Comparison between the error on the link side for the right Knee joint (walking velocity 3.4 km/h) for the cascaded *PD* and cascaded *PD+I* with deformation compensation controllers.

They also show the equivalence between feedback linearization and differential flatness (see Moberg [97]). If the measurement of the link position Θ is chosen as flat output of the system, the model of the flexible robotic joint can be rewritten in terms of Θ and its successive time derivatives. In Hagenmeyer [63] the author investigates the application of feed-forward linearization of non-linear differential flat systems. He also demonstrates that a robust trajectory tracking performance can be achieved in presence of model parameter errors and limited inconsistent system initial conditions.

It must be mentioned that both the feed-forward compensation and feedback linearization control scheme, strongly depends on the correctness of the system model and on the smoothness of the trajectories used to drive the system under control. These are two fundamental requirements for a successful implementation of the mentioned methods.

Recalling that the proposed control law should be equally implementable on the DSCBs and the CCU, a controller based on the combination of a feed-forward steering signal and feedback action based on previously described local joint model is proposed.

5.4.1 Tracking Error System

The motor side elastic joint model described in Section 4.5.2, with constant stiffness and the friction function (4.29), constant link equivalent inertia and no load torque, can be written as follows:

$$J_{lm}\ddot{\Theta}_m + K(\Theta_m - \varphi) = \tau_{lm} \tag{5.12}$$

$$J_m\ddot{\varphi} + b_v\dot{\varphi} + \tau_{f0}\,\mathrm{sgn}(\dot{\varphi}) + K(\varphi - \Theta_m) = \tau_m, \tag{5.13}$$

$$J_{lm} = \frac{J_l}{N^2}, \tag{5.14}$$

$$\tau_{lm} = \frac{\tau_l}{N} = 0, \tag{5.15}$$

$$N = \frac{\Theta_m}{\Theta} = \frac{\dot{\Theta}_m}{\dot{\Theta}} = \frac{\ddot{\Theta}_m}{\ddot{\Theta}}. \tag{5.16}$$

The link equation can be differentiated two times:

$$J_{lm}\dddot{\Theta}_m + K(\ddot{\Theta}_m - \ddot{\varphi}) = 0, \tag{5.17}$$

$$J_{lm}\Theta_m^{[4]} + K(\dddot{\Theta}_m - \dddot{\varphi}) = 0. \tag{5.18}$$

where $x^{[n]}$ denotes the n^{th} time derivative of x: $d^n x/dt^n$.

From (5.18) the expression of $\ddot{\varphi}$ can be computed:

$$\ddot{\varphi} = \frac{J_{lm}}{K}\Theta_m^{[4]} + \ddot{\Theta}_m. \tag{5.19}$$

Substituting (5.19) into (5.13) and reorganizing the terms, the following equation is

obtained:

$$\Theta_m^{[4]} + \frac{b_v}{J_m}\dddot{\Theta}_m + K\frac{J_{lm}+J_m}{J_{lm}J_m}\ddot{\Theta}_m + K\frac{b_v}{J_{lm}J_m}\dot{\Theta}_m + K\frac{\tau_{f0}}{J_{lm}J_m}\operatorname{sgn}(\dot{\Theta}_m) = \frac{K}{J_{lm}J_m}\tau_m. \tag{5.20}$$

The latter is equivalent to the system (5.12)-(5.13) in terms of the link position Θ_m and its subsequent derivatives $\Theta_m^{[4]}, \dddot{\Theta}_m, \ddot{\Theta}_m$ and $\dot{\Theta}_m$. The function $\operatorname{sgn}(\dot{\varphi})$ has also been substituted by $\operatorname{sgn}(\dot{\Theta}_m)$, under the assumption of a non-negligible stiffness coefficient.

The reference torque τ_{mff} can be computed by inverting the system equation (5.20):

$$\tau_{mff} = \frac{J_{lm}J_m}{K}\left[\Theta_m^{[4]} + \frac{b_v}{J_m}\dddot{\Theta}_m + K\frac{J_{lm}+J_m}{J_{lm}J_m}\ddot{\Theta}_m + K\frac{b_v}{J_{lm}J_m}\dot{\Theta}_m + K\frac{\tau_{f0}}{J_{lm}J_m}\operatorname{sgn}(\dot{\Theta}_m)\right]. \tag{5.21}$$

By Substituting Θ_m and its subsequent derivatives with their measurement, (5.21) can be seen as the feedback linearizing torque of the system. On the other side, substituting the desired trajectories Θ_{md} yields the feed-forward torque reference.

Applying the reference torque signal (5.21) to the system equation (5.20) the following is obtained:

$$\begin{aligned}\Theta_m^{[4]} - \Theta_{md}^{[4]} = {} & \frac{b_v}{J_m}(\dddot{\Theta}_{md} - \dddot{\Theta}_m) \\ & + K\frac{J_{lm}+J_m}{J_{lm}J_m}(\ddot{\Theta}_{md} - \ddot{\Theta}_m) \\ & + K\frac{b_v}{J_{lm}J_m}(\dot{\Theta}_{md} - \dot{\Theta}_m) \\ & + K\frac{\tau_{f0}}{J_{lm}J_m}\Delta_f \\ \Delta_f = {} & \operatorname{sgn}(\dot{\Theta}_{md}) - \operatorname{sgn}(\dot{\Theta}_m).\end{aligned} \tag{5.22}$$

Defining the tracking error as $e_{\Theta_m} = (\Theta_{md} - \Theta_m)$ the equation can be rewritten as:

$$\begin{aligned}e_\Theta^{[4]} = {} & -\frac{b_v}{J_m}\dddot{e}_{\Theta_m} \\ & - K\frac{J_{lm}+J_m}{J_{lm}J_m}\ddot{e}_{\Theta_m} \\ & - K\frac{b_v}{J_{lm}J_m}\dot{e}_{\Theta_m} \\ & - K\frac{\tau_{f0}}{J_{lm}J_m}\Delta_f.\end{aligned} \tag{5.23}$$

A necessary but not sufficient condition for a stable error dynamics is that equation (5.23) must fulfill the Hurwitz criterion (see Slotine and Weiping [128]), i. e., all the

coefficients on the right hand side of the equation must be negative. This condition is satisfied, since the system parameter are all positive.

The last step to solve the trajectory tracking problem is the definition of the feedback action to add to the feed-forward torque reference, in order to steer the system states on the desired trajectory. One possible implementation is:

$$\tau_{md} = \tau_{mff} + k_4 \dddot{e}_{\Theta_m} + k_3 \ddot{e}_{\Theta_m} + k_2 \dot{e}_{\Theta_m} + k_1 e_{\Theta_m} + k_0 \int e_{\Theta_m} \mathrm{d}t, \tag{5.24}$$

where the factors $k_i, i = 0,1,\ldots,4$ are suitable control gains.

Adding the feedback correction to (5.23), the error equation can be rewritten as:

$$\begin{aligned} e_{\Theta}^{[4]} = &- \left(\frac{b_v}{J_m} + k_4 \frac{K}{J_{lm} J_m}\right) \dddot{e}_{\Theta} \\ &- \left(K \frac{J_{lm} + J_m}{J_{lm} J_m} + k_3 \frac{K}{J_{lm} J_m}\right) \ddot{e}_{\Theta} \\ &- \left(K \frac{b_v}{J_{lm} J_m} + k_2 \frac{K}{J_{lm} J_m}\right) \dot{e}_{\Theta_m} \\ &- k_1 \frac{K}{J_{lm} J_m} e_{\Theta_m} \\ &- k_0 \frac{K}{J_{lm} J_m} \int e_{\Theta} \mathrm{d}t \\ &- K \frac{\tau_{f0}}{J_{lm} J_m} \Delta_f . \end{aligned} \tag{5.25}$$

Under the assumption of correct model parameters, consistent initial conditions of the system and for smooth enough reference trajectories[1], equation (5.24) feed-forward linearize the system and leads the error dynamic (5.25) to zero.

5.4.2 Implementation of the Feed-Forward Compensation Control Law

Some considerations must be done on the proposed controller with feed-forward compensation. The feed-forward compensation signal (5.21) is a reference torque which, for the assumption made in Section 4.4.1, can be seen as a reference motor current. This implies that the controller (5.24) directly delivers a desired current signal to the current controller (4.12). In this way the velocity controller (4.17) would not be needed anymore. This implementation would have two disadvantages. In its current implementation, the SERCOS-III update rate cannot be lower than 1.5 ms, while the current controller is executed every 70 μs. That results in a delay (21 times slower) in the desired trajectory delivery that is unacceptable for the stability of the real robot, except for very slow movements. Moreover, the velocity controller adds a velocity dependent damping to the joint control loop that increases the stability of the system. For these reasons the use of the velocity

1 At least four time differentiable.

controller is highly desirable. Equation (5.21) can be rewritten on the motor velocity level. Using (4.17) and (4.12) the motor torque can be expressed as follows:

$$\tau_m = k_\tau I_d = \beta(\dot{\varphi}_d - \dot{\varphi}), \tag{5.26}$$

$$\beta = k_\tau K_{pv} sf_i sf_{\dot{\varphi}}. \tag{5.27}$$

Where sf_i and $sf_{\dot{\varphi}}$ are the scaling factors of the motor current and motor velocity controllers respectively. Equation (5.27) can be substituted in the system (5.12)-(5.13). A new feed-forward reference signal and tracking error equation can be computed in terms of the desired velocity for the controller (4.17):

$$\begin{aligned}\dot{\varphi}_{ff} = \frac{J_{lm}J_m}{K\beta}\Big[\Theta_m^{[4]} + \frac{b_v+\beta}{J_m}\dddot{\Theta}_m + K\frac{J_{lm}+J_m}{J_{lm}J_m}\ddot{\Theta}_m \\ + K\frac{b_v+\beta}{J_{lm}J_m}\dot{\Theta}_m + K\frac{\tau_{f0}}{J_{lm}J_m}\operatorname{sgn}(\dot{\Theta}_m)\Big].\end{aligned} \tag{5.28}$$

$$\begin{aligned}e_{\Theta_m}^{[4]} = &-\Big(\frac{b_v+\beta}{J_m} + k_4\frac{K\beta}{J_{lm}J_m}\Big)\dddot{e}_\Theta \\ &-\Big(K\frac{J_{lm}+J_m}{J_{lm}J_m} + k_3\frac{K\beta}{J_{lm}J_m}\Big)\ddot{e}_\Theta \\ &-\Big(K\frac{b_v}{J_{lm}J_m} + k_2\frac{K\beta}{J_{lm}J_m}\Big)\dot{e}_{\Theta_m} \\ &-k_1\frac{K\beta}{J_{lm}J_m}e_{\Theta_m} \\ &-k_0\frac{K\beta}{J_{lm}J_m}\int e_{\Theta_m}\,\mathrm{d}t \\ &-K\frac{\tau_{f0}}{J_{lm}J_m}\Delta_f.\end{aligned} \tag{5.29}$$

Being $\beta \gg 1$ due to the current and velocity conversion factors, this implementation of the tracking error equation enables stabilization with lower values of the feedback gains k_i.

The available measurement of the joint state are motor position, motor velocity and link position. While the second, third and fourth derivative of the desired trajectories can be computed, they cannot be physically measured. Their value can be estimated using (5.12), (5.17) and (5.18). In this way the link position measurement must be differentiated only one time to compute $\dot{\Theta}_m$.

Another possibility is to use the same equation and rewrite (5.28) and (5.29) in the more suitable set of state variables $\boldsymbol{x}_m = [\varphi, \Theta_m, \dot{\varphi}, \dot{\Theta}_m]^T$. The tracking error can be written in terms of $\boldsymbol{e} = [e_\varphi, e_{\Theta_m}, \dot{e}_\varphi, \dot{e}_{\Theta_m}, \int e_{\Theta_m}\,\mathrm{d}t]^T$ considering the following expressions:

$$\dddot{e}_{\Theta_m} = \frac{K}{J_{lm}}(\dot{e}_\varphi - \dot{e}_{\Theta_m}), \tag{5.30}$$

$$\ddot{e}_{\Theta_m} = \frac{K}{J_{lm}}(e_\varphi - e_{\Theta_m}), \tag{5.31}$$

$$e_\varphi = (\varphi_d - \varphi), \tag{5.32}$$

$$\dot{e}_\varphi = (\dot{\varphi}_d - \dot{\varphi}). \tag{5.33}$$

Equation (5.29) can then be rewritten as:

$$\begin{aligned} e_\Theta^{[4]} = &- \frac{K}{J_{lm}}\Big(\frac{b_\nu + \beta}{J_m} + k_4\frac{K\beta}{J_{lm}J_m}\Big)\dot{e}_\varphi \\ &- \frac{K}{J_{lm}}\Big(K\frac{J_{lm}+J_m}{J_{lm}J_m} + k_3\frac{K\beta}{J_{lm}J_m}\Big)e_\varphi \\ &- \frac{K}{J_{lm}}\Big(k_2\frac{\beta}{J_m} - k_4\frac{K\beta}{J_{lm}J_m}\Big)\dot{e}_{\Theta_m} \\ &- \frac{K}{J_{lm}}\Big(k_1\frac{\beta}{J_m} - k_3\frac{K\beta}{J_{lm}J_m} - \frac{J_{lm}+J_m}{J_m}\Big)e_{\Theta_m} \\ &- k_0\frac{K\beta}{J_{lm}J_m}\int e_\Theta \,\mathrm{d}t \\ &- K\frac{\tau_{f0}}{J_{lm}J_m}\Delta_f. \end{aligned} \tag{5.34}$$

The eigenvalues of (5.34), neglecting the Δ_f term, can be computed to be stable using pole placement and guaranteeing that the equation fulfill the Hurwitz stability criterion.

The final form of the proposed control law can be written as follows:

$$\begin{aligned} \dot{\varphi}_d = \dot{\varphi}_{ff} + k_4\frac{K}{J_{lm}}\dot{e}_\varphi + k_3\frac{K}{J_{lm}}e_\varphi + (k_2 - k_4\frac{K}{J_{lm}})\dot{e}_{\Theta_m} \\ + (k_1 - k_3\frac{K}{J_{lm}})e_{\Theta_m} + k_0\int e_{\Theta_m}\,\mathrm{d}t. \end{aligned} \tag{5.35}$$

The first term $\dot{\varphi}_{ff}$ is the feed-forward action defined in (5.28). The remaining terms are the components of the feedback action in the system state space $\boldsymbol{x}_m$. The feedback gains k_i are the same as for (5.24). Advantage of (5.35) is that it can be implemented without changing the inner motor velocity control loop and motor current control loop.

Another aspect that must be taken into account is the generation of the desired trajectories. In the current implementation the available reference trajectories are the desired link position Θ_d and velocity $\dot{\Theta}_d$. They are computed in real-time while the robot is walking. To generate the feed-forward reference (5.28) the desired velocity should be derived at least four times to obtain the reference signal for the joint acceleration $\ddot{\Theta}_d$, jerk $\dddot{\Theta}_d$ and $\Theta_d^{[4]}$. The values of the weighting factors of (5.28) for the knee and hip joint of the robot are reported in Table 5.3[2]. The relative weight of the jerk factor (IV) is three orders of magnitude smaller than the acceleration (III) and the static friction (I) factors. It is also

2 The factors have been computed using the model parameters from Chapter 4

Table 5.3: feed-forward reference motor velocity factors.

	Factor	Hip Flexion	Knee
I)	$K\frac{\tau_{f0}}{J_{lm}J_m}$	1.628×10^{-3}	$6.180e^{-4}$
II)	$K\frac{b_v+\beta}{J_{lm}J_m}$	1	1
III)	$K\frac{J_{lm}+J_m}{J_{lm}J_m}$	24.677×10^{-3}	4.402×10^{-3}
IV)	$\frac{b_v+\beta}{J_m}$	47.014×10^{-6}	7.909×10^{-6}

six orders of magnitude smaller in comparison with the velocity factor (II) and can be, therefore, neglected. A further simplification is to avoid also the computation of $\Theta_d^{[4]}$. Only one differentiation must then be carried out to compute the reference acceleration $\ddot{\Theta}_d$ and the feed-forward compensation signal (5.28). With these considerations the reference trajectory generator of LOLA can further be used. The reference trajectories for the motor position and velocity can be computed from the desired link position and velocity using the following expressions:

$$\varphi_d = \frac{K}{J_{lm}}\ddot{\Theta}_{md} + \Theta_{md}, \tag{5.36}$$

$$\dot{\varphi}_d = \frac{K}{J_{lm}}\dddot{\Theta}_{md} + \dot{\Theta}_{md} \approx \dot{\Theta}_{md}. \tag{5.37}$$

5.4.3 Simulation Results

In order to verify the validity and performance of the control law and the simplifications presented in the previous section, the feed-forward compensation controller (5.24) has been implemented in the multibody simulation of LOLA. The tracking controller has been computed for the two hip flexion and two knee joints because those are the joints which present the largest tracking error (see Chapter 4). Nevertheless, it can equally be implemented for all other joints of the robot.

The feedback gains k_i are computed using the pole placement algorithm for the reference joint model (5.12)-(5.13). The stiffness and friction parameters used for the reference model are the values estimated in Chapter 4. To verify the robustness of the control law against parameter variations at least in static conditions, it has been verified that the tracking error equation (5.34) is stable also for different values of the link inertia J_{lm} computed from the multibody simulation and for different stiffness values calculated using (4.31).

Figure 5.9 and Figure 5.10 show the link position error of the hip and knee joints, for the robot walking at 3.6 km/h. Using the feed-forward controller (5.35) the tracking error has been drastically reduced for both the considered joints. The controller is able to compensate the static elastic error which globally improves the tracking performance.

Figure 5.11 and Figure 5.12 show a comparison between the feed-forward reference computed with (5.28) and the desired link velocity $\dot{\Theta}_m$ used for the cascaded *PD* controller.

Table 5.4: Knee and Hip joint tracking error statistics for the *FFC* and *PD* control methods (Simulation).

Knee	*FFC* $[mrad]$	*PD* $[mrad]$	Hip	*FFC* $[mrad]$	*PD* $[mrad]$
Mean	$\mathbf{-0.007}$	-2.920		$\mathbf{0.001}$	0.164
Peak	$\mathbf{9.143}$	18.149		$\mathbf{3.909}$	12.694
RMS	$\mathbf{59.483}$	146.086		$\mathbf{25.183}$	83.338

Table 5.5: Knee and Hip joint tracking error statistics for *FFC* and *PD* control methods (Experiment).

Knee	*FFC* $[mrad]$	*PD* $[mrad]$	Hip	*FFC* $[mrad]$	*PD* $[mrad]$
Mean	$\mathbf{-0.096}$	1.548		$\mathbf{0.001}$	0.407
Peak	$\mathbf{7.006}$	8.923		$\mathbf{4.516}$	5.111
RMS	$\mathbf{61.394}$	92.058		$\mathbf{29.802}$	37.931

Both signals are very similar. Nevertheless, the signal improvements in case of the feed-forward reference due to the acceleration and static friction terms of (5.28), slightly modify the signal.

In Table 5.4 the relevant statistics of the tracking error for the *PD* and *FFC* are reported. As can be seen, the *FFC* control method compensates for the static deformation of both joints, reduces the peak error and drastically improves the RMS value of the tracking error.

5.4.4 Experimental Results

The feed-forward controller (5.35) has been implemented in the CCU replacing the position controller (4.19). The sampling frequency of the position loop remains 1.5 ms. The motor velocity and current control loop have not changed and (4.17) and (4.12) are still valid.

As a first experimental test, a walk without ground contact has been conducted. The robot hang in the air attached to a safety rope while performing the walking experiment. The link side tracking error for this case is shown in Figure 5.13 for the right hip joint and in Figure 5.14 for the right knee, in comparison with the cascaded *PD* performance. While the error is globally smaller for the feed-forward controller, the performance improvement is not as good as in the simulation. At any rate, the controller compensates for static errors and also slightly reduces the peak value. In Table 5.5 the relevant statistics of the tracking error for the *PD* and *FFC* are reported. As can be seen, the *FFC* method improves the tracking error for both joints.

Experiments with ground contact has also been performed. Letting the robot stepping the systems become unstable and the experiments must be interrupted. The analysis of

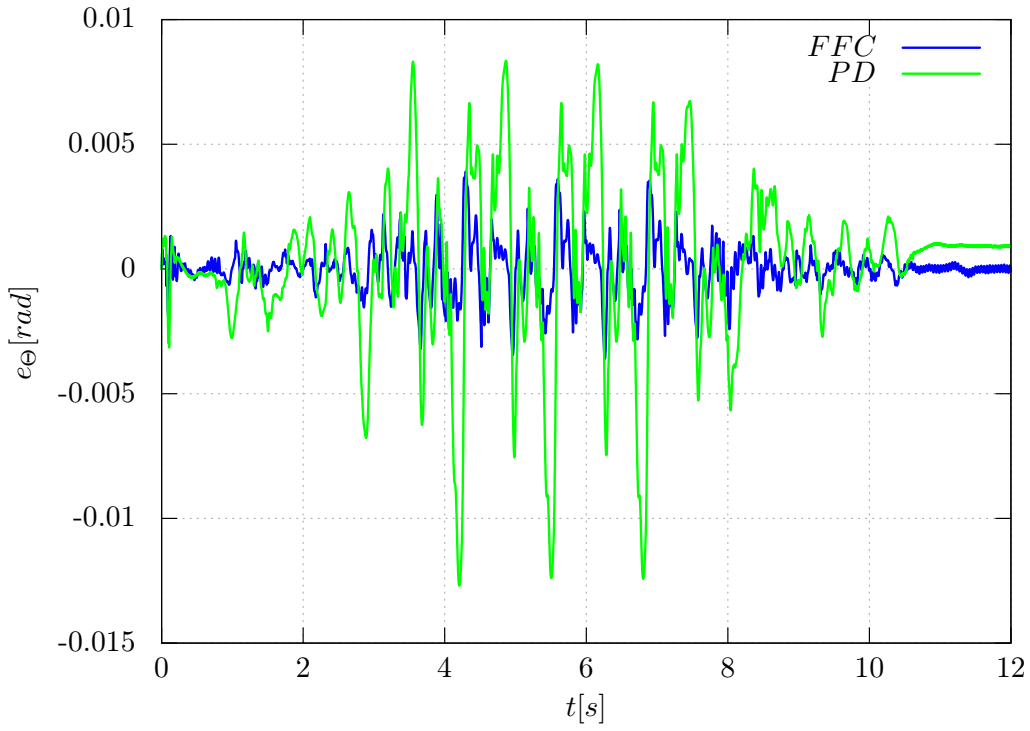


Figure 5.9: Link tracking error comparison of *FFC* and cascaded *PD* controllers for the right hip (Simulation).

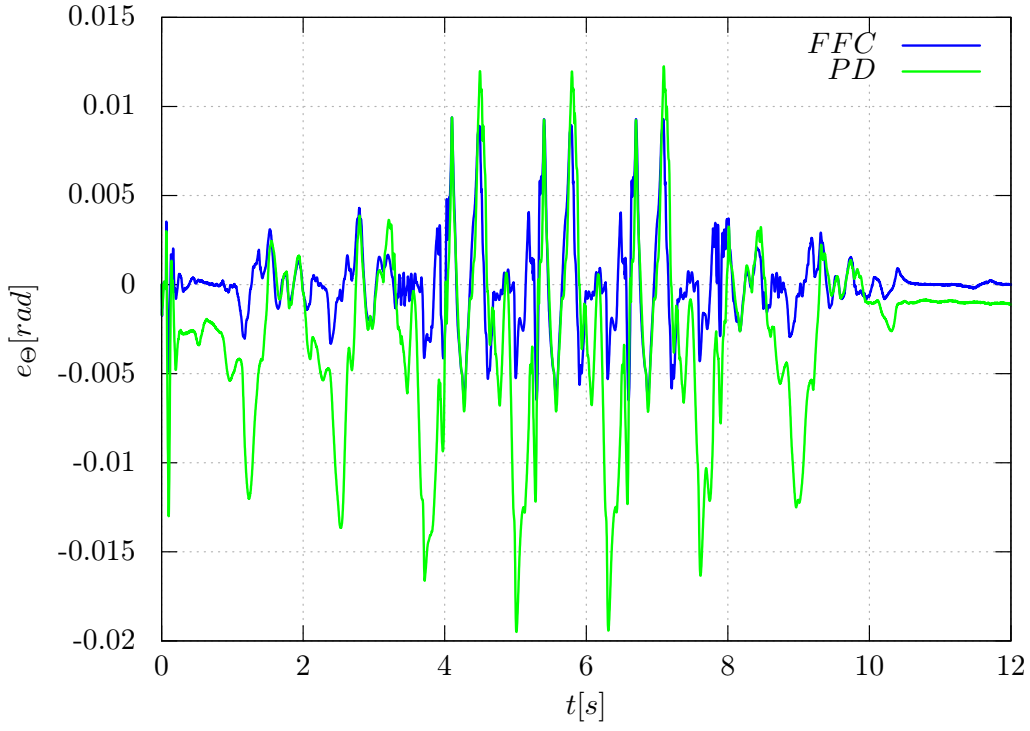


Figure 5.10: Link tracking error comparison of *FFC* and cascaded *PD* controllers for the right knee (Simulation).

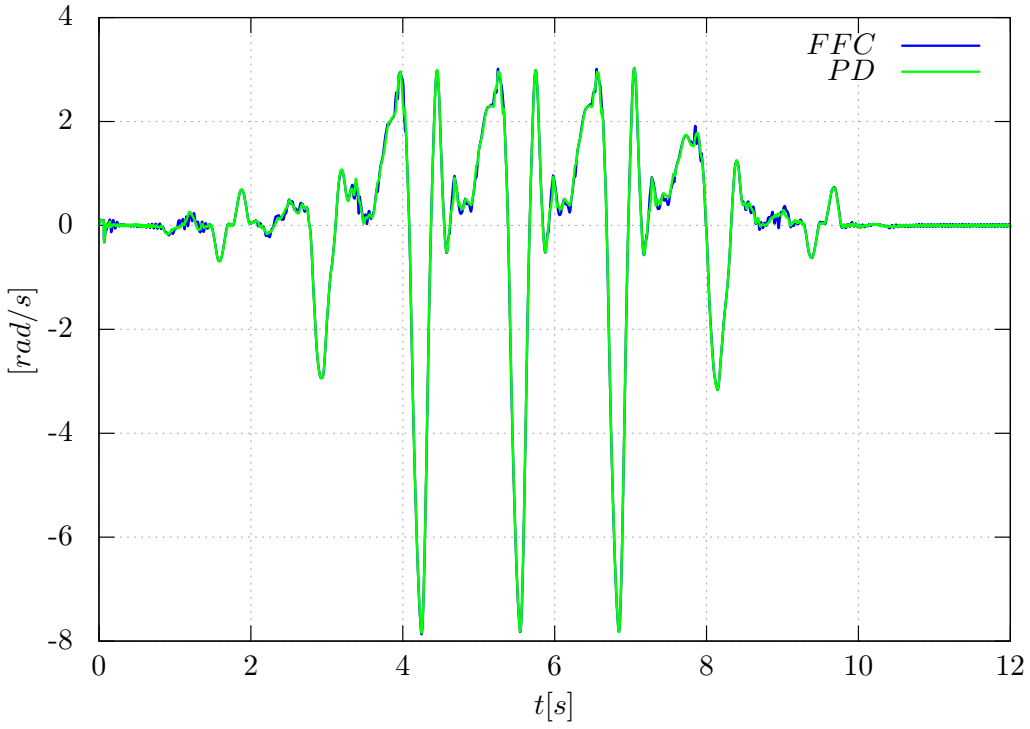


Figure 5.11: Comparison between of *FFC* reference signal and desired velocity of the cascaded *PD* for the right hip (Simulation).

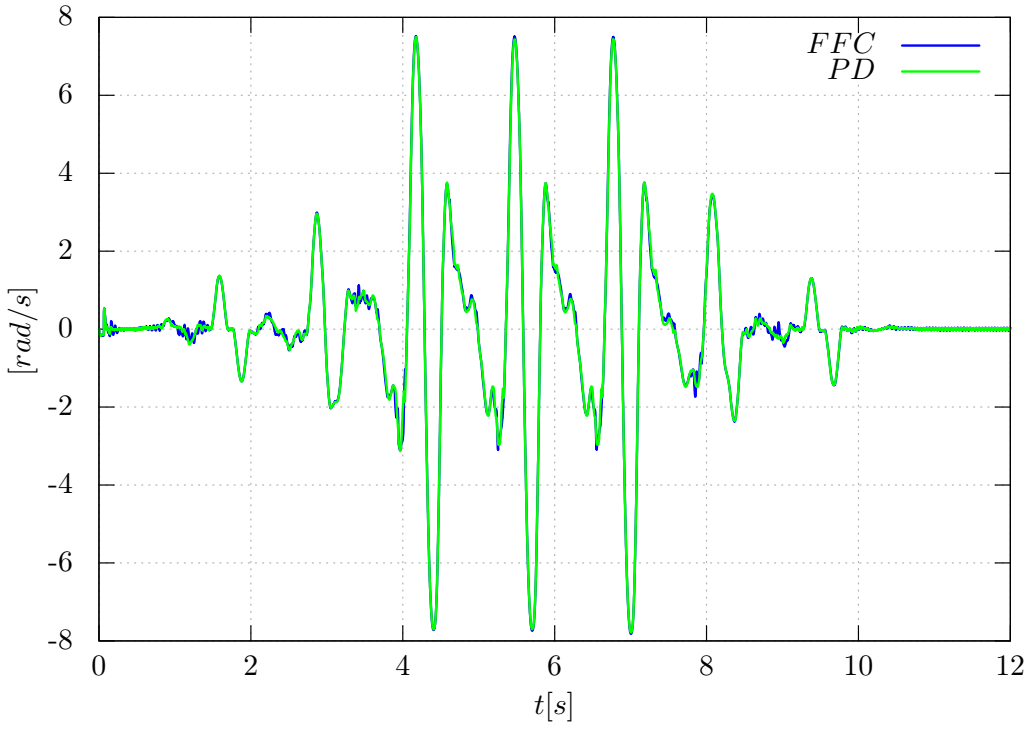


Figure 5.12: Comparison between of *FFC* reference signal and desired velocity of the cascaded *PD* for the right knee (Simulation).

the experiment data have shown that the joints controlled by the feed-forward controller present strong oscillations in the feed-forward reference caused by the desired acceleration signal. The reason for this effect must be related with activation of the force stabilization control. In the previous experiment this component of the robot system was not necessary and the acceleration signal computed from the reference velocity was sufficiently smooth. The force stabilization controller is one of the most important components of the robotic system and must be used to achieve stable walking. To successfully use the proposed feed-forward controller the trajectory generation unit of the robot should be able to deliver a sufficiently smooth reference joint acceleration.

5.5 Chapter summary

In this chapter a study of a suitable control law for joint trajectory tracking has been presented. The state of the art in the field of robotic joint control has been discussed. In the literature many different approaches to achieve reliable and robust control of robotic joints can be found. Nevertheless, the main focus of the research is directed to robots for industrial application. In the field of humanoid robotics, the control of the joints is (in almost all cases) implemented as classic a PID controllers, relying on the use of gears with high transmission ration to limit the effect of the non-linear coupling of the joints and the disturbance torques.

The problem of controlling robotic joints with elastic gears has been addressed. Two kinds of model based approaches has been presented and their performance are analyzed with the use of simulation tools and experimental data. The *PD+I* control law has shown improvement in terms of static deformation and RMS value of the tracking error (at least for the knee joint) and higher peak in comparison with the *PD* controller. The *FFC* control law has shown a drastic improvement for the tracking error of both joints. It compensates for static deformation, reduces the peak and the RMS value of the error. Nevertheless, these control law has proven to be difficult to implement because it relies on higher order derivatives of the reference trajectory, which, for the walking controller, also depends on the contact force measurement. The desired joint velocity has a component proportional to the contact force, therefore, the reference acceleration contains a force derivative, resulting in a very noisy signal.

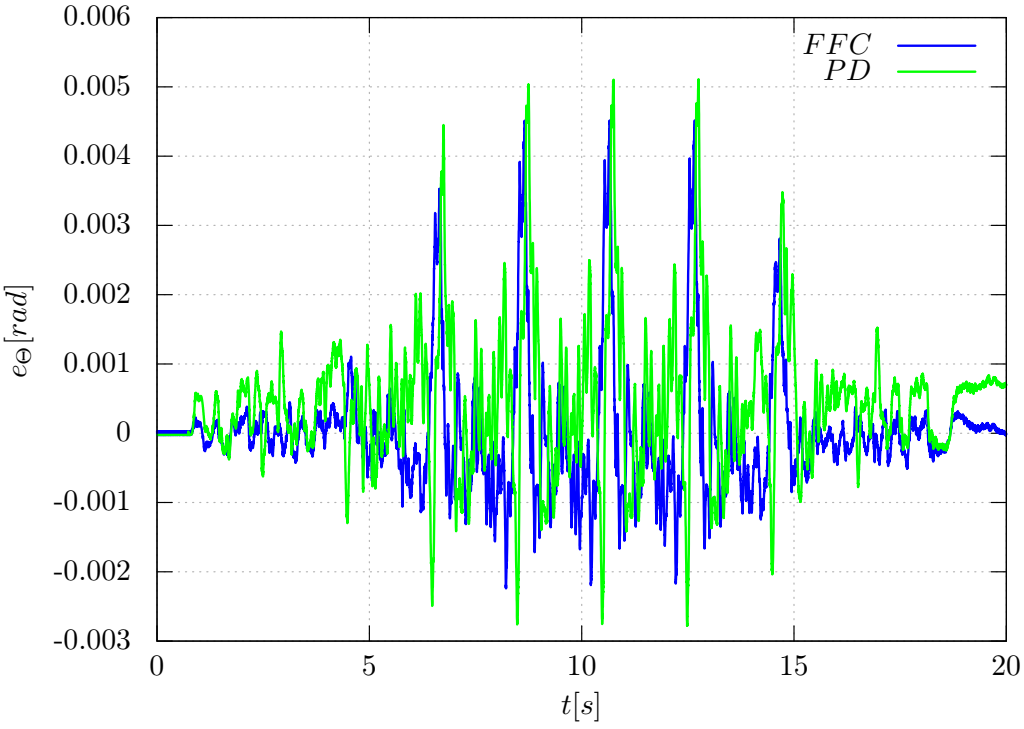


Figure 5.13: Link tracking error comparison of *FFC* and cascaded *PD* controllers for the right Hip (Experiment).

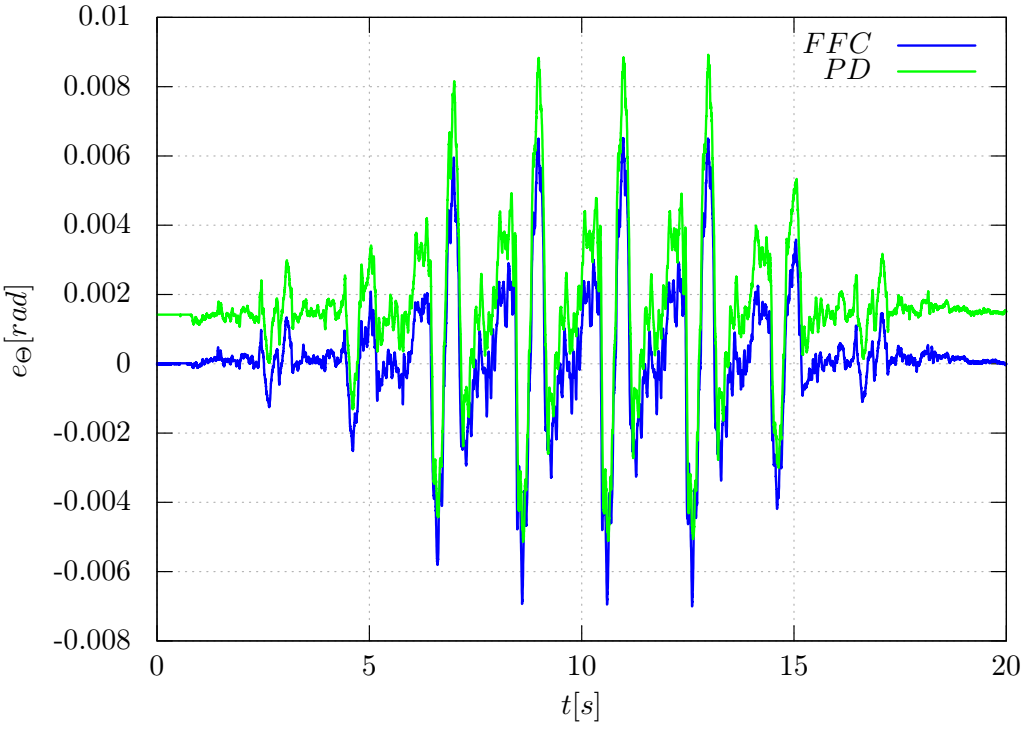


Figure 5.14: Link tracking error comparison of *FFC* and cascaded *PD* controllers for the right Knee (Experiment).

6 Conclusions

Humanoid robotics is a growing research field in the last forty years. The possible applications vary from service robotics and human machine collaboration in the industry to the employment of bipedal machines in hazardous or calamity environments. Nevertheless, they are also one of the most complete mechatronic systems and, therefore, very interesting as research platforms. They are the perfect intersection of different engineering fields like mechanics, electronics, information technology, sensor technology and control.

The following sections give a brief discussion and summary of the subjects introduced in this work and some suggestion for future development are presented.

6.1 Chapters Discussion

This thesis covers different aspects of the concept development and realization of complex distributed control structures. In the first part, the topology and implementation of the communication framework of the humanoid robot LOLA is presented. The complexity of the information generated by local sensors are managed by using a decentralized network that is able to harvest the data and make them available to the main controller of the system. The different communication protocols used by the sensors and the actuation unit are described. The key point of the entire structure are the DSCB custom developed boards. They locally interface the sensors and actuators of the system and make their data transparently available for further elaboration. A “bare metal” software is developed to take advantage of the system framework and to guarantee the safety of the system.

An extension of the position sensor system for FPGA implementation is discussed in the third chapter. To overcome the bottleneck of the motor position data transmission in the first implementation of the sensor network, a suitable IP-Core is developed. The motor state information is extended with a motor velocity estimator. Different velocity estimation algorithms are discussed and compared. Exploiting the hardware features of the available FPGA resources on the DSCBs, an improved realization of a first order differentiator method is successfully realized. Key points of the implementation are automatic scalability of the working sampling frequency of the system and the simple parameterization. In comparison with comparable algorithms, the Extended Constant Elapsed Time differentiation algorithm shows improvements in terms of rms, mean and maximum error and better performance for slow velocity detection.

The second part of the thesis deals with the extension to former works of the robot simulation and the modeling, parameterization and control of the robotic joints.

A necessary and powerful tool for the design and behavior prediction of every mechatronic system is its simulation program. The multibody simulation of LOLA is a complex software which enables a detailed analysis of the robotic system. Different detail levels of

the robot model can be used depending on the purpose of the simulation. In this work, the extension of the full simulation is integrated with an accurate model of the actuation train. The local controllers are implemented with their real sampling times for each control loop, enabling the use of the control gains used on the real system also in the robot simulation.

To improve the accuracy of the multibody simulation software, the extension to joints with elastic gears of the robot is discussed. The model developed in previous works is based on catalog data of the gears delivered good results. Nevertheless, it has never been experimentally verified. In the fourth chapter, a reduced generic model of an elastic joint with friction is proposed. The model parameters are computed using a recursive least square algorithm using experimental data of the robot. It is shown that the model can describe the real gear deformation in the multibody simulation.

In the last chapter, different joint control methods are studied. The performance of the implemented cascaded PD controller for rigid joints is analyzed. Based on the elastic joint model developed in the Chapter 4, two model based control laws are also proposed. The first one is an extension of the cascaded PD controller with integral elasticity compensation. Taking advantage of the link side position measurement of the robot joints, the algorithm can reduce the offset in the link side position trajectory tracking. However, it has also the drawback of higher peak errors for high dynamic trajectories.

To overcome this problem, a combination of a model based feed-forward and feedback controller is proposed. The feed-forward part delivers a steering trajectory reference to the system while the feedback controller corrects the system response and adds robustness to the control law. Both, the feed-forward and feedback part of the controller require a high degree of smoothness of the system reference trajectory, until the fourth derivative of the link position. The control law is rewritten in the joint system state space of motor and link position and velocity, avoiding the estimation or computation of higher derivatives of the reference trajectories and measured system states. Furthermore, the control scheme is adapted to the already existing software structure of LOLA.

The results obtained in the multibody simulation of the robot are very good on terms of joint position trajectory tracking. The control scheme shows, nevertheless, some limitations in the direct implementation on the real system. The reference trajectory generator in combination with the force stabilization controller, delivers a noisy acceleration signal which is directly injected in the system via the feed-forward path. This fact limits the improvement of the described control law.

6.2 Future Work

Based on the experience gained during this work, some suggestions can be made for future research in humanoid robotics and for mechatronic systems in general.

One of the advantages of a decentralized topology as it is implemented in this project is the limitation of the system cabling. This is an aspect that must not be underestimated in particular for the system maintenance and in case of system failures. The distribution of information in the local controllers add a desirable level of abstraction to the physical system. Moreover, the latest developments in the micro-controller and FPGA industry could

enable an improvement in terms of quantity of data that can be exchanged and also for the control performance. On the other side, having a complete custom made solution for the decentralized structure also adds more complexity to the overall system development and to the software and hardware maintenance. Depending on the specific field of research, industry standard products can be a more viable solution.

For the sensor extension algorithms and, in general, for the data processing system, better performing algorithms can be implemented if the latest computing devices are used. Nowadays micro-controller, FPGA or System on Chip feature floating point units that can greatly improve the estimation algorithms or enable the use of more complex schemes like Kalman filter.

The joint modeling and model verification should be carried out before the system is completely built, giving the possibility of testing the robotic joints under different conditions.

Regarding the control of the joints, the proposed control laws have improved the performance of the whole robot. It must be said that the used model still contains many simplifications and does not directly consider the joints coupling and non-linear terms like variable inertia and gravity. To address these effects, the control can only be implemented in the CCU and not in the local controllers because of the missing information on the states of the other joints.

To reach high walking velocities with robust tracking performance, the feed-forward or feedback linearization approaches are still the most suitable. A necessary condition is the use of desired systems trajectories which are smooth enough to avoid noisy injection into the system. The generation of suitable smooth trajectories in real-time is not a trivial task in particular for systems with many DoFs.

Other possible solutions for local control algorithms with uncertain system parameters, are the implementation of adaptive techniques or robust control methods like H_∞ for non-linear systems.

Appendix A

Sensor Parameters

List of the sensor characteristics of LOLA.

Table A.1: incremental encoder list of encoder counts per second and maximum velocity

Joint	**Resolution** $[\frac{\text{counts}}{\text{revolution}}]$	**Max Velocity** $[rpm]$	**Max Frequency** $[Hz]$	**Max Counts Frequency** $[MHz]$
Head				
Convergence	800	10000	166.7	0.133
Pan	2000	25500	425	0.850
Tilt	2000	10000	166.7	0.333
Body				
Toe	4000	8500	141.7	0.567
Hip Flexion	11520	4800	80	0.922
Knee	11520	5600	93.3	1.075
Shoulder Adduction	11520	8000	133.3	1.536
Shoulder Flexion	11520	8000	133.3	1.536
Elbow	11520	8000	133.3	1.536
Pelvis Adduction	11520	8000	133.3	1.536
Hip Rotation	11520	8000	133.3	1.536
Hip Adduction	11520	8000	133.3	1.536
Ankle Adduction	11520	8500	141.7	1.632
Ankle Flexion	11520	8500	141.7	1.632

Table A.2: IMU iMAR iVRU-FC-C16 parameters

Characteristic	**Dimension**	**Value**
Update rate	$[Hz]$	200
Weight	$[kg]$	0.8
Angular rate		
Sensor range	$[deg/s]$	± 200
Short-term bias	$[deg/s]$	± 0.003
Long-term bias	$[deg/h]$	± 36
Scale error	$[\%]$	< 0.2
Linearity	$[\%]$	< 0.2
Resolution	$[deg/s]$	< 0.001
Linear acceleration		
Sensor range	$[g]$	± 2
Scale error	$[\%]$	< 0.3
Linearity	$[\%]$	< 0.3
Resolution	$[mg]$	< 0.1

Table A.3: EnDat parameters

Model	**Dimension**	**ECI 1116**	**ECI 1317**
Interface		EnDat 2.1	EnDat 2.1
Signal period per revolution		$65536(16Bit)$	$131072(17Bit)$
Mechanical Permissible speed	$[RPM]$	12000	12000
Position Computation Time	$[\mu s]$	8	8
Supply Voltage	$[V \pm 5\%]$	5	5
Rotor Inertia	$[kgm^2]$	0.7610^{-6}	3.210^{-6}
Weight	$[kg]$	0.1	0.13

Table A.4: FTS parameters

Maximum Measurable Forces		**Dimension**	**Value**
	F_x	$[N]$	500
	F_y	$[N]$	100
	F_z	$[N]$	1200
Maximum Measurable Torquex			
	M_x	$[Nm]$	100
	M_y	$[Nm]$	120
	M_z	$[Nm]$	50

Appendix B

Sercos Lola Custom Protocol Data

In the following tables the custom data structure for the SERCOS-III protocol used for LOLA is listed.

Table B.1: Sercos-III Standard IDNs: Device Control and State.

Name	IDN_dec	Description
wManC1D	129	Shutdown Error
wManC2D	181	Warning
wPrimarOpMode	32	POM[1] – IDLE
wSecOpMode1	33	SOM[2] 1 – Motor Free
wSecOpMode2	34	SOM 2 – Torque Mode
wSecOpMode3	35	SOM 3 – Velocity Mode
wSecOpMode4	284	SOM 4 – Position Inc Enc Mode
wSecOpMode5	285	SOM 5 – Position EnDat Mode
wSecOpMode6	286	SOM 6 – Position Double Loop Mode
wSecOpMode7	287	SOM 7 – Fault Mode
wResCtrl	134	Drive Control
wResStatus	135	Drive Status

Table B.2: Sercos-III custom IDN for Lola: Robot State Control.

Name	IDN_dec	Description
wDSPStateSVC	33268	DSC Current fsm[3] state (Service Channel)
IDSPCommandSVC	33368	DSC Command over Service Channel
wDSPManC1DExtraData1	34058	Extra error information

1 POM: Primary operation mode
2 SOM: Secondary operation mode
3 fsm:Finite State Machine
4 J: Joint
5 InEn: Incremental Encoder
6 EdEn: EnDat Encoder
7 LPF:Low Pass Filter
8 ff:feed forward

Table B.3: Sercos-III custom IDN for Lola: Joint Control Desired Values.

Name	IDN_dec	Description
lDbLolaPosJoint0_cmn	32869	Desired position J[4] 0 (Scaled)
lDbLolaPosJoint1_cmn	98405	Desired position J 1 (Scaled)
lDbLolaPosJoint2_cmn	163941	Desired position J 2 (Scaled)
lDbLolaPosMotort0_cmn	32870	Desired position Motor 0 (Scaled)
lDbLolaPosMotor1_cmn	98406	Desired position Motor 1 (Scaled)
lDbLolaPosMotor2_cmn	163942	Desired position Motor 2 (Scaled)
LolaPosEncMotElmo0_cmn	32868	InEn[5] desired position J 0
LolaPosEncMotElmo1_cmn	98404	InEn desired position J 1
LolaPosEncMotElmo2_cmn	163940	InEn desired position J 2
LolaPosEnDatEncoder0_cmn	32888	EdEn[6] desired position J 0
LolaPosEnDatEncoder1_cmn	98424	EdEn desired position J 1
LolaPosEnDatEncoder2_cmn	163960	EdEn desired position J 2
lDbLolaCurrentElmo0_cmn	32948	Desired Motor Current J 0
lDbLolaCurrentElmo1_cmn	98484	Desired Motor Current J 1
lDbLolaCurrentElmo2_cmn	164020	Desired Motor Current J 2

Table B.4: Sercos-III custom IDN for Lola: Encoders Measured Values.

Name	IDN_dec	Description
lDbLolaPosJoint0_fb	32879	Feedback postion J 0 (Scaled)
lDbLolaPosJoint1_fb	98415	Feedback position J 1 (Scaled)
lDbLolaPosJoint2_fb	163951	Feedback position J 2 (Scaled)
lDbLolaPosMotor0_fb	32880	Feedback postion Motor 0 (Scaled)
lDbLolaPosMotor1_fb	98416	Feedback position Motor 1 (Scaled)
lDbLolaPosMotor2_fb	163952	Feedback position Motor 2 (Scaled)
LolaPosEncMotElmo0_fb	32878	InEn measured position J 0
LolaPosEncMotElmo1_fb	98414	InEn measured position J 1
LolaPosEncMotElmo2_fb	163950	InEn measured position J 2
LolaPosEnDatEncoder0_fb	32898	EdEn measured position J 0
LolaPosEnDatEncoder1_fb	98434	EdEn measured position J 1
LolaPosEnDatEncoder2_fb	163970	EdEn measured position J 2
lDbLolaCurrentElmo0_fb	32949	Measured Motor Current J 0
lDbLolaCurrentElmo1_fb	98485	Measured Motor Current J 1
lDbLolaCurrentElmo2_fb	164021	Measured Motor Current J 2
lDbLolaVelEncMot0	32883	InEn Velocity J0
lDbLolaVelEncMot1	98419	InEn Velocity J1
lDbLolaVelEncMot2	163955	InEn Velocity J2
lDbLolaVelEncMotElmo0	32884	InEn Velocity from ELMO J0
lDbLolaVelEncMotElmo1	98420	InEn Velocity from ELMO J1
lDbLolaVelEncMotElmo2	163956	InEn Velocity from ELMO J2
LolaPosEncMotElmo0_offset	229486	InEn Offset position J 0
LolaPosEncMotElmo1_offset	295022	InEn Offset position J 1
LolaPosEncMotElmo2_offset	360558	InEn Offset position J 2
LolaPosEnDatEncoder0_offset	229506	EdEn Offset position J 0
LolaPosEnDatEncoder1_offset	295042	EdEn Offset position J 1
LolaPosEnDatEncoder2_offset	360578	EdEn Offset position J 2
lDbLolaCurrentElmo0_RatCurr	32950	Rated Motor Current J 0
lDbLolaCurrentElmo2_RatCurr	98486	Rated Motor Current J 1
lDbLolaCurrentElmo2_RatCurr	164022	Rated Motor Current J 2

Table B.5: Sercos-III custom IDN for Lola: IMU and FTS Measured Values.

Name	IDN_dec	Description
LolaFTS_fx	32908	Force torque Sensor: F_x
LolaFTS_fy	98444	Force torque Sensor: F_y
LolaFTS_fz	163980	Force torque Sensor: F_z
LolaFTS_Tx	229516	Force torque Sensor: T_x
LolaFTS_Ty	295052	Force torque Sensor: T_y
LolaFTS_Tz	360588	Force torque Sensor: T_z
LolaSpindelDef_adc_0	32913	Spindle Deformation Data: ADC_0
LolaSpindelDef_adc_1	229521	Spindle Deformation Data: ADC_1
LolaIMU_nrpy_x	32918	IMU angle x
LolaIMU_nrpy_y	98454	IMU angle y
LolaIMU_nrpy_z	163990	IMU angle z
LolaIMU_omgs_x	229526	IMU angle velocity x
LolaIMU_omgs_y	295062	IMU angle velocity y
LolaIMU_omgs_z	360598	IMU angle velocity z
LolaIMU_accs_x	426134	IMU angle acceleration x
LolaIMU_accs_y	491670	IMU angle acceleration y
LolaIMU_accs_z	557206	IMU angle acceleration z
LolaIMU_status	622742	IMU status word

Table B.6: Sercos-III custom IDN for Lola: Controller Variables.

Name	IDN_dec	Description
LolaController_GG	32928	Controller Global Gain
LolaPosition_PID_KP_0	32929	Position Proportional gain J 0
LolaPosition_PID_KP_1	98465	Position Proportional gain J 1
LolaPosition_PID_KP_2	164001	Position Proportional gain J 2
LolaPosition_PID_KD_0	32930	Position Derivative gain J 0
LolaPosition_PID_KD_1	98466	Position Derivative gain J 1
LolaPosition_PID_KD_2	164002	Position Derivative gain J 2
LolaPosition_PID_KD_LPF_D_f_0	229538	Position Derivative LPF[7] J 0
LolaPosition_PID_KD_LPF_D_f_1	295074	Position Derivative LPF J 1
LolaPosition_PID_KD_LPF_D_f_2	360610	Position Derivative LPF J 2
LolaPosition_FF_0	32931	Position ff[8] current value J 0
LolaPosition_FF_1	98467	Position ff current value J 1
LolaPosition_FF_2	164003	Position ff current value J 2
LolaPosition_FF_gain_0	229539	Position ff gain J 0
LolaPosition_FF_gain_1	295075	Position ff gain J 1
LolaPosition_FF_gain_2	360611	Position ff gain J 2
LolaPosition_FF_mult	426147	Position ff multiplicator
LolaController_Position_err_IncEnc_0	32932	InEn current position error J 0
LolaController_Position_err_IncEnc_1	98468	InEn current position error J 1
LolaController_Position_err_IncEnc_2	164004	InEn current position error J 2
LolaController_Position_err_Endat_0	32933	EdEn current position error J 0
LolaController_Position_err_Endat_1	98469	EdEn current position error J 1
LolaController_Position_err_Endat_2	164005	EdEn current position error J 2

Table B.7: Sercos-III custom IDN for Lola: Utility Variables.

Name	IDN_dec	Description
wLolaCameraTrig	32938	Camera Trigger Information
wDSPCameratrigFreq	32939	Camera Trigger Frequency
DSP_Sercosiface_SW_Ver	36768	Current Sercos-III interface Version bit 31...16 : SERCOS_VERSION bit 15...0 : USER_DB_VERSION
lCycle_Timers	34818	DSC Timer Measures

Table B.8: Sercos custom Commands: DSC State bit meaning.

	wDSPStateSVC	
Bit	**Name**	**Meaning**
31	reserved	
30	reserved	
29	reserved	
28	reserved	
27	reserved	
26	reserved	
25	reserved	
24	reserved	
23	reserved	
22	reserved	
21	reserved	
20	DSPStateSVC_Elmo_vel_fb	Elmo velocity Feedback set
19	DSPStateSVC_Current_fb	Current Feedback set
18	DSPStateSVC_IncEnc_fb	Incremental Encoder Feedback set
17	DSPStateSVC_Endat_fb	EnDat Encoder Feedback set
16	DSP_Warning	DSP warning
15...12	DSP_State	
		DSP_IDLE
		DSP_BOOT
		DSP_MFREE
		DSP_TCONT
		DSP_VCONT
		DSP_PCONT
		DSP_ELMO_PCONT
		DSP_FAULT
		DSP_MOTOR_ALIGNMENT
		DSP_START_UP
11	Motor Aligned	Motors have been aligned
10	New Desired Value	New desired value available
9	Joint_Offset_2_Set	Inc Enc 2 offset have been set
8	Joint_Offset_1_Set	Inc Enc 1 offset have been set
7	Joint_Offset_0_Set	Inc Enc 0 offset have been set
6	Error	DSP error
5	Error Extra data Available	Extra Erro data available
4	IMU OK	IMU status OK
3	KMS OK	KMS status OK
2	EnDats OK	EnDat status OK
1	Inc Encs Ok	Enc Encstatus OK
0	Elmos Ok	ELMO status OK

Table B.9: Sercos custom Commands: DSC Command Array.

	IDSPCommandSVC[0]		
Bit	**Name**	**Command**	**Values**
31...16	SVC_Command		
		goto_idle_NRT	0x8000
		goto_mON	0x8001
		reset_fts	0x8002
		reset_imu	0x8003
		reboot	0x8004
		elmo_bc	0x8005
		set_position_offset	0x8006
		head_homing	0x8007
		set_elmo_ratedCurrent	0x8008
		set_endat_fb	0x8101
		reset_endat_fb	0x8102
		set_current_fb	0x8103
		reset_current_fb	0x8104
15...0	Elmo_index		0-2

	IDSPCommandSVC[1]
31...24	BC_ch1 / OFFSET_ELMO_0
23...16	BC_ch2 / OFFSET_ELMO_0
15...8	BC_Idx / OFFSET_ELMO_0
7...0	BC_Int_flt / OFFSET_ELMO_0

	IDSPCommandSVC[2]
31...0	BC_DataIDX / OFFSET_ELMO_1

	IDSPCommandSVC[3]
31...0	OFFSET_ELMO_2

Table B.10: Sercos custom Commands: Camera Trigger.

	wLolaCameraTrig
Bit	**Name**
15...8	Camera Trigger Actual Level
7...0	Camera Trigger have been set the Last Cycle

Appendix C

DSCB Application Program Interface

List and definition of the software API developed for the DSCBs controllers:

Table C.1: DSC List of API files.

Function Group	FIle Name	Description
DSC Configuration and System management	dsp_config.c	DSC register configuration
	config.h	DSCB configuration file
	PE_Const.h	Generic constant definitions
	PE_Error.h	Generic error definitions
	PE_Types.h	Generic data type definition
	IO_Map.h	DSC peripheral address definition
	fsm.h	FSM basic definitions
	SM.h	FSM state and state change definition
	DSPHW_FSM.c	DSCB fsm implementation
	DSPHW_FSM.h	
Communication	FlexCAN.c	CAN low level communication
	FlexCAN.h	
	iface_External_RAM.c	External RAM low level communication
	iface_External_RAM.h	
	iface_sercos_lola.h	Sercos-III data/macro definition
Sensor Driver	IMU.c	IMU communication implementation
	IMU.h	
	KMSAPI.c	FTS communication implementation and low level SPI communication
	KMSAPI.h	
	inc_enc.c	incremental encoder communication implementation and low level FPGA communication
	inc_enc.h	
	endat.c	EnDat communication implementation and low level EMI communication
	endat.h	
	sens_iface.c	Sensor read, elaboration functions
	sens_iface.h	

Function Group	FIle Name	Description
Actuator Driver	ElmoAPI.c	ELMO CANOpen Communication implementation
	ElmoAPI.h	
Joint Control	Controller.h	Control functions
	Controller.c	
Spindle kinematics	lola_ankle_joint_left_int32.c	Left Ankle look-up table
	lola_ankle_joint_left_int32.h	
	lola_ankle_joint_right_int32.c	Right Ankle look-up table
	lola_ankle_joint_right_int32.h	
	lola_joint_kinematics_tabular.c	
	lola_joint_kinematics_tabular.h	
	lola_knee_joint_left_int32.c	Left Knee look-up table
	lola_knee_joint_left_int32.h	
	lola_knee_joint_right_int32.c	Right Knee look-up table
	lola_knee_joint_right_int32.h	
	lola_vis_conv_joint.c	Head Convergence look-up table
	lola_vis_conv_joint.h	
	lola_dsp_kinematics.c	Look-up table search functions
	lola_dsp_kinematics.h	
Utility and debug	file_backup.c	Save data remotely to file
	file_backup.h	
	time_measure.c	Timing measure functions
	time_measure.h	
	timers.c	DSC timers API
	timers.h	
	IRQ_DSP.c	DSC Interrupt Request functions
	IRQ_DSP.h	
Safety and Error management	config.h	Error management macros
	ElmoAPI.c	Safety functions
	ElmoAPI.h	

Appendix D

Incremental Encoder IP-Core

The following table lists the communication interface between the DSC and the FPGA Incremental Encoder IP-Core.

Table D.1: Incremental Encoder IP-Core: DSC Interface Signals Description.

Signal Name	I/O	Signal Description
ADD_Bus	In	Address Bus
DATA_Bus	In/Out	Data Bus
CK	In	System Clock
RST	In	Reset
$\overline{CS}$	In	Chip Select
$\overline{Rd}$	In	Read Signal
$\overline{WR}$	In	Write Signal
Incremental Encoder		
A_n1	In	A Negative Signal Encoder 1
B_n1	In	B Negative Signal Encoder 1
A_p1	In	A Negative Signal Encoder 1
B_p1	In	B Negative Signal Encoder 1
A_n2	In	A Negative Signal Encoder 2
B_n2	In	B Negative Signal Encoder 2
A_p2	In	A Negative Signal Encoder 2
B_p2	In	B Negative Signal Encoder 2
A_n3	In	A Negative Signal Encoder 3
B_n3	In	B Negative Signal Encoder 3
A_p3	In	A Negative Signal Encoder 3
B_p3	In	B Negative Signal Encoder 3
LED Outputs		
LED_0	Out	Bit 0 of Encoder 1
LED_1	Out	Bit 0 of Encoder 2
LED_2	Out	Bit 0 of Encoder 3
LED_Power	Out	FPGA Power LED

The following tables list the internal communication interfaces of the units of the FPGA Incremental Encoder IP-Core.

Table D.2: Incremental Encoder IP-Core: Debounce Module Signals Description.

Signal Name	I/O	Signal Description
clk	In	Unit Clock
reset	In	Reset
raw	In	Raw Encoder Signal Input
PreLoadPort	In	Low Pass Filter Value
Debounced	Out	Filtered Encoder Signal

Table D.3: Incremental Encoder IP-Core: Encoder Decoder Module Signals Description.

Signal Name	I/O	Signal Description
clk	In	Unit Clock
a_reg	In	A Filtered signal
b_reg	In	B Filtered signal
reset	In	Reset
counter_CK	In	New Count Available
direction	Out	Encoder Current Moving Direction

Table D.4: Incremental Encoder IP-Core: Encoder Position Counter Module Signals Description.

Signal Name	I/O	Signal Description
clk	In	Unit Clock
ce	In	Chip Select
reset	In	Reset
Inverted_Dir	In	Inverted or Not Counting Direction
PreLoad	In	Counter Offset Trigger
PreLoadPort	In	Counter Offset
direction	In	Encoder Movement Direction
count	Out	Encoder Current Count

Table D.5: Incremental Encoder IP-Core: Encoder Status Register Bit.

Bit	Bit Name	Bit Description
32...4		Reserved
3	CD	Counting Direction
2	OF	Set Encoder Offset
1	LP	Set Low Pass Filter
0	SR	Software Reset

Table D.6: Incremental Encoder IP-Core: Velocity Module Signals Description.

Signal Name	I/O	Signal Description
clk	In	Unit Clock
reset	Out	Unit Reset
trig_comp	In	Trigger Velocity computation
vel_comp	Out	Velocity Computation running
position_0	In	Position From Encoder 1
position_1	In	Position From Encoder 2
position_2	In	Position From Encoder 3
velocity	Out	Velocity
pos_rd_EN	In	Enable Position Read
comp_module	In	Module Number

Table D.7: Incremental Encoder Module FPGA: Internal Unit Memory Mapping.

Unit	Memory Address	Address Space
Incremental Encoder 1	0x000-0x00F	0x00F
Incremental Encoder 2	0x010-0x01F	0x00F
Incremental Encoder 3	0x020-0x02F	0x00F
CAN Module 1	0x080-0x0C0	0x040
CAN Module 2	0x100-0x140	0x040
CAN Module 3	0x180-0x1C0	0x040

Table D.8: Incremental Encoder IP-Core: CAN Module Signals Description.

Signal Name	I/O	Signal Description
can_phy_tx	Out	CAN bus transmit signal to PHY
can_phy_rx	In	CAN bus receive signal from PHY
Bus2IP_DATA	In	Write Data bus
IP2Bus_DATA	OUT	Read Data bus
Bus2IP_RNW	In	Read or Write signaling
Bus2IP_ADD,	In	Address Bus
CAN Control Signals		
CLK0_BUF,	In	Input interface clock
CK_CAN,	In	24 MHz oscillator clock input
Bus2IP_RST,	In	Active high reset
Bus2IP_CS	In	Active high CS
IP2Bus_ACK	OUT	R/W data acknowledgment
IP2Bus_IntrEvent	OUT	Active high interrupt line1
IP2Bus_Error	OUT	Active high R/W Error signal

Appendix E

Joint Model Catalog Parameters

List of the gear model parameter for different Harmonic-Drive gears and Roller screws[1].

Table E.1: Harmonic Drive Stiffness Model.

Type	τ_{e1} $[Nm]$	τ_{e2} $[Nm]$	K_0 $[\frac{Nm}{rad}]$	K_1 $[\frac{Nm}{rad}]$	K_2 $[\frac{Nm}{rad}]$
HFUC-11-100	0.8	2.0	2.7e3	3.4e3	4.4e3
HFUC-14-100	2	6.9	0.47e4	0.61e4	0.71e4
HFUC-17-100	3.9	12.0	1.0e4	1.4e4	1.6e4
HFUC-20-100	7.0	25.0	1.6e4	2.5e4	2.9e4
HFUC-25-100	14.0	48.0	3.1e4	5.0e4	5.7e4
HFUC-32-50	29.0	108.0	5.5e4	6.3e4	7e4

Table E.2: Harmonic Drive Friction Model.

Type	b_v $[\frac{Nms^2}{rad}]$	τ_{f0} $[Nm]$	μ $[/]$	γ $[\frac{s^2}{rad}]$	Gear Ratio N $[/]$
HFUC-11-100	$6.63 \cdot 10^{-6}$	$7.75 \cdot 10^{-3}$	0.0	$6.59 \cdot 10^{-4}$	100
HFUC-14-100	$1.04 \cdot 10^{-5}$	$1.21 \cdot 10^{-2}$	0.0	$6.59 \cdot 10^{-4}$	100
HFUC-17-100	$3.20 \cdot 10^{-5}$	$3.72 \cdot 10^{-2}$	0.0	$6.59 \cdot 10^{-4}$	100
HFUC-20-100	$5.33 \cdot 10^{-5}$	$6.20 \cdot 10^{-2}$	0.0	$6.58 \cdot 10^{-4}$	100
HFUC-25-100	$8.94 \cdot 10^{-5}$	$1.04 \cdot 10^{-1}$	0.0	$6.58 \cdot 10^{-4}$	100
HFUC-32-50	$1.52 \cdot 10^{-4}$	$2.46 \cdot 10^{-1}$	$5.49 \cdot 10^{-3}$	$6.21 \cdot 10^{-}4$	50

1 Data from Buschmann [20]

Table E.3: Roller screw Friction Model.

Part	η_D	η_I
Knee joint roller screw	0.860	0.858
Ankle joint roller screw	0.855	0.839
Ankle joint timing belt	0.98	0.98

List of Abbreviations

ADC Analogue to Digital Converter

API Application Programming Interface

AU Actuation Unit

CAD Computer Aided Design

CAN Control Area Network

CCL Communication Control Logic unit

CCU Central Control Unit

CPLD Complex Programmable Logic Device

DAC Digital to Analogue Converter

DoF Degrees of Freedom

DSB Distributed Sensor Control Board

DSC Digital Signal Controller

DSCB Distributed Sensor Control Board

DSP Digital Signal Processor

EEPROM Electric Erasable Programmable Read-Only Memory

FIR Finite Input Response

FPGA Field Programmable Gate Array

FPU Floating Point Unit

FTS Force Torque Sensor

HD Harmonic Drive (gear)

HLC High Level Control

I/O Input Output

IEF Incremental Encoder FPGA

IMU Inertial Measurement Unit

IRR Infinite Input Response

LLC Low Level Control

LSE Least Square Estimation

MT Multichannel Connectors

PCI Peripheral Component Interconnect

PHY Physical Layer

PID Proportional Integral Derivative controller

rms Root Mean Square

SPI Serial Communication Interface

VPS Video Processing Server

Bibliography

[1] Boston Dynamics Atlas - the agile anthropomorphic robot. URL http://www.bostondynamics.com/robot_Atlas.html.

[2] Cannon Automata. URL http://www.cannon-automata.com/.

[3] Elmo Motion Control. URL http://www.elmomc.com.

[4] Harmonic Drive gears. URL www.harmonic-drive.de.

[5] Heidenhain Absolute Encoders. URL http://www.heidenhain.de.

[6] Boston Dynamics PETMAN. URL http://www.bostondynamics.com/robot_petman.html.

[7] Sensitec Incremental Encoders. URL http://www.sensitec.com.

[8] Sercos International. URL http://www.sercos.org/.

[9] Basler. URL http://www.baslerweb.com.

[10] iMAR Navigation. URL www.imar-navigation.de.

[11] R. Al Ashoor, K. Khorasani, R. Patel, and A. Al-Khalili. Adaptive control of flexible joint manipulators. *IEEE International Conference on Systems, Man and Cybernetics*, pages 627 – 632, 1990.

[12] A. Albu-Schäffer and G. Hirzinger. Parameter identification and passivity based joint control for a 7 DoF torque controlled light weight robot. *IEEE International Conference on Robotics and Automation (ICRA)*, 3:2852 – 2858, 2001.

[13] T. Asfour, K. Regenstein, P. Azad, J. Schroder, A. Bierbaum, N. Vahrenkamp, and R. Dillmann. Armar-iii: An integrated humanoid platform for sensory-motor control. *IEEE-RAS International Conference Humanoid Robot (Humanoids)*, pages 169 – 175, 2006.

[14] P. W. Bayside Motion 27 Seaview Boulevard. *Product Manual: Frameless Kit Motors. Rev. 2.0 / 1100*, 2008.

[15] P. R. Belanger. Estimation of angular velocity and acceleration from shaft encoder measurements. *IEEE International Conference on Robotic and Automation (ICRA)*, 1: 585 – 592, 1992.

[16] M. Bodson, J. Chiasson, and N. R. Nonlinear speed observer for high-performance induction motor control. *IEEE Transactions on Industrial Electronics*, 42(4):337 – 343, 1995.

[17] L. Bonometti. *Convertitori di potenza e servomotori brushless*. (Italian) Editoriale Delfino, second edition, 2001.

[18] B. Brogliato, R. Ortega, and R. Lozanos. Global tracking controllers for flexible-joint manipulators. *Automatica a Comparative Study.*, pages 941–956, 1995.

[19] R. Brown, S. Schneider, and M. Mulligan. Analysis of algorithms for velocity estimation from discrete position versus time data. *IEEE Transaction on Industrial Electronics*, 39(1):11 – 19, 1992.

[20] T. Buschmann. *Simulation and Control of Biped Walking Robots*. PhD thesis, Technischen Universität München, Lehrstuhl für Angewandte Mechanik Fakultät für Maschinenwesen, 2010.

[21] T. Buschmann, S. Lohmeier, H. Ulbrich, and F. Pfeiffer. Dynamics simulation for a biped robot: Modeling and experimental verification. *IEEE International Conference in Robotics and Automation (ICRA)*, pages 2673 – 2678, 2006.

[22] T. Buschmann, S. Lohmeier, M. Bachmayer, H. Ulbrich, and F. Pfeiffer. A collocation method for real-time walking pattern generation. *IEEE-RAS International Conference Humanoid Robot. (Humanoids)*, pages 1 – 6, 2007.

[23] T. Buschmann, S. Lohmeier, and H. Ulbrich. Biped walking control based on hybrid position/force control in intelligent robots and systems. *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 3019 – 3024, 2009.

[24] T. Buschmann, V. Favot, S. Lohmeier, M. Schwienbacher, and H. Ulbrich. Experiments in fast biped walking. *IEEE International Conference on Mechatronics (ICM)*, pages 863 – 868, 2011.

[25] T. Buschmann, M. Schwienbacher, V. Favot, A. Ewald, and H. Ulbrich. The biped walking robot Lola – hardware design and walking control –. *Journal of the Robotics Society of Japan*, 30, 2012.

[26] T. Buschmann, M. Schwienbacher, V. Favot, A. Ewald, and H. Ulbrich. Dynamics and control of the biped robot Lola. *Multibody System Dynamics, Robotics and Control.*, pages 161 – 173, 2012.

[27] CAN. *CiA 301 V4.0 - CANopen application layer and communication profile*. CAN in Automation (CiA), Oct. 2002.

[28] CAN. *CAN Specification 2.0 Part A*. CAN in Automation (CiA), Oct. 2007.

[29] CAN. *CAN Specification 2.0 Part B*. CAN in Automation (CiA), Jan. 2007.

[30] G. Carbone, H. Lim, A. Takanishi, and M. Ceccarelli. Stiffness analysis of the humanoid robot wabian-riv: modelling. *IEEE International Conference on Robotics and Automation (ICRA)*, 3:3654 – 3659, 2003.

[31] P. S. Carpenter, R. H. Brown, J. A. Heinen, and S. Schneider. On algorithms for velocity estimation using discrete position encoders. *IEEE Industrial Electronics, Control, and Instrumentation*, 2(2):844 – 849, 1995.

[32] H. I. Christensen and G. D. Hager. Sensing and and estimation. In B. Siciliano and O. Khatib, editors, *Springer Handbook of Robotics*, chapter 4, pages 133–159. Springer Berlin Heidelberg, May 2008.

[33] W. Chung, L. Fu, and S. Hsu. Motion control. In B. Siciliano and O. Khatib, editors, *Springer Handbook of Robotics*, chapter 6, pages 133–159. Springer Berlin Heidelberg, May 2008.

[34] E. Clarke. *Circuit Analysis of AC Power Systems. Vol. I.* Wiley, first edition, 1943.

[35] J. L. Crassidis and J. L. Junkins. *Optimal Estimation of Dynamic Systems*. CRC Press, second edition, 2011.

[36] A. De Luca. Feedforward/feedback laws for the control of flexible robots. *IEEE International Conference on Robotics and Automation (ICRA)*, 1:233 – 240, 2000.

[37] A. De Luca. Robots with elastic joints: Modeling and control. 2003. Scuola di Dottorato CIRA Controllo di Sistemi Robotici per la Manipolazione e la Cooperazione.

[38] A. De Luca and W. Book. Robots with flexible elements. In B. Siciliano and O. Khatib, editors, *Springer Handbook of Robotics*, chapter 13, pages 287–319. Springer Berlin Heidelberg, May 2008.

[39] A. De Luca and P. Lucibello. A general algorithm for dynamic feedback linearization of robot with elastic joints. *IEEE International Conference on Robotics and Automation (ICRA)*, May 1998.

[40] M. Diftler, J. Mehling, M. Abdallah, N. Radford, L. Bridgwater, A. Sanders, R. Askew, D. Linn, J. Yamokoski, F. Permenter, B. Hargrave, R. Piatt, R. Savely, and R. Ambrose. Robonaut 2 - the first humanoid robot in space. *Conference on Robotics and Automation (ICRA)*, pages 2178 – 2183, 2011.

[41] R. Dutta and B. Kumar. On digital differentiators, hilbert transformers, and half-band low-pass filters. *IEEE Transactions on Education*, 32(3):314 – 318, 1989.

[42] DS 402. *Elmo Motion Control CANopen DS 402 Implementation Guide*. Elmo Motion Control, 2004.

[43] *SimplIQ Software Manual*. Elmo Motion Control, 2004.

[44] DS 301. *Elmo Motion Control CANopen DS 301 Implementation Guide*. Elmo Motion Control, 2006.

[45] *Composer User Manual for SimplIQ Servo Drives*. Elmo Motion Control, 2007.

[46] *SimplIQ Command Reference Manual*. Elmo Motion Control, 2007.

[47] J. Englsberger, A. Werner, C. Ott, B. Henze, M. Roa, G. Garofalo, R. Burger, A. Beyer, O. Eiberger, K. Schmid, and A. Albu-Schaffer. Overview of the torque-controlled humanoid robot toro. *IEEE-RAS International Conference on Humanoid Robots (Humanoids)*, pages 916 – 923, 2014.

[48] V. Favot, M. Schwienbacher, T. Buschmann, S. Lohmeier, and H. Ulbrich. The humanoid robot Lola - experimental results. *International Conference of Numerical Analysis and Applied Mathematics (ICNAAM)*, 1(1281):398–401, 2010.

[49] V. Favot, T. Buschmann, M. Schwienbacher, A. Ewald, and H. Ulbrich. The sensor-controller network of the humanoid robot Lola. *IEEE-RAS International Conference Humanoid Robot (Humanoids)*, pages 805 – 810, 2012.

[50] F. Flacco, A. De Luca, I. Sardellitti, and N. Tsagarakis. Robust estimation of variable stiffness in flexible joints. *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 4026 – 4033, 2011.

[51] F. Flacco, A. De Luca, I. Sardellitti, and N. G. Tsagarakis. Residual-based stiffness estimation in robots with flexible transmissions. *IEEE International Conference on Robotics and Automation (ICRA)*, pages 5541 – 5547, 2011.

[52] B. Fornberg. Generation of finite difference formulas on arbitrarily spaced grids. *Mathematics of Computation*, 51(184):699 – 706, 1988.

[53] E. Galvan, A. Torralba, and L. G. Franquelo. Asic implementation of a digital tachometer with high precision in a wide speed range. *IEEE Transaction on Industrial Electronics*, 43(6):655 – 660, 1996.

[54] C. F. Gauss. *Theory of Motion of the Heavenly Bodies Moving About the Sun in Conic Sections, A Translation of Theoria Motus*. Dover, 1963.

[55] M. Gautier, A. Janot, A. Jubien, and P. Vandanjon. Joint stiffness identification from only motor force/torque data. *IEEE Conference on Decision and Control and European Control Conference (CDC-ECC)*, pages 5088 – 5093, 2001.

[56] C. F. Gerald. *Applied Numerical Analysis*. Addison-Wesley, third edition, 1970.

[57] F. Ghorbel, J. Y. Hung, and M. W. Spong. Adaptive control of flexible-joint manipulators. *IEEE International Conference on Decision and Control*, 2:1188 – 1193, 1990.

[58] M. Gienger. *Entwurf und Realisierung einer zweibeinigen Laufmaschine*. Fortschritt-Berichte VDI: Reihe 1, Konstruktionstechnik, Maschinenelemente (German). VDI-Verlag, 2005.

[59] M. Grotjahn and B. Heimann. Model-based feedforward control in industrial robotics. *The International Journal of Robotics Research*, 21:45 – 60, 2002.

[60] M. Grotjahn, M. Daemi, and B. Heimann. Friction and rigid body identification of robot dynamics. *The International Journal of Solids and Structures*, 38:1889 – 1902, 2001.

[61] K. A. Grunauer Brachetti. Control of flexible joints for humanoid robots. Master's thesis, Technischen Universität München, Lehrstuhl für Angewandte Mechanik Fakultät für Maschinenwesen, 2012.

[62] B. Habibullah, H. Singh, K. L. Soo, and L. C. Ong. A new digital speed transducer. *IEEE Transaction on Industrial Electronics and Instrumentation*, ECI-25(4):339 – 342, 1978.

[63] V. Hagenmeyer. *Robust nonlinear tracking control based on differential flatness*. PhD thesis, Laboratoire des Signaux et Systèmes, C.N.R.S. Supèlec-Universitè Paris Sud, 2002.

[64] R. H. Hamming. *Numerical Methods for Scientists and Engineers*. Dover, second edition, 1986.

[65] R. H. Hamming. *Digital Filters*. Dover, third edition, 1997.

[66] T. Hardeman. *Modelling and Identification of Industrial Robots including Drive and Joint Flexibilities*. PhD thesis, Netherlands Institute for Metal Research, 2008.

[67] *EnDat Interface Version 2.2 Bidirectional Syncronous-Serial Interface for Position Encoders*. Heidenhain, specification release f93889 edition, 2007.

[68] G. Hirzinger, N. Sporer, A. Albu-Schäffer, M. Hähnle, R. Krenn, A. Pascucci, and M. Schedl. DLR's torque-controlled light weight robot III - are we reaching the technological limits now? *IEEE International Conference in Robotics and Automation (ICRA)*, 2:1710 – 1716, 2002.

[69] J. D. Hoffman. *Numerical Methods for Engineers and Scientists*. Marcel Dekker, Inc. Prentice Hall, second edition, 2001.

[70] J. Hollerbach, W. Khalil, and M. Gautier. Model identification. In B. Siciliano and O. Khatib, editors, *Springer Handbook of Robotics*, chapter 14, pages 321–344. Springer Berlin Heidelberg, May 2008.

[71] P. Ioannou and J. Sun. *Robust Adaptive Control*. Prentice-Hall, 1996.

[72] S. Jörg, M. Nickl, A. Nothhelfer, T. Bahls, and G. Hirzinger. The computing and communication architecture of the DLR hand arm system. *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 1055 – 1062, 2011.

[73] S. Kajita and B. Espiau. Legged robots. In B. Siciliano and O. Khatib, editors, *Springer Handbook of Robotics*, chapter 16, pages 361–389. Springer Berlin Heidelberg, May 2008.

[74] R. E. Kalman. A new approach to linear filtering and prediction problems. *Transactions of the ASME-Journal of Basic Engineering*, D(82):35 – 45, 1960.

[75] K. Kaneko and R. Horowitz. Repetitive and adaptive control of robot manipulators with velocity estimation. *IEEE Transaction on robotics and Automation (ICRA)*, 13 (2):204 – 217, 1997.

[76] K. Kaneko, F. Kanehiro, S. Kajita, H. Hirukawa, T. Kawasaki, M. Hirata, K. Akachi, and T. Isozumi. Humanoid robot HRP-2. *IEEE International Conference in Robotics and Automation (ICRA)*, 2:1083 – 1090, 2004.

[77] K. Kaneko, K. Harada, F. Kanehiro, G. Miyamori, and K. Akachi. Humanoid robot HRP-3. *IEEE International Conference in Robotics and Automation (ICRA)*, pages 2471 – 2478, 2006.

[78] R. Kavanagh. Signal processing techniques for improved digital tachometer. *IEEE International Symposium on Industrial Electronics*, 2:511 – 517, 2002.

[79] C. Kemp, P. Fitzpatrick, H. Hirukawa, K. Yokoi, K. Harada, and Y. Matsumoto. Humanoids. In B. Siciliano and O. Khatib, editors, *Springer Handbook of Robotics*, chapter 56, pages 1307–1333. Springer Berlin Heidelberg, May 2008.

[80] N. Kircanski, A. Goldenberg, and S. Jia. An experimental study of nonlinear stiffness, hysteresis and friction effects in robot joints with harmonic drives and torque sensors. *International Symposium on Experimental Robotics*, 200:326 – 340, 1993.

[81] E. Kreyszig. *Advanced Engineering Mathematics*. John Wiley & Sons Inc, fifth edition, 1983.

[82] L. Le Tien, A. Albu-Schäffer, A. De Luca, and G. Hirzinger. Friction observer and compensation for control of robots with joint torque measurement. *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 3789 – 3795, 2008.

[83] K. Lee, H. J. Yim, S. Jang, Y. Kang, Y. You, and T. W. Park. A study on joint compliance for a biped robot. *IEEE-RAS International Conference on Humanoid Robots*, 2005.

[84] F. Lewis, D. Dawson, and C. Abdallah. *Robot Manipulator Control: Theory and Practice*. CRC Press, second edition, 2003.

[85] H. D. A. Limburg/Lahn. *Harmonic Drive Catalog*, 2007.

[86] K. Löffler. *Dynamik und Regelung einer zweibeinigen Laufmaschine*. Fortschritt-Berichte VDI: Reihe 8, Meß-, Steuerungs- und Regelungstechnik (German). VDI-Verlag, 2006.

[87] K. Löffler, M. Gienger, and F. Pfeiffer. Sensors and control concept of a biped robot. *IEEE Transactions on Industrial Electronics*, 51(5):972 – 980, 2004.

[88] S. Lohmeier. *Design and Realization of a Performance Enhanced Humanoid Robot*. PhD thesis, Technischen Universität München, Lehrstuhl für Angewandte Mechanik Fakultät für Maschinenwesen, 2010.

[89] S. Lohmeier, T. Buschmann, M. Schwienbacher, H. Ulbrich, and F. Pfeiffer. Leg design for a humanoid walking robot. *IEEE-RAS International Conference Humanoid Robot. (Humanoids)*, pages 536 – 541, 2006.

[90] S. Lohmeier, T. Buschmann, H. Ulbrich, and F. Pfeiffer. Modular joint design for performance enhanced humanoid robot Lola. *IEEE International Conference in Robotics and Automation (ICRA)*, 2006.

[91] R. Lozano and B. Brogliato. Adaptive control of robot manipulators with flexible joints. *IEEE Transaction on Automation Control*, 37, 1992.

[92] D. Luenberger. Observing the state of a linear system. *IEEE Transactions on Military Electronics*, 8(2):74 – 80, 1964.

[93] R. G. Lyons. *Understanding digital signal processing*. Pearson Education Prentice Hall, third edition, 2010.

[94] R. Marino and M. W. Spong. Nonlinear control techniques for flexible joint manipulators: A single link case of study. *IEEE International Conference on Robotics and Automation*, 3:1030 – 1036, 1986.

[95] R. Martinez-Guerra, A. Poznyak, E. Gortcheva, and V. Diaz de Leon. Robot angular link velocity estimation in the presence of high-level mixed uncertainties. *IEEE Proceedings in Control Theory and Applications*, 147(5):515 – 522, 2001.

[96] *EnDat 2.2 - Softmacro For Master Component*. MAZeT GmbH, 2007.

[97] S. Moberg. *Modeling and Control of Flexible Manipulators*. PhD thesis, Department of Electrical Engineering Linköping University, Sweden, 2008.

[98] N. Mohan, T. M. Undeland, and W. P. Robbins. *Power Electronics: Converters, Applications, and Design*. Wiley, second edition, 1995.

[99] Y. Nakanishi, Y. Namiki, K. Hongo, J. Urata, I. Mizuuchi, and M. Inaba. Design of the musculoskeletal trunk and realization of powerful motions using spines. *IEEE International Conference on Humanoid Robots (Humanoids)*, pages 96 – 101, 2007.

[100] S. Nicosia and P. Tomei. Robot control by using only measurements joint position. *IEEE Transcation on Automatic Control*, 35(6):1058 – 1061, 1990.

[101] K. Nishiwaki, S. Kagami, J. Kuffner, M. Inaba, and H. Inoue. Online humanoid walking control system and a moving goal tracking experiment. *IEEE International Conference on Robotics and Automation (ICRA)*, 1:911 – 916, 2003.

[102] K. Nishiwaki, J. Kuffner, S. Kagami, M. Inaba, and H. Inoue. The experimental humanoid robot h7: a research platform for autonomous behavior. *Philosophical Transaction of the Royal Society. A 365*, pages 79 – 107, 2006.

[103] Y. Ogura, H. Aikawa, K. Shimomura, A. Morishima, H.-O. Lim, and A. Takanishi. Development of a new humanoid robot wabian-2. *IEEE International Conference in Robotics and Automation (ICRA)*, pages 76 – 81, 2006.

[104] K. Otani and T. Kakizaki. Motion planning and modeling for accurately identifying dynamic parameters of an industrial robotic manipulator. *IEEE Transactions on Robotics and Automation*, 13:743 – 748, 1993.

[105] C. Ott, O. Eiberger, W. Friedl, B. Bauml, U. Hillenbrand, C. Borst, A. Albu-Schaffer, B. Brunner, H. Hirschmuller, S. Kielhofer, R. Konietschke, M. Suppa, T. Wimbock, F. Zacharias, and G. Hirzinger. A humanoid two-arm system for dexterous manipulation. *IEEE-RAS International Conference Humanoid Robot (Humanoids)*, pages 276 – 283, 2006.

[106] C. Ott, C. Baumgartner, J. Mayr, M. Fuchs, R. Burger, O. Dongheui Leeand Eiberger, A. Albu-SchaÌĹffer, M. Grebenstein, and G. Hirzinger. Development of a biped robot with torque controlled joints. *IEEE-RAS International Conference Humanoid Robot (Humanoids)*, pages 167 – 173, 2010.

[107] S. Ozgoli and H. D. Taghirad. A survey on the control of flexible joint robots. *Asian Journal of Control*, 8:1 – 15, 2006.

[108] R. H. Park. Two-reaction theory of synchronous machines: Generalized method of analysis - part i. *Transactions of the AIEE*, 48:716 – 730, 1929.

[109] T. Parks and J. McClellan. Chebyshev approximation for nonrecursive digital filters with linear phase. *IEEE Transaction on Circuit Theory*, 19(2):189 – 194, 1972.

[110] F. Pfeiffer. *Mechanical System Dynamics*, volume 40. Lecture Notes in Applied and Computational Mechanics. Springer-Verlag Heidelberg, 2008.

[111] M. T. Pham, M. Gautier, and P. Poignet. Identification of joint stiffness with bandpass filtering. *IEEE International Conference on Robotics and Automation (ICRA)*, 3:2867 – 2872, 2001.

[112] M. Prokin. Double buffered wide-range frequency measurement method for digital tachometers. *IEEE transaction on Instrumentation and Measurement*, 40(3):606 – 610, 1991.

[113] M. Prokin. Extremely wide-range speed measurement using a double-buffered method. *IEEE Transactions on Industrial Electronics*, 41(5):550 – 559, 1994.

[114] T. Rossmann. Eine laufmaschine für rohre. fortschrittberichte. *VDI-Verlag, VDI, Reihe 8 Nr. 732 (German)*, 1998.

[115] P. Rouchon, J. Fliess, M.and Lèvine, and P. Martin. Flatness, motion planning and trailer systems. *In Proceedings of the 32nd Conference on Decision and Control*, 3: 2700 – 2705, 1993.

[116] S. Sastry and M. Bodson. *Adaptive control: stability, convergence, and robustness*. Prentice-Hall, first edition, 1989.

[117] M. Schwienbacher. Entwicklung eines kraft-momentensensors für einen humanoiden roboter. Master's thesis, Technischen Universität München, Lehrstuhl für Angewandte Mechanik Fakultät für Maschinenwesen (German), 2007.

[118] M. Schwienbacher. *Efficient Algorithms for Biped Robots*. PhD thesis, Technischen Universität München, Lehrstuhl für Angewandte Mechanik Fakultät für Maschinenwesen, 2014.

[119] M. Schwienbacher, T. Buschmann, S. Lohmeier, V. Favot, and H. Ulbrich. Self-collision avoidance and angular momentum compensation for a biped humanoid robot. *IEEE International Conference in Robotics and Automation (ICRA)*, pages 581 – 586, 2011.

[120] I. Selesnick. Maximally flat low-pass digital differentiators. *IEEE Transactions on Circuits and Systems*, 49(3):219 – 223, 2002.

[121] Sercos. *Specification SERCOS III interface*. SERCOS International e.V., 2004.

[122] Sercos. *Generic Device Profile*. SERCOS International e.V., 2009.

[123] Sercos. *Description of identification numbers*. SERCOS International e.V., 2009.

[124] B. Siciliano and E. O. Khatib. *Springer Handbook of Robotics*. Springer-Verlag Berlin Heidelberg, 2008.

[125] B. Siciliano, L. Sciavicco, L. Villani, and G. Oriolo. *Robotics. Modelling, Planning and Control*. Springer, second edition, 2010.

[126] D. Simon. *Optimal State Estimation Kalman, H_∞ and Nonlinear Approaches*. Wiley, first edition, 2006.

[127] J. Slotine and W. Li. Adaptive manipulator control: A case study. *IEEE Transactions on Automatic Control*, 11:995 – 1003, 1988.

[128] J. Slotine and L. Weiping. *Applied nonlinear control*. Prentice-Hall International Editions, College Station, Texas, 1st edition, 1990.

[129] M. W. Spong. Modeling and control of elastic joint robots. *Journal of Dynamic Systems, Measurement, and Control*, pages 310 – 319, 1987.

[130] M. W. Spong, K. Khorasani, and P. V. Kokotovic. An integral manifold approach to the feedback control of flexible joint robots. *IEEE Journal of Robotics and Automation*, 3 (4):291 – 300, 1987.

[131] J. Steuer and F. Pfeiffer. Regelstruktur einer laufmaschine für autonomes laufen in unebenem gelände. In *Autonome Mobile Systeme 1997*, Informatik aktuell, pages 13–23. (German) Springer Berlin Heidelberg, 1997.

[132] C. Swevers, J.and Ganseman, D. Tukel, J. De Schutter, and H. Van Brussel. Optimal robot excitation and identification. *IEEE Transactions on Robotics and Automation*, 13:730 – 740, 1997.

[133] H. Taghirad, P. Belanger, and A. Helmy. An experimental study on harmonic drives. technical report. *McGill University*, 1996.

[134] R. Tajima, D. Honda, and K. Suga. Fast running experiments involving a humanoid robot. *IEEE International Conference on Robotics and Automation (ICRA)*, pages 1571 – 1576, 2009.

[135] T. Takenaka, T. Matsumoto, T. Yoshiike, , and S. Shirokura. Real time motion generation and control for biped robot -2nd report: Running gait pattern generation. *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 1092 – 1099, 2009.

[136] T. Takenaka, T. Matsumoto, and T. Yoshiike. Real time motion generation and control for biped robot -1st report: Walking gait pattern generation. *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 1084 – 1091, 2009.

[137] T. Takenaka, T. Matsumoto, and T. Yoshiike. Real time motion generation and control for biped robot -3rd report: Dynamics error compensation. *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 1594 – 1600, 2009.

[138] T. Takenaka, T. Matsumoto, T. Yoshiike, T. Hasegawa, H. Kaneko, and A. Orita. Real time motion generation and control for biped robot -4th report: Integrated balance control. *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 1601 – 1608, 2009.

[139] T. Thuemmel and M. Rossner. Introduction to modelling and parameter identification methodology of linkages by measurements and simulation. *World Congress in Mechanism and Machine Science*, pages 1889 – 1902, 2011.

[140] T. Thuemmel, J. Rutzmoses, M. Rossner, and H. Ulbrich. Friction modelling and parameter value estimation of mechanism. *Joint International Conference on Multibody System Dynamics*, 2012.

[141] T. D. Tuttle. Understanding and modeling the behavior of a harmonic drive gear transmission. technical report. *MIT AI Lab.*, 1993.

[142] H. Ulbrich. *Maschinendynamik*. (German) Wiesbaden: B.G. Teubner Verlag, 1996.

[143] J. Urata, Y. Nakanishi, K. Okada, and M. Inaba. Design of high torque and high speed leg module for high power humanoid. *IEEE International Conference on Intelligent Robots and Systems (IROS)*, pages 4497 – 4502, 2010.

[144] K. Čapek's. *R.U.R. (Rossum's Universal Robots)*. Springer-Verlag Berlin Heidelberg, 1920.

[145] J. Villagra and C. Balaguer. Robust motion control for humanoid robot flexible joints. *Mediterranean Conference on Control & Automation Congress*, 2010.

[146] F. von Hundelshausen, M. Himmelsbach, F. Hecker, A. Mueller, and H.-J. Wuensche. Driving with tentacles: Integral structures for sensing and motion. *J. Field Robotics*, 25(9):640 – 73, 2008.

[147] E. E. Wallingford and J. D. Wilson. High-resolution shaft speed measurements using a microcomputer. *IEEE Transaction on Instrumentation and Measurment*, 26(2):113 – 116, 1977.

[148] *DS485 - Digital Clock Manager (DCM) Module*. Xilinx Inc., 2009.

[149] *DS265 - CAN v. 3.2*. Xilinx Inc., v. 3.2 edition, 2010.

[150] *DS530 - LogiCORE IP DividerGenerator v3.0*. Xilinx Inc., v. 3.0 edition, 2011.

[151] S. Yoshiaki, R. Watanabe, C. Aoyama, S. Matsunaga, N. Higaki, and K. Fujimura. The intelligent asimo: System overview and integration. *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 3:2478 – 2483, 2002.

[152] V. Zeman, R. Patel, and K. Khorasani. A neural network based control strategy for flexible joint manipulators. *IEEE International Conference on Decision and Control*, 2: 1759 – 1764, 1989.