

École polytechnique de Louvain

Towards abstraction-based control of a bipedal robot

Modelling of a 5-link planar bipedal robot

Author: **Simon DELIGNE**
Supervisor: **Raphaël JUNGERS**
Readers: **Paul FISETTE, François WIELANT**
Academic year 2023–2024
Master [120] in Electro-mechanical Engineering

Acknowledgements

First of all, I would like to thank Prof. Raphaël Jungers for the supervision of my master thesis and for welcoming me into his research team. I am grateful for the opportunity to participate in all the group meetings. Even though I did not always grasp all the complex mathematical details discussed, I always left with new concepts in mind, which was highly enriching.

My warmest thanks go to François Wielant. Throughout the year, he was always available to guide and validate my approaches. His motivation and sound advice were invaluable, especially when things became complicated, helping me to maintain a scientific approach and ultimately achieve the desired results.

I would also like to thank Julien Calbert for offering to help me integrate my work into Dionysos, and Adrien Banse for his support in debugging Julia-related problems and resolving library conflicts with Dionysos.

More generally, I thank the entire research team for their warm welcome and for their attention during my project update presentations at group meetings.

I take this opportunity to mention the use of ChatGPT in the writing of this thesis, in particular for text restructuring, correction, and translation.

Abstract

The increasing complexity of modern Cyber-Physical Systems (CPS) necessitates advanced control techniques. Abstraction-based control, a promising approach, involves discretizing the continuous state space of a system into a finite abstraction, synthesizing a controller on this abstraction, and then applying it back to the original system. The Dionysos.jl framework emerges as a tool to address the scalability challenges of abstraction-based control by implementing "smart" abstractions that leverage non-uniform cell shapes and local feedback controllers.

To demonstrate and validate the effectiveness of these techniques, a bipedal robot serves as an ideal testbed due to the inherent complexity of achieving stable locomotion. This thesis focuses on identifying a model of the bipedal robot and implementing it in Julia for integration with Dionysos. The model, based on the Euler-Lagrange formalism, incorporates physical parameters and actuator characteristics.

Contents

Introduction	viii
1 State of the art	1
1.1 Fundamentals and implementations of planar bipedal robots	2
1.2 5-link planar bipedal robot modeling	3
1.2.1 Full Lagrangian model	3
1.2.2 Simplified models	7
1.3 Actuator modeling	9
1.3.1 Direct Current (DC) motor model	9
1.3.2 Friction models	10
1.3.3 Parameters identification	11
1.4 Abstraction-based control strategy for bipedal robots	11
1.5 Conclusion	13
2 Robot Model and Friction Compensation	15
2.1 Presentation of the real working environment	15
2.1.1 The real robot	15
2.1.2 Software	16
2.2 Choice of model: 5-Link planar bipedal robot	16
2.3 Actuators description	18
2.3.1 Torque control	18
2.3.2 Friction compensation	20
2.4 Conclusion	20
3 Parameter identification	21
3.1 Basic experimental parameter identification	21
3.1.1 Experimental Setup and Data Collection	23
3.1.2 Armature resistance estimation	23
3.1.3 Back EMF constant estimation	23
3.1.4 Inertia, Coulomb's torque, and viscous component estimation . .	25
3.2 Shift from current control to PWM control	28
3.3 Optimization-based parameter calibration	29
3.3.1 Experimental setup and data collection	29
3.3.2 Frequency analysis	29
3.3.3 Reference signal construction	30
3.3.4 Parameters calibration	32
3.4 Conclusion	33

4	Model Validation	35
4.1	Experimental setup	35
4.2	Model performance evaluation	36
4.2.1	Comparison of predictor and simulator	36
4.2.2	Evaluation metrics	37
4.2.3	Model performance	38
4.2.4	Error bound	39
4.3	Improvement with Stribeck effect	40
4.3.1	Stribeck model performance	41
4.4	Conclusion	42
5	Implementation in Julia	43
5.1	Component-based model	43
5.2	Enhanced <i>URDF</i> File Construction	44
5.3	Simulation of Coulomb friction	46
6	Abstraction-based controller synthesis	49
6.1	Problem Formulation	49
6.1.1	Single-pendulum problem	50
6.1.2	Double-pendulum problem	50
7	Conclusion	53
7.1	Summary	53
7.2	Future Work	54
7.3	Final Thoughts	54
A	LabVIEW to Julia connection	59
B	GitHub Repository	61
B.1	Repository Structure	61

Introduction

The L2C (Learning to Control) project ¹ is a pioneering initiative aimed at addressing the growing complexity of modern Cyber-Physical Systems (CPS) through the development of advanced control techniques. The core strategy of the project is to construct correct-by-design controllers by discretizing the continuous state space of a system, synthesizing a controller from this abstraction, then applying the controller back to the real system. This process is known as abstraction-based or symbolic control.

Within this context, the Dionysos.jl framework emerges as a concrete tool to reach the goal of L2C. A significant issue in abstraction-based control is the "curse of dimensionality", where the computational effort grows exponentially with the system's size, affecting scalability. Dionysos.jl addresses this issue by implementing new "smart" abstractions, precisely to reduce this scalability issue. These smart abstractions leverage various techniques, such as using non-uniform cell shapes (e.g., ellipsoids) and employing local feedback controllers within each cell to improve the accuracy and efficiency of the abstraction process.

To demonstrate and validate the effectiveness of these advanced control techniques, a bipedal robot has been designed and constructed. Bipedal robots provide an ideal testbed, due to their inherent complexity of achieving stable walking or running gaits under various conditions. Successful control of bipedal robots would not only demonstrate the capabilities of Dionysos.jl but also contribute to the advancement of robotic field with a novel control approach.

The objective of this thesis is to identify a model of the bipedal robot and implement this model towards further integration with Dionysos in Julia. The model will be based on the Euler-Lagrange formalism, a widely used method in robotics for deriving equations of motion. The model will incorporate the physical parameters of the robot, including masses, lengths, and inertias of the links, as well as the characteristics of the actuators, such as friction.

The thesis starts with the State of the Art in Chapter 1, which provides an overview of the existing literature on bipedal robots, their modelling, and control strategies. It covers the fundamentals of planar bipedal robots, the derivation of their dynamic models using the Euler-Lagrange formalism, and the various simplified models used in the literature. Additionally, it discusses actuator modelling, focusing on DC motors and friction compensation techniques, and concludes with abstraction-based control strategies already applied on bipedal robots.

¹<https://perso.uclouvain.be/raphael.jungers/content/erc-consolidator-grant>

Next, Chapter 2 details the modelling of the specific 5-link planar bipedal robot used in this thesis. It describes the robot’s hardware and software components, the derivation of its dynamic model, and the necessity of friction compensation.

After that, Chapter 3 focuses on identifying the parameters of the robot’s dynamic model, particularly those related to friction in the actuators. It presents two methodologies: a basic experimental approach and an optimization-based approach.

Once the model is identified, Chapter 4 validates the model through a series of experiments with various input signals, and also explores potential model improvements.

Chapter 5 details the implementation of the validated model in Julia. It discusses the advantages of a component-based modelling approach and addresses the challenges of simulating Coulomb friction.

Finally, Chapter 6 outlines the formulation of two control problems as a first step towards integration with the Dionysos framework.

Chapter 1

State of the art

Bipedal robots, a subset of legged robots, are engineered with the objective to replicate some human characteristics. The motivations of engineers and researchers to study this field are diverse and numerous, ranging from prosthesis improvements to operations in dangerous environments. Nature has done things well, by optimizing human during several million years to develop an efficient form of locomotion—characterized by energy efficiency, stability across various terrains, and the ability to maintain balance even when disturbed. Since human environments are thought with bipedal movements in mind, let's think at the stairs for example, ensuring that robots can seamlessly integrate in our environments requires them similar locomotion abilities. Moreover, understanding and being able to replicate human movements allows the use of well-designed actuated prostheses for human rehabilitations. On technical side, bipedal robots are cyber-physical systems, which integrate both software (discrete) and real-world physics (continuous) processes, presenting challenges at control level. Combining low-level constraints, such as actuators maximum torque, to high-level objectives, like navigating hazardous terrain, is an open challenge. The complexity of accurately formulating the control problem for such hybrid (both continuous and discrete) systems raises new questions and leads to new technical innovations [12].

Non-classical control approaches are necessary for such complex systems. The correct-by-design synthesis of controllers, ensuring formal guarantees, can be achieved through abstraction-based technique. This technique relies on transforming the complex state-space of the hybrid system, the bipedal robot, into a finite state-space, in a sense reducing the complexity of the original system, while ensuring control performances. However, this technique is subject to the "curse of dimensionality", the abstraction becomes exponentially more complex with the number of degrees of freedom of the original system. To address this, the Dionysos.jl framework implements novel smart abstraction strategies [42]. Applying these model-based techniques on bipedal robots necessitates one thing : a model of the robot. Therefore, the subsequent sections will present the modeling of the bipedal robot to ensure its seamless integration with Dionysos.jl, setting the stage for advanced control applications.

The chapter begins with an introduction to the fundamental principles and diverse implementations of planar bipedal robots. This is followed by an in-depth exploration of the modeling of 5-link planar bipedal robots, including the derivation of their full Lagrangian models and the presentation of various simplified models. Then, the actuator modeling is detailed by presenting various friction models and methods for parameter identification. It is followed by a discussion on abstraction-based control strategies of

bipedal robots. The last section concludes the chapter by summing up the discussed notions.

1.1 Fundamentals and implementations of planar bipedal robots

Actuated bipedal robots can be categorized into two main groups: planar (2D) and three-dimensional (3D) robots. The 2D bipedal robots operate within the sagittal plane (the plane that splits human body into left and right parts), being able to move only forward and backward. On the other hand, 3D bipedal robots function in both the sagittal and the frontal planes, the last being the plane that splits human body into front and back parts. 3D robots are inherently more complex, due to their additional degrees of freedom compared to 2D robots.

For the purpose of the study, we will focus primarily on 2D robots, studying fundamental locomotion dynamics without the complexities of full spatial ones. Eight specific 2D models from the literature are presented, illustrating the diversity in implementations. Each planar bipedal robot is constrained to the sagittal plane by a boom, either radial or linear. Figure 1.1 visually illustrates each of the discussed robots, allowing for a better understanding of their structures and mechanical designs.

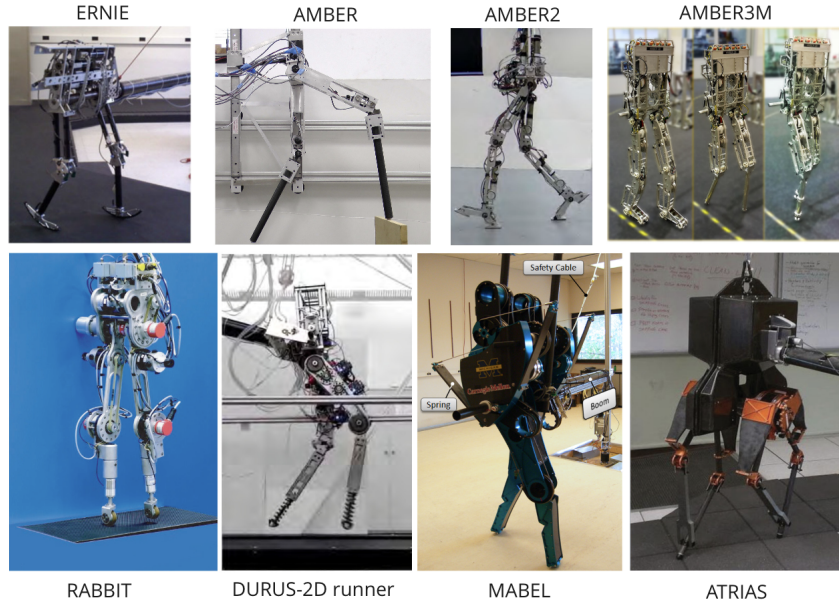


Figure 1.1: Eight notable 2D bipedal robots, each using a boom mechanism to constrain movement to the sagittal plane, including : ERNIE [27], AMBER [15], AMBER2 [18], AMBER3M [20], RABBIT [4], DURUS-2D Runner [24], MABEL [22], and ATRIAS [26]. This illustrates the diversity in design and mechanical structure within planar bipedal robotics.

Starting with **RABBIT**, **ERNIE** and **AMBER**, these robots feature a five-link structure (two tibias, two femurs, and a torso). Their knees and hips are powered with DC motors, and the feet are curved or pointed, and so, underactuated [4, 27, 15]. **AMBER2** advances the design with actuated feet, adding two links and two DC

motors on it. This allows multi-domain walking gaits, across phases of full, under, and over-actuation, approaching human locomotion [18]. **DURUS-2D** distinguishes itself by its legs with springs to store energy during impacts, giving efficient high-speed running [24]. **AMBER3M** represents a modular upgrade of AMBER and AMBER2, offering interchangeable feet designs—flat, point, or spring—to adapt to varied experimental needs [20]. **MABEL** is designed with built-in springs that store kinetic energy during running gaits, improving overall efficiency and speed [22]. Lastly, **ATRIAS**, a 3D bipedal robot, is mentioned to illustrate the possibility of constraining into the sagittal plane a 3D robot using a boom. This adaptation allows it to effectively be used in planar studies, demonstrating a method of simplifying complex robots for specific research needs [26].

Each of these designs highlights different aspects of the existing mechanical and control strategies of bipedal locomotion. By constraining the motion to the sagittal plane, researchers can focus on the key dynamics of walking and running, thus providing valuable insights into the development of more advanced 3D bipedal robots in the future.

1.2 5-link planar bipedal robot modeling

For the remainder of this study, the focus will be on the specific case of 5-link planar bipedal robots, illustrated by the previously presented RABBIT, ERNIE and AMBER's series robots with pointed feet, and consequently underactuated. First, the full hybrid model of the robot is derived. Then, possible simplifications of the model are presented.

1.2.1 Full Lagrangian model

The following analysis is primarily derived from the work of C. Chevallereau et al. in [4] and [6]. In the general case, without constraints or contacts, the robot has 7 degree-of-freedom (DOF), i.e. :

- 5 DOF for the orientation of each links
- 2 DOF for the position of the body in the sagittal plane, along x and y axis

As the robot has 4 actuators, the system is under-actuated. 14 state variables are necessary to describe the system, 7 variables to describe the positions, and 7 variables to describe the velocities. Figure 1.2 presents the system and 3 possible phases: the flight phase, the stance phase or single support phase, and the double support phase. In the single support phase the center of mass of the robot is fully determined by the orientation of the links, reducing the system to 5 DOF, hence 10 state variables. The actual continuous dynamic model of each phase can be obtained by deriving the Euler-Lagrange equations:

$$\frac{d}{dt} \left(\frac{\partial \mathcal{L}}{\partial \dot{q}_i} \right) - \frac{\partial \mathcal{L}}{\partial q_i} = \Gamma_i \quad (1.1)$$

where $\mathcal{L}$ is the Lagrangian, defined as the difference between the kinetic energy and the potential energy, q_i is the vector of state variables, and Γ_i is the vector of generalized forces (external forces and torques) [4].

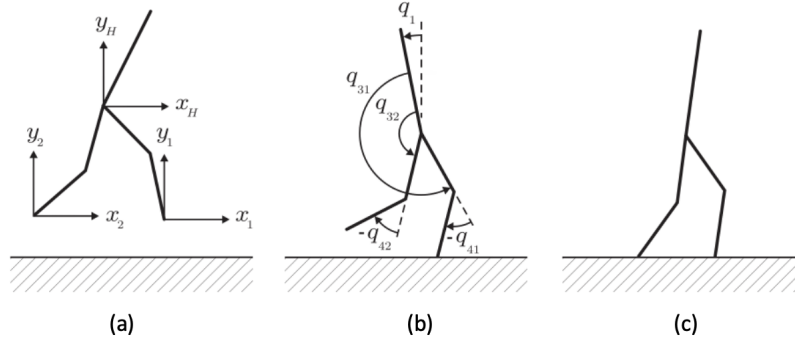


Figure 1.2: The robot can be in 3 possible phases: (a) in flight phase, (b) in single support or stance phase, and (c) in double support phase [4]

The derivation of the Euler-Lagrange equation for each phase is presented now.

Flight phase

During the flight phase, the robot has no contact with the ground, and therefore, no ground reaction forces are present. Applying the Lagrange equation yields [6]:

$$M_f(q)\ddot{q} + C_f(q, \dot{q})\dot{q} + G_f(q) = B_f u \quad (1.2)$$

where :

- $q \in \mathbb{R}^7$ is the vector of generalized coordinates, expressed as $(X \ Y \ q_1 \ q_{31} \ q_{32} \ q_{41} \ q_{42})$, where (X, Y) represents, for example, the position of the center of mass in the sagittal plane.
- $M_f(q) \in \mathbb{R}^{7 \times 7}$ is the mass or inertia matrix.
- $C_f(q, \dot{q}) \in \mathbb{R}^{7 \times 7}$ is the Coriolis and centripetal force matrix.
- $G_f(q) \in \mathbb{R}^7$ is the gravity vector.
- The vector of generalized forces Γ_i simplifies to $B_f u$, where $B_f \in \mathbb{R}^{7 \times 4}$ defines the input matrix that maps the joint torques vector $u \in \mathbb{R}^4$ into the model.

The dynamics described by the equation 1.2 can be represented in a more abstract form. For control design and simulation, it is often useful to introduce a state-space representation of the system, expressed as [6]:

$$\dot{x} = f_f(x) + g_f(x)u$$

Here, x is the state vector with both the generalized coordinates q and their velocities $\dot{q}$, $f_f(x)$ represents the system's dynamics, including the effects of inertia, gravity, Coriolis and centripetal forces, while $g_f(x)$ characterizes how the control inputs u influence the state.

Single support phase

During the single support phase, one leg (the stance leg) maintains contact with the ground, acting as an ideal pivot, while the other leg (the swing leg) is free. The ground contact introduces a holonomic constraint, reducing the system's degrees of freedom from 7 to 5. Consequently, the generalized coordinate vector (q) is redefined as $(q_1 \ q_{31} \ q_{32} \ q_{41} \ q_{42})$ [6].

Applying the Euler-Lagrange equation (Equation 1.1) to this constrained system yields the following dynamic equation [10]:

$$M_s(q)\ddot{q} + C_s(q, \dot{q})\dot{q} + G_s(q) = B_s u + E_s(q)F_{\text{ext}} \quad (1.3)$$

where:

- $M_s(q) \in \mathbb{R}^{5 \times 5}$, $C_s(q, \dot{q}) \in \mathbb{R}^{5 \times 5}$, $G_s(q) \in \mathbb{R}^5$, and $B_s \in \mathbb{R}^{5 \times 4}$ differ respectively from M_f , C_f , G_f , and B_f due to the reduced degrees of freedom.
- $E_s \in \mathbb{R}^{5 \times 5}$ is derived from the principle of virtual work and maps the vector of ground reaction forces acting on the stance foot F^{ext} into the model.

Moreover, if we assume that the Cartesian coordinates $(X \ Y)$ represent the position of the stance leg end $(x_1 \ y_1)$, and that we assume no slipping at the contact point (i.e., the pointed foot is stationary), we can then solve the ground reaction forces F^{ext} as explicit functions of the generalized coordinates of the links (q), their velocities ($\dot{q}$), and the joint torques (u) [10]. The dynamic equation 1.3 simplifies to:

$$M_s(q)\ddot{q} + C_s(q, \dot{q})\dot{q} + G_s(q) = B_s u \quad (1.4)$$

Similar to the flight phase, we can represent the single support dynamics in a state-space form:

$$\dot{x} = f_s(x) + g_s(x)u$$

Instantaneous double-support phase, or impact phase

The impact phase is the brief moment when the swing leg contacts the ground, the double-support phase being considered as instantaneous. This contact is often modeled as an instantaneous, inelastic collision [4, 6, 10]. The robot's configuration remains the same during this phase, but the velocities undergo abrupt changes due to the impact. To simplify the analysis and simulation, the following assumptions are made [4]:

1. Instantaneous impact.
2. No rebound or slip of the swing leg upon contact with the ground.
3. Stance leg lift off the ground without any interaction or influence on the impact dynamics.
4. Impulse forces models the ground reaction forces during impact.
5. No configuration change of the robot during the impact.
6. Negligible actuator impulses: the torques generated by the robot's actuators are assumed to be negligible during the impact.

Under these assumptions, the impact model accounts for the relabeling of the robot's coordinates, which occurs after each instantaneous phase of double support. The development of the impact model requires the full 7 DOF of the robot. By taking the (X, Y) Cartesian coordinates as the (x_H, y_H) coordinates of the hip, the following model is once again obtained through the method of Lagrange:

$$M_i(q)\ddot{q} + C_i(q, \dot{q})\dot{q} + G_i(q) = B_i u + E_i(q)F^{\text{ext}} \quad (1.5)$$

It is shown in [5], that it is possible to solve the equation for the velocities of the robot's links just before and just after the impact, respectively $\dot{q}^-$ and $\dot{q}^+$. These last should then be used to re-initialize the model for the next phase just after the impact. A relabeling is then necessary, as the swing leg becomes the stance leg just after impact. The final expression of the impact dynamic is :

$$x^+ = \Delta(x^-) \quad (1.6)$$

where $x^+ := (q^+, \dot{q}^+)$ is the state value just after impact expressed in terms of $x^- := (q^-, \dot{q}^-)$, the state value just before impact.

Hybrid Model

Finally, the complete dynamics of the 5-link planar bipedal robot can be represented as a hybrid dynamical system. A hybrid system is one that integrates both continuous and discrete dynamics within a finite automaton. In the context of bipedal robots, this system combines the continuous dynamics of the flight and single support phases with the discrete dynamics of the impact phase. Mathematically, the hybrid model can be expressed as:

$$\begin{cases} \dot{x} = \begin{cases} f_f(x) + g_f(x)u, & \text{if } x \in \mathcal{X}_f \\ f_s(x) + g_s(x)u, & \text{if } x \in \mathcal{X}_s \end{cases} \\ x^+ = \Delta(x^-), & \text{if } x^- \in \mathcal{S} \end{cases} \quad (1.7)$$

where:

- x is the state vector, with both generalized coordinates q and their velocities $\dot{q}$.
- $f_f(x)$ and $g_f(x)$ define the continuous dynamics during the flight phase.
- $f_s(x)$ and $g_s(x)$ define the continuous dynamics during the single support phase.
- $\Delta(x^-)$ describes the discrete dynamics during the impact phase, mapping the state just before impact (x^-) to the state just after impact (x^+).
- $\mathcal{X}_f \subset \mathcal{R}^{14}$ is the set of all states belonging to the flight phase.
- $\mathcal{X}_s \subset \mathcal{R}^{10}$ is the set of all states belonging to the single support phase.
- $\mathcal{S}$ is the switching surface that triggers the impact event.

This hybrid model allows for precise analysis and simulation of 5-link planar bipedal robots. However, its high order and hybrid dynamics pose challenges in terms of computational efficiency for both simulation and control design. To mitigate these challenges, the subsequent section will explore simplified models that offer reduced complexity while still capturing the essential dynamics of bipedal locomotion.

1.2.2 Simplified models

As the full dynamic model can be challenging to control due to its complexity, simplified models and methodologies are often used in the literature. These models, while less complex, are still highly effective for many applications, offering a practical balance between computational efficiency and necessary dynamical fidelity. The most common ones are briefly introduced.

Linear Inverted Pendulum Model (LIPM)

The Linear Inverted Pendulum Model (LIPM) is a simplified model used to represent the dynamics of a bipedal robot during the single support phase. It differs from a simple inverted pendulum model (SIPM) in the following sense : the robot's center of mass (CoM) is constrained to move at a constant height, as shown in Figure 1.3. This assumption allows the dynamics of the robot to be linearized, making it easier to analyze and control [29].

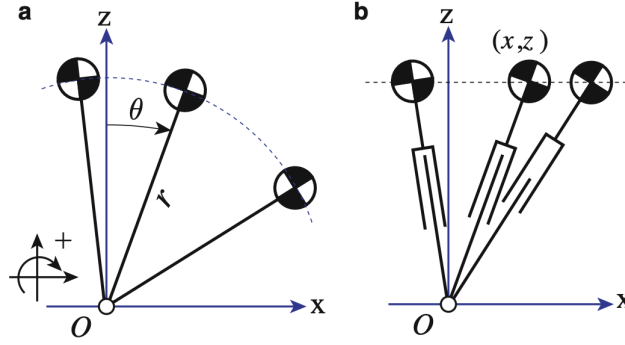


Figure 1.3: (a) The simple inverted pendulum compared with (b) the linear inverted pendulum model, where the CoM remains at constant height [29].

The dynamic equation of the LIPM is given by:

$$\ddot{x} = \frac{g}{z_{com}} x \quad (1.8)$$

where x and z are respectively the horizontal and vertical positions of the CoM, and g is the gravity acceleration.

In [29], the author presents a method to derive the linear dynamics of a 2D biped robot that walks on flat ground. This method can be effectively extended to walk on uneven ground in three dimensions.

In [34], the authors use the LIPM to determine the step duration and desired momentum at the beginning of each step of the bipedal robot. They then use online optimization to generate the desired vertical CoM trajectory.

Spring Loaded Inverted Pendulum Model (SLIPM)

The Spring Loaded Inverted Pendulum Model (SLIPM) represents the hip of the bipedal robot as a point mass, connected to two massless springy legs. By adding the compliance of the legs, running gaits can be more energy efficient as it becomes closer to the motion observed in human gaits.

The dynamic equation of the SLIPM is given by:

$$\ddot{x}_G = \frac{1}{m}(f_L(x_G) + f_R(x_G)) + g_0 \quad (1.9)$$

where $\ddot{x}_G$ is the acceleration of the center of mass (CoM), m is the total mass of the robot, $f_L(x_G)$ and $f_R(x_G)$ are the spring forces of the left and right legs, and g_0 is the acceleration due to gravity [14].

SLIPM was successful at generating stable periodic gaits for planar robot, as presented in [14]. The authors design a controller that tracks the desired CoM trajectory generated by the SLIPM, taking advantage of the model's open-loop stability. The controller also includes additional actions to increase robustness and reject external disturbances.

Hybrid Zero Dynamics (HZD)

The Hybrid Zero Dynamics (HZD) method is a framework for designing feedback control laws that induce stable walking and running motions. The main idea is to reduce the dynamics of the robot by defining a set of output functions, called virtual constraints, and to control them to 0. These virtual constraints are designed to create a lower-dimensional zero dynamics domain in the original robot's state space, where the system's dynamics are simplified and easier to control [21].

The HZD method involves the following steps:

1. Virtual constraint design: Define a set of output functions that take into account the desired motion of the robot y_d , such as the position of the center of mass, joint angles, or foot trajectories. Then we define the following equation, and the objective will be to control it to zero :

$$y(q) := y_a(q) - y_d(\tau(q), \alpha) \quad \in \mathbb{R}^m \quad (1.10)$$

where y_a is the actual output, $\tau(q)$ is a phase variable that parameterizes time, and α is a set of parameters that can be adjusted to tune the desired output.

2. Feedback linearization: Design a feedback control law that drives the actual outputs to track the desired outputs, effectively restricting the robot's motion to the zero dynamics domain. Such law can be defined as :

$$Z := \{x \in D \mid y(x) = 0 \text{ and } \dot{y}(x) = 0\} \quad (1.11)$$

where x is the state of the robot and D is the domain of the system.

3. Hybrid Invariance: To ensure stability of the overall dynamics, it is necessary to ensure hybrid invariance of the zero dynamics, mathematically expressed as:

$$x^- \in S \cap Z \quad \Rightarrow \quad x^+ = \Delta(x^-) \in Z \quad (1.12)$$

By enforcing hybrid zero dynamics, the HZD method guarantees that the robot's motion will converge to and remain on the desired zero dynamics domain, resulting in stable and robust locomotion [21]. This approach has been successfully applied to various bipedal robots, including AMBER [15], MABEL [22], and DURUS [24], to achieve stable walking and running gaits.

1.3 Actuator modeling

Electric motors are widely used in bipedal robot thanks to their ease of low-level joint control, increased bandwidth capabilities, and simpler assembly relative to hydraulic or pneumatic alternatives [11]. Servomotors, in particular, offer lot of control capabilities, including position, velocity, and torque control, making them suitable for bipedal locomotion. Servomotors can also have integrated sensors, like encoders, allowing compact mechanical designs.

However, servomotors often act as "black-box" systems with proprietary firmware and incomplete datasheets, limiting their potential. While internal controllers (often Proportional-Integral-Derivative (PID) controllers) effectively manage position control in static conditions, they fail for complex trajectory tracking, leading to delays and distortions. This is a problem for humanoid robots that require precise, dynamic, and torque-intensive movements [23].

Friction compensation allows to build more robust controllers, as the latter needs then to manage smaller perturbations. This not only reduces energy consumption but also results in more compliant robot movements when dealing with disturbances [17].

The rest of the section will first present the model of a direct current motor, followed by common friction models and lastly several ways to identify the parameters of the models.

1.3.1 Direct Current (DC) motor model

The motor model that will be further referred is the one of the permanent magnet direct current (PMDC) motor. Figure 1.4 shows the motor schematic, with the electrical and mechanical parts.

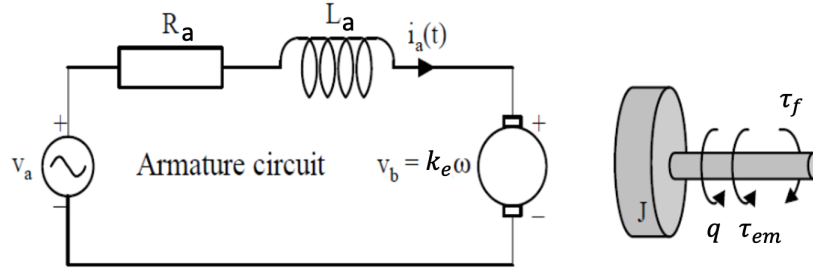


Figure 1.4: DC motor schematic with on the left the electrical part, and on the right the mechanical one [16].

The electrical equation of PMDC motors is classically :

$$U_a = R_a i_a + L_a \frac{di_a}{dt} + k_e \dot{q} \quad (1.13)$$

In this equation, U_a is the motor's armature voltage, R_a is the armature resistance, and k_e is the back electromotive force (EMF) constant expressed in Vs/rad. The mechanical equation is described by the following equation of motion:

$$\tau_{em} - \tau_f = J \frac{d\dot{q}}{dt} \quad (1.14)$$

where τ_{em} and τ_f denote the electromagnetic and friction torques, respectively, and J is the system's inertia.

The electromagnetic torque for a DC machine with permanent magnet excitation is defined as:

$$\tau_{em} = k_t i_a \quad (1.15)$$

where k_t is the torque constant.

1.3.2 Friction models

In the literature, there are many friction models that can be used to describe the friction torque τ_f of Equation 1.14. The most common friction models are discussed below.

The classical static models

The most common approach to model friction combines Coulomb friction, viscous friction, and the Stribeck effect. Coulomb friction, τ_c , represents a constant friction torque characteristic of dry contacts. The viscous friction component, K_v , is proportional to the relative velocity of the two surfaces in contact. It is observed that higher torque, τ_s , is needed to initiate movement than to maintain it, leading to well-known stick-slip behaviour. The Stribeck effect, identified by Stribeck, highlights that the transition from static to dynamic friction is not discontinuous but follows a curve that decreases with increasing velocity in lubricated conditions. The complete classical model can then be expressed as [9]:

$$\tau_f = \text{sign}(\dot{q}) \left(\tau_c + (\tau_s - \tau_c) e^{-(\dot{q}/\dot{q}_s)^\delta} \right) + K_v \dot{q} \quad (1.16)$$

where τ_c is the Coulomb friction coefficient, τ_s is the static friction coefficient, $\dot{q}_s$ is the Stribeck velocity, δ is an empirical parameter, and K_v is the viscous friction coefficient. Figure 1.5 illustrates each friction curves.

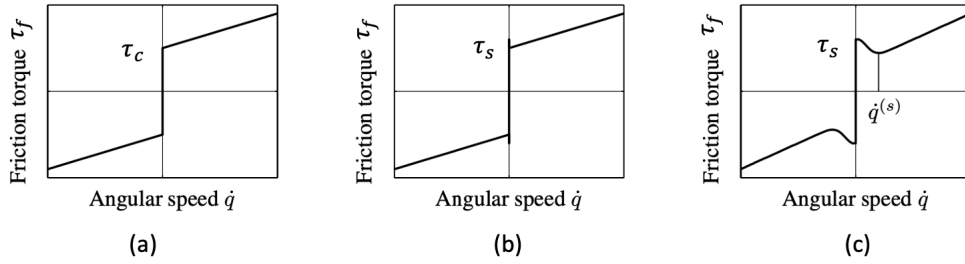


Figure 1.5: From left to right: (a) Coulomb friction and viscous friction; (b) Addition of static friction; (c) Stribeck curve [9].

Dynamic models

If more precise friction modeling is needed, the listed dynamic models below take most friction behaviours into account :

- **The Dahl model** describes in a simple way observed hysteresis and pre-sliding displacement. It is often expressed as :

$$\frac{d\tau_f}{dx} = \sigma \left(1 - \frac{\tau_f}{\tau_c} \text{sign}(\dot{q}) \right)$$

where σ is a stiffness coefficient [1, 2].

- **The LuGre model**, widely used, combines the Dahl with the Bristle model, where the contact surface is composed of many little bristles. These bristles represent elastic deformation, each generating friction. A set of non-linear equations is needed to describe the model, and combines the classical static friction parameters (τ_s , τ_c , K_v , and the Stribeck velocity $\dot{q}_s$) with 2 dynamic parameters that characterize the bristles state. Six parameters need to be identified, which is a challenging task [39].
- **The Generalized Maxwell-Slip (GMS) model** is considered as the most advanced among the discussed models. It is composed of multiple Maxwell-Slip elements (spring and damper element) in series or parallel configurations. Stiction, Stribeck effect, transient behaviours like pre-sliding, and hysteresis phenomena are all modelled with GMS [7]. Its accuracy is verified experimentally in [7, 8].

For the curious reader about other models of friction, [2] goes through a variety of static and dynamic models.

1.3.3 Parameters identification

Identification of parameters in DC motors is challenging due to the non-linear nature of friction and its direct impact on motor behaviour. Moreover, it is observed that friction varies over time [3]. Factors like temperature and lubricant distribution can impact measurements, making it difficult to maintain consistent conditions. This section proposes an overview of the way searchers have identified parameters of PMDC motors and its friction.

Experimental setups varies from isolated motor connected to variable inertia's [33] to setups that are close to the real operational conditions, such as actual soccer robot performing kicking [17, 23]. The input signal is designed to excite as much states as possible of the motor system. Square waves, sinusoids with varying frequencies (Fourier series), or step changes are commonly used signals [28, 32]. The selected friction model depends on the actual requirements of searchers, varying from simple viscous friction models to more complex ones like the previously introduced GMS model.

Once the experiment and the model is clearly defined, several methods are used to identify the parameters. Optimization algorithms like Iterative Learning Control (ILC) [17], Covariance Matrix Adaptation Evolution Strategy (CMA-ES) [23], or bio-inspired algorithms are used [32, 40]. Other searchers isolate parameters by doing different experiments, each isolating specific parameters [33].

The validation of the identified models is done with experimental data by assessing the goodness of fit. Metrics like the Normalized Root Mean Square Error (NRMSE) and Mean Square Error (MSE) are used for this purpose [32]. Another solution is validate the model by integrating it directly into the real-world application and evaluating its performance, compared to simpler models [17, 23].

1.4 Abstraction-based control strategy for bipedal robots

The main goal to be reached with this thesis is to build the model of the bipedal robot available in the lab in order to apply the smart abstraction-based control technique implemented in the framework Dionysos.jl presented in paper [42]. This section will briefly introduce the main idea of this technique, and compare it with similar researches.

The infinite state-space of the concrete system, namely the bipedal robot, is discretized into a finite state-space, called the abstraction of the original system. An abstract controller is then synthesized on this finite state-space, and finally concretized on the original system. A visual representation of the steps is presented in Figure 1.6.

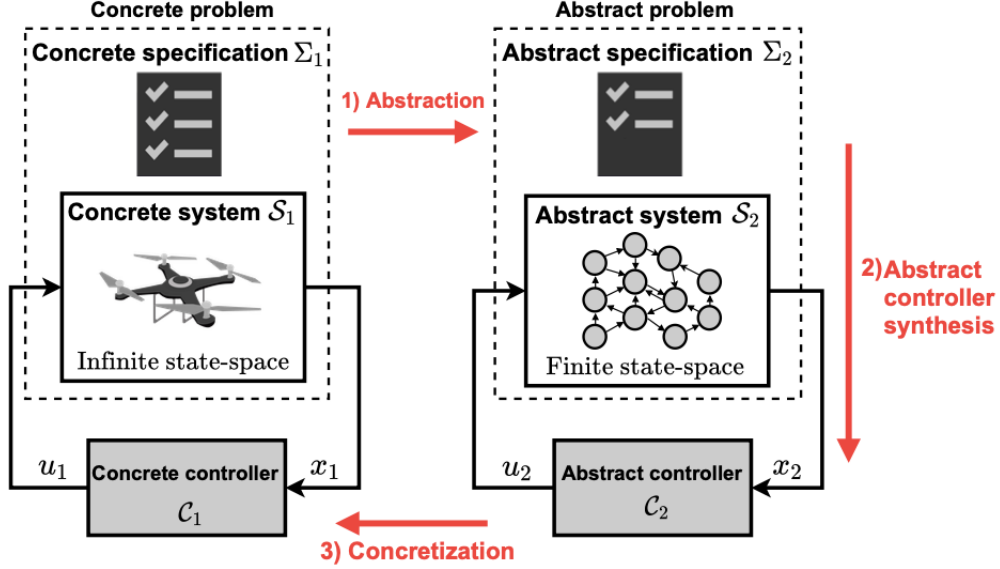


Figure 1.6: Abstraction principle, the original system is first abstracted into a finite state-space. An abstract controller is synthesised and concretized back on the original system [42].

This approach of using classical abstraction has already been done by Aaron D. Ames et al. in [22] in a certain way. The main problem with abstraction-based control is the "curse of dimensionality", i.e. exponential increase in the number of states within an abstraction as the dimensions of the original system grow. To address this challenge, the authors propose a two-step approach :

1. Dimensionality reduction : using feedback linearization (partial hybrid zero dynamics, a variation of previously introduced hybrid zero dynamics), they control a lower-dimensional surface of the robot's states.
2. Abstraction-based control on reduced dynamics : this simplified system is then abstracted into a finite-state model, and is further used to synthesize a controller that enforces the system to remains in safe states.

The authors acknowledge a gap between the theoretical controller and its practical implementation. The formally correct controller, designed in the abstract space, doesn't perfectly concretize to the real-world robot due to factors like un-modeled phenomena. On real experiments, they followed these steps :

- Trajectory Generation: Instead of directly implementing the abstract controller, they integrate the simplified model's equations to generate reference trajectories for the robot's joints.
- PD Control: They use PD controllers at each joint to track these reference trajectories.

A notable feature of abstraction-based control is the correct-by-design one, which gives formal guarantees to the controller. Other researchers have implemented discretization techniques of the state space, like the graph-based model predictive control [13], but no formal guarantees are present.

Dionysos.jl aims at dealing with the "curse of dimensionality" challenge by building smart abstractions of the state-space, using ellipsoids to discretize the state-space. Then affine feedback laws allows more precise and flexible control compared to traditional methods that use piecewise constant control signals.

1.5 Conclusion

In conclusion, there is a wide range of bipedal robots, categorized mainly into three-dimensional and planar robots. This chapter has primarily focused on planar robots, starting with several existing designs and their characteristics.

The discussion then narrowed to the specific case of the 5-link planar bipedal robot. The Lagrangian model was derived for several motion phases of the robot, the flight, the single support, and the impact phases. These models were integrated into a complex hybrid model. To manage the complexity of the dynamics, simplified models such as the LIPM, the SLIPM, and the HZD approach were introduced.

The chapter then covered DC motor modeling and the detailed analysis of friction models. The Coulomb friction, static friction, viscous component, and Stribeck effect were discussed in the classical static models section. Then more advanced friction models were introduced, the dynamic models, including the Dahl, LuGre, and GMS models.

Lastly, the chapter explored abstraction-based control strategy, highlighting how these have been applied by other research groups and identifying existing gaps. This sets the stage for the integration of the Dionysos.jl framework, which aims to overcome the curse of dimensionality through smart abstractions.

Chapter 2

Robot Model and Friction Compensation

This chapter explores the modeling and control framework for the actual bipedal robot available in the lab. It begins by presenting the real working environment, including a description of the physical robot and its software components. Next, the chosen model of the robot is introduced, a 5-link planar bipedal model, using Euler-Lagrange method to derive the dynamical equations. Finally, a description of the actuators is provided, focusing on the actual control methods used. A first experiment is presented, identifying first model's inaccuracies, and motivating the choice of friction compensation.

2.1 Presentation of the real working environment

2.1.1 The real robot

The bipedal robot available in the lab has been designed some years ago and has a similar design as the planar bipedal robots presented in Section 1.1. The robot is composed of 5 links: 2 tibias, 2 femurs and a very thin torso. Four servomotors are actuating the robot: 2 for the knees and 2 for the hips. The feet of the robot are modular, one can choose between pointed feet or flat feet. The second option was designed for the purpose of Zero Moment Point (ZMP) based walking gaits generator studied by a previous master student in [36]. For the remaining of this thesis, we will consider the pointed feet only.

The hip is connected to a radial boom, restricting robot movements on the sagittal plane. Two assumptions are made here:

1. The radius of the boom is assumed to be infinite, effectively limiting the robot movements on a planar surface.
2. The mass and inertia of the boom are neglected in the model.

These assumptions help at simplifying the dynamics, future work could include modelling the actual finite radius of the boom and/or its mass and inertia if higher precision is required. A picture of the robot and its boom is shown in Figure 2.1.

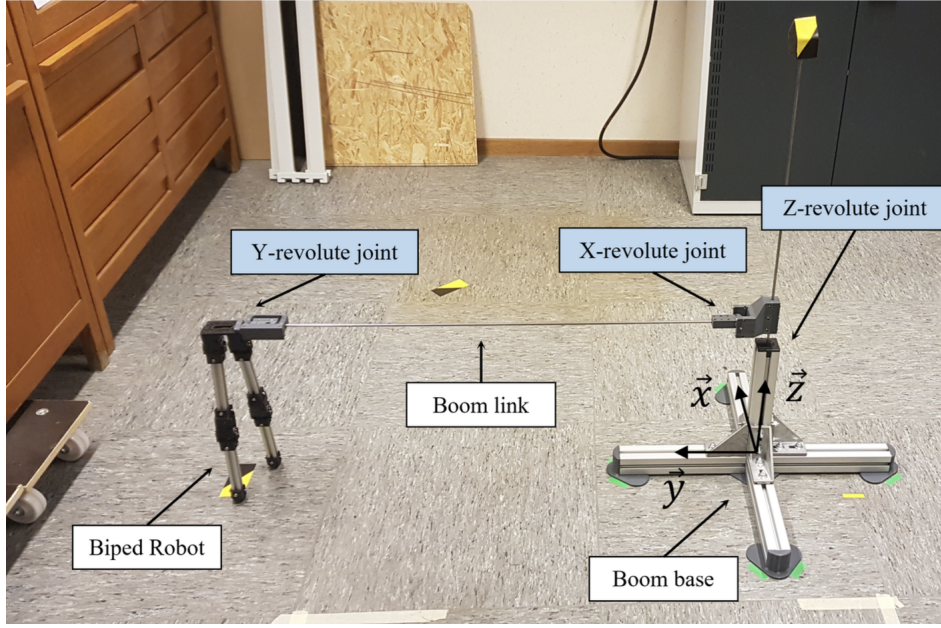


Figure 2.1: Setup of the bipedal robot in the lab, with the boom restricting it to the sagittal plane, from [36].

2.1.2 Software

The four servomotors are connected via a U2D2 adapter to a Windows computer, which uses LabVIEW software for real-time data acquisition and control. The achievable sampling rate is 20 ms. During each 20 ms cycle, both control and sensing of the four Dynamixels are performed.

A technical question is how the control inputs computed in Julia are applied to the real robot using LabVIEW. The controllers computed offline by `Dionysos.jl` have the format of a look-up table. This look-up table can then be directly uploaded into LabVIEW for real-time control of the robot. However, at some point the controller could become too complex to be directly uploaded into LabVIEW. Another approach must then be followed. Annex A proposes an interface where both Julia and LabVIEW are running concurrently. This setup allows Julia to perform complex computations while LabVIEW handles real-time execution, ensuring seamless integration and control.

2.2 Choice of model: 5-Link planar bipedal robot

The real robot can be model as a 5-link planar bipedal robot, as introduced in Section 1.2. The principal elements are recalled here.

- The robot has 7 degree of freedom (5 for the orientation of each link + 2 for the position of the body in the sagittal plane), hence 14 state variables to describe it (7 for the positions and 7 for associated velocities).
- The robot can be in 3 phases: the flight phase, the single-support phase, and the impact phase (an instantaneous double-support phase).

- In single support phase, the foot in contact with the ground acts as an ideal pivot point (no sliding). The system reduces to 5 DOF as the ground contact introduces a holonomic constraint.
- The impact can be assumed instantaneous, and, under some assumptions, the equation for the velocities of the robot's links just before and just after the impact can be solved. This can then be used to reinitialize the system for the next phase just after impact.
- The continuous and discrete dynamics can be represented as a single hybrid system.

The generalized coordinates are chosen as $q = (x, y, q_1, q_2, q_3, q_4, q_5)$, where q_{1-5} are the orientation of the torso, the right and left femurs, and the right and left tibias. The coordinates (x, y) describe the position of the body in the sagittal plane, one can choose the position of the hip (x_H, y_H) as reference for example. This configuration is illustrated in Figure 2.2.

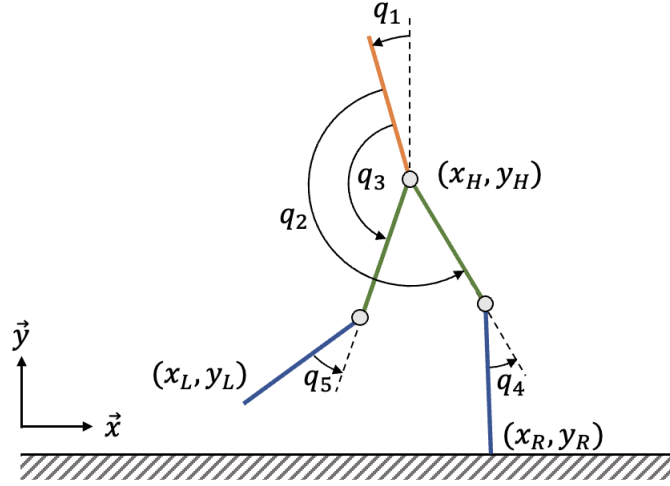


Figure 2.2: Schematic of the 5-link planar bipedal robot model, with the generalized coordinates q_1 to q_5 for the torso (in orange), femurs (in green), and tibias (in blue). The coordinate points (x_H, y_H) , (x_R, y_R) , and (x_L, y_L) represent the positions of the hip and right and left feet.

As discussed in Section 1.2, the equation of motion of the robot can be derived with the method of Euler-Lagrange. A more compact form is expressed here:

$$M(q)\ddot{q} + C(q, \dot{q}) = \tau \quad (2.1)$$

where $M(q)$ is the mass matrix, $C(q, \dot{q})$ is called the generalized bias force taking into account the Coriolis and centrifugal forces, gravity, and other forces acting on the system different from τ , the vector of generalized forces.

As the objective is to build a model in Julia for further integration in `Dionysos.jl`, the use of some Julia library can help. Notably, the matrices $M(q)$ and $C(q, \dot{q})$ can be determined using `RigidBodyDynamics.jl` given the *URDF* (Unified Robotics Description Format) geometry of the system [19]. The library allows parsing of the URDF model of the robot to produce a mechanism object. Then, rigid body dynamics algorithms are

used to determine the kinematics and dynamics of the mechanism. Moreover, as the objective of the model will be to simulate the robot behaviour for a given torque input in the actuators, we are mainly interested in the forward dynamics algorithm, which calculates the acceleration $\ddot{q}$ given τ .

2.3 Actuators description

The robot's actuators are Dynamixel units, developed by the South Korean company ROBOTIS. The choice of this kind of actuator was originally made for several reasons. The DC motor, the gearbox, the driver, the sensors, and the controller are all in a single compact unit. This simplifies the design of the robot compared to traditional motor and driver setups. Additionally, they can be chained together through a single connector for both power and data, which simplifies cable management. The actuators sense the current and voltage applied to the DC motor, as well as the position and velocity of the motor's shaft. They can be controlled in multiple modes: position, velocity, current, and Pulse Width Modulation (PWM).

Two different models of Dynamixel are selected for the knees and the hips; they are essentially the same, only the gearbox ratio differs:

- Dynamixel XH430-W350: gearbox ratio of 353:1 for the hips.
- Dynamixel XH430-W210: gearbox ratio of 212:1 for the knees, requiring more balance of torque and speed compared to the hips, making a lower gearbox ratio more suitable.

2.3.1 Torque control

The controller synthesized offline with `Dionysos.jl` is designed to be piecewise affine. This means the state space is discretized into cells covered by ellipsoids, within which affine controllers are used. An affine controller in each cell is defined by:

$$\kappa(x) = Kx + l$$

where $x = (q, \dot{q})$ is the state of the system, K is a matrix that determines the state feedback gain, and l is a vector that adds an offset. These parameters are optimized within each cell to ensure that the controller can effectively drive the system from any state within the cell towards a desired target cell or state.

To design this controller, we rely on the model of the robot's dynamics, given by the forward dynamics:

$$\ddot{q} = M(q)^{-1}(\tau - C(q, \dot{q})) \quad (2.2)$$

By integrating the acceleration over a certain time step, we can evaluate the position and velocity evolution of the system. The external torque, τ , therefore, acts as the system's control input.

There are two possibilities to control the Dynamixels in torque, either through the current or through the PWM control. As derived in Section 1.3, the model of the actuator is that of a DC motor. There is a direct proportionality between current and the electromagnetic torque produced by the motor:

$$\tau_{em} = k_t i_a \quad (2.3)$$

Control through current is then often preferred thanks to this direct proportionality. One parameter that needs to be identified is the torque constant k_t . For this, the datasheet of the Dynamixel provides results of stall experiments. These experiments yield the following measurements of torque, voltage, and current:

Torque (N.m)	Voltage (V)	Current (A)
3.1	11.1	1.2
3.4	12.0	1.3
4.2	14.8	1.5

Table 2.1: Torque, Voltage, and Current measurements in stall tests.

Given the torque and current measurements from the stall tests, the torque constant k_t can be directly deduced from Equation 2.3. By averaging the three measurements, the estimated torque constant numerical value is $k_t = 2.6657 [Nm/A]$.

We now know the relation between the input current and the torque that will be produced. We can now simulate the model, as we know all the parameters of it. We will focus only on the hip motor, with the knee fixed and without any ground interaction, thus simplifying the system to a single-pendulum configuration for this first experiment.

The idea is to apply the same input signal to both the real system and the model and compare the evolution of both systems. Three separate non-trivial current reference signals were used. The signals were built by summing the first five odd harmonics of three specific waveforms: a square wave, a saw-tooth wave, and a triangle wave. The reference torque profiles are depicted in green in the top graphs of Figure 2.3.

This reference signal is applied to both the real system and the model in simulation, with the resulting position evolution depicted in Figure 2.3. The bottom graphs illustrate real position evolution in dotted red and simulation data in blue.

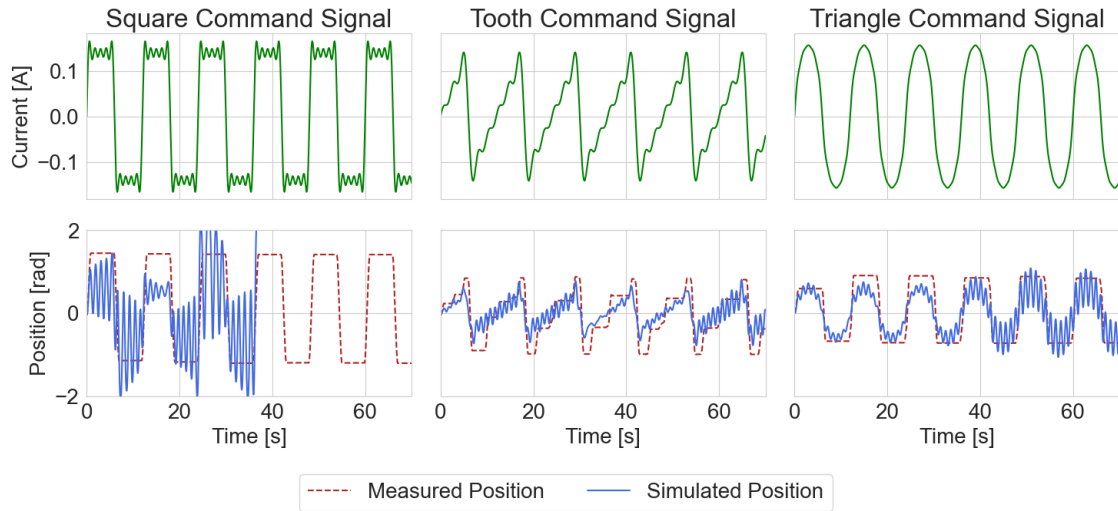


Figure 2.3: Comparison of reference torque profiles applied to hip motor, with the knee of the robot fixed, simplifying the system to a single-pendulum. The signal is applied both on the real system and on the model. Real position evolution is shown in dotted red and simulated position evolution in blue.

It is observed that the simulation doesn't accurately represent reality. For each waveform, oscillations are observed due to the intrinsic mathematical behaviour of an undamped single-pendulum. For a square wave torque reference, the model diverges completely from real behaviour.

2.3.2 Friction compensation

One identified cause of the simulation's inaccuracy is that the model doesn't take into account friction yet. By adding friction compensation, we can expect a much better approximation of reality. Equation 2.2 is then revised as:

$$\ddot{q} = M(q)^{-1}(\tau - \tau_f - C(q, \dot{q})) \quad (2.4)$$

The friction torque τ_f is explicitly added. As discussed in Section 1.3.2, friction in DC motors can be modelled in several ways.

Initially, the Stribeck curve was considered, which is often used to model friction in geared DC motors [17, 23]. The Stribeck model involves identifying four parameters: Coulomb friction τ_c , viscous friction K_v , Stribeck velocity $\dot{q}_s$, and an empirical parameter δ . The equation is recalled here:

$$\tau_f = \text{sign}(\dot{q}) \left(\tau_c + (\tau_s - \tau_c) e^{-(\dot{q}/\dot{q}_s)^\delta} \right) + K_v \dot{q} \quad (2.5)$$

Despite lot of attempts to apply the Stribeck model through numerous experimental approaches, achieving the desired accuracy was challenging. The initial limitations in system knowledge and the complexity of Stribeck model motivated the choice to start with a simpler, yet effective, friction model. We disregard then Stribeck effect from Equation 2.5, and choose the following friction model:

$$\tau_f = \text{sign}(\dot{q}) \tau_c + K_v \dot{q} \quad (2.6)$$

with two parameters to identify : the Coulomb torque τ_c and viscous parameter K_v . While simplifying the parameter identification process, starting with a simpler model allowed for a clearer understanding of the system. This foundational model serves as a basis for future enhancements, such as incorporating the Stribeck effect through optimization-based approaches.

2.4 Conclusion

This chapter has presented the real robot and the choice of model for it: a 5-link bipedal robot. The forward dynamics of the model have been derived, and an initial experiment was conducted on one of the hip joints by controlling the Dynamixel in torque through current. A comparison between the real experiment and the simulation highlighted the necessity for friction compensation to accurately represent the real system. A first simple friction model was chosen, incorporating both dry and viscous components of friction.

The next chapter will present two methodologies for identifying the parameters of the friction model: a basic experimental approach and an optimization-based approach.

Chapter 3

Parameter identification

The objective of this chapter is to identify the parameters of the robot model, with a focus on the friction characteristics of the actuators. Two parameter identification methodologies are presented, offering a dual perspective on modelling.

Initially, preference is given to the optimization-based methodology, which directly calibrates model parameters using experiments on the real system. However, initial attempts revealed inconsistencies. It was therefore necessary to switch to a more experimental methodology to gain a better understanding of Dynamixel behaviour. This less direct approach is based on a series of experiments, where each experiment isolates and identifies specific parameters using the DC motor's electrical and mechanical equations. This experimental approach enabled the identification of an initial model, and it was found that current measurements were unreliable. As a result, a switch to PWM control was necessary to obtain consistent results. This finally enabled to go back with the optimization-based methodology, successfully identifying the model parameters.

By detailing both methodologies, the text aims to illustrate the multifaceted nature of experimental research.

3.1 Basic experimental parameter identification

This section details the first approach towards identifying the parameters of both the DC motor model and its friction. The methodology revisits the fundamental equations of DC motor and analyzes specific regimes—namely, the acceleration and deceleration phases—to isolate and estimate various parameters.

The primary goal is to determine the parameters of the motor's friction model. However, this approach necessitates first identifying the intrinsic parameters of the motor. Once these motor parameters are known, they can be used to estimate the friction parameters: Coulomb torque (τ_c) and the viscous damping coefficient (K_v). The DC motor can be described by the following fundamental equations, as already presented in Section 1.3:

- **Electrical Equation:**

$$U_a = R_a i_a + L_a \frac{di_a}{dt} + k_e \omega_m \quad (3.1)$$

In this equation, U_a and i_a are respectively the motor's armature voltage and current, R_a is the armature resistance, L_a the inductance, k_e is the back electromotive force (EMF) constant, and ω_m the motor's shaft rotational speed.

- **Electromagnetic torque Equation:**

$$\tau_{em} = k_t i_a \quad (3.2)$$

where τ_{em} is the motor torque and k_t is the torque constant.

- **Mechanical Equation:**

$$J \frac{d\omega_m}{dt} = \tau_{em} - \tau_f \quad (3.3)$$

where J is the inertia and τ_f is the friction torque, modelled by the dry friction torque τ_c and the viscous damping coefficient K_v .

The time constant of the electrical equation is much smaller than that of the mechanical equation; in other words, the electrical dynamics are much faster than the mechanical dynamics. Therefore, we assume that the inductive voltage drop across the armature windings is negligible in Equation 3.1, leading to the simplified electrical equation:

$$U_a = R_a i_a + k_e \omega_m \quad (3.4)$$

Moreover, the analysis proceeds in the joint space, where a mathematical relationship between the motor's shaft rotational speed, ω_m , and the angular velocity of the joint, $\dot{q}$, can be expressed as follows:

$$\omega_m = k_{gear} \times \dot{q} \quad (3.5)$$

where the gear ratio constant k_{gear} is equal to:

- 353.5:1 for Dynamixel XH430-W350
- 212.6:1 for Dynamixel XH430-W210

The analysis focus first only on the motor of the hips, namely the Dynamixel XH430-W350. The same approach can be followed to identify the parameters for the second model of motors.

To recapitulate, the electrical and mechanical equations can be written as:

$$\begin{cases} U_a = R_a i_a + \overbrace{k_e k_{gear}}^{k'_e} \dot{q} \\ J \frac{d\dot{q}}{dt} = k_t i_a - \tau_c - K_v \dot{q} \end{cases} \quad (3.6)$$

For simplification, we introduce the constant $k'_e = k_e k_{gear}$. By conducting specific experiments, we can isolate and measure the parameters R_a , k_e , and $J = J_{motor}$. The total inertia of the system J is only composed of the motor's inertia J_{motor} , as the experiments will be conducted in no-load conditions. The torque constant k_t can be directly estimated using the datasheet of the Dynamixels, as stated in Section 2.3.1. Finally, these measurements allow us to solve for the friction parameters τ_c , and K_v using the above equations.

The following sections detail the experimental setup and the methodologies followed to identify these parameters, starting with the intrinsic motor parameters and progressing to the friction model parameters.

3.1.1 Experimental Setup and Data Collection

The experiments are conducted on Dynamixel units directly, dismounted from the robot. The data collection is performed using the Dynamixel Wizard 2.0 tool, which achieves a theoretical sampling rate of 1 ms, though a practical rate of approximately 2 ms is observed. This choice allows for a higher sampling rate compared to the LabVIEW Virtual Instrument (VI) developed for full robot control, which achieves a sampling rate of 20 ms. This allows for more precise observations of the motor's step responses.

3.1.2 Armature resistance estimation

The armature resistance value is estimated through stall tests on the Dynamixel, with the output shaft immobilised and the input voltage incrementally increased. During these tests, both voltage and current are measured, as shown in Figure 3.1 and 3.2.

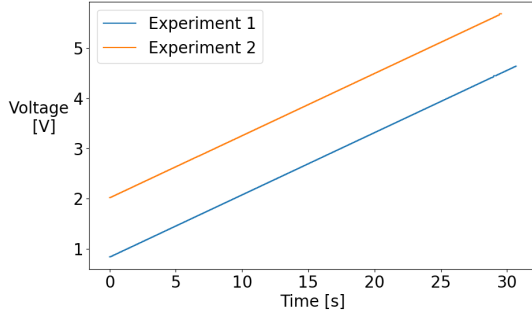


Figure 3.1: Evolution of voltage over time during two stall tests

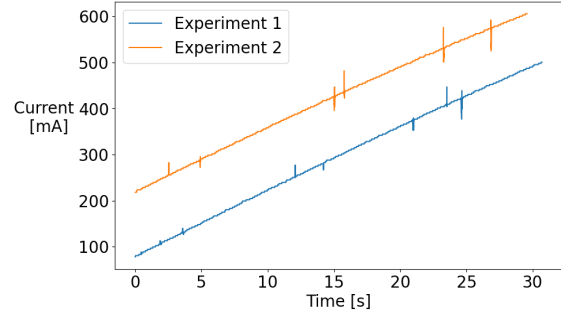


Figure 3.2: Corresponding current measurements during two stall tests

Given the motor's static condition, the electrical equation for the DC motor 3.4 gives a direct proportionality between voltage and current, defined by the armature resistance R_a . Figure 3.3 presents a scatter plot of the voltage-to-current ratio ($\frac{U_a}{i_a} = R_a$) from the two separate experiments. The computed mean from both experiments provides an estimated armature resistance of $R_a = 9.3756[\Omega]$.

3.1.3 Back EMF constant estimation

Steady-state experiments were performed to estimate the back EMF constant k'_e , with a constant voltage applied through PWM over a range from -100% to 100%, across 108 experiments with varied PWM inputs. The system stabilise to steady-state, where current, voltage, and angular velocity are constants. The back EMF constant k'_e is then isolated in Equation 3.6:

$$k'_e = \frac{U_a - R_a i_a}{\dot{q}} \quad (3.7)$$

Measurements of voltage, current, and angular velocity in steady-state allowed this estimation, illustrated in Figure 3.4.

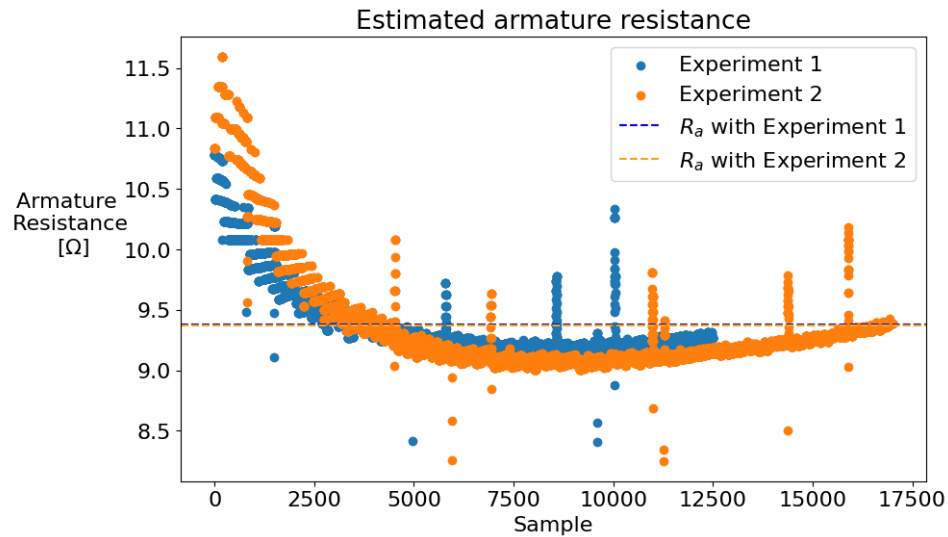


Figure 3.3: Voltage-to-Current Ratio and the calculated mean giving the estimated armature resistance R_a .

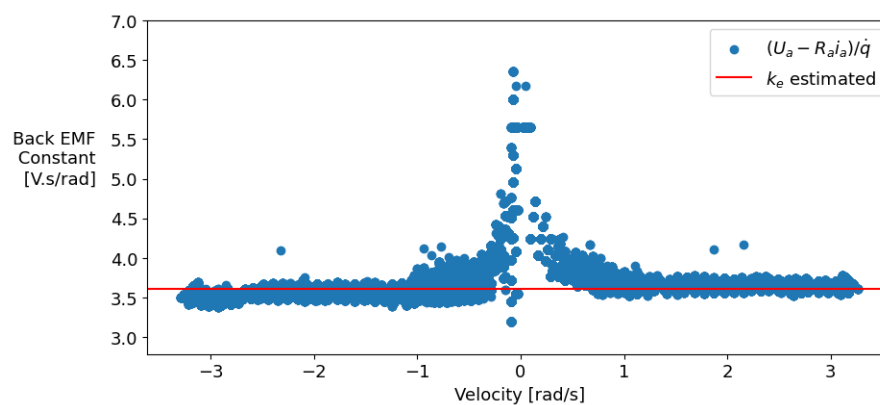


Figure 3.4: Estimation of the back EMF constant through steady-state experiments.

At velocities near zero, the precision of measurements (current, voltage, and velocity) significantly impacts the calculated k'_e . This effect is illustrated in Figure 3.5, where at low voltage commands (corresponding to low velocities), the current measurements approach the sensor's quantum level of 2.69 mA .

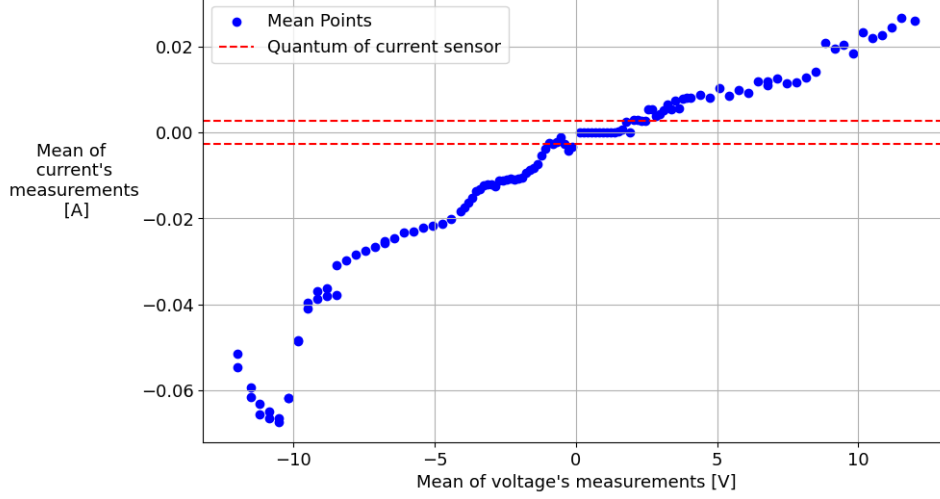


Figure 3.5: The relationship between voltage commands and measured current in the 108 experiments. The quantum level of current is shown with red dotted lines. The mean values are taken from steady-state measurements.

To mitigate the impact of these measurement limitations on parameter estimation, the analysis was restricted to velocities greater than 1 [rad/s] in absolute value. This approach gives a back EMF constant of $k'_e = 3.6103 \text{ [V} \cdot \text{s/rad]}$.

3.1.4 Inertia, Coulomb's torque, and viscous component estimation

To determine the inertia J_{motor} of the motor and gearbox assembly, the Coulomb's torque, and the viscous damping component, two types of experiments are conducted on the Dynamixel unit under no-load conditions :

1. **Acceleration Phase:** A sudden voltage input is applied to the motor (step response experiment).
2. **Deceleration Phase:** The voltage source is disconnected, allowing the motor to decelerate naturally due to friction losses.

During both experiments, the resultant angular velocity ($\dot{q}$) response over time is observed. It is then possible to isolate and estimate the parameters of the Dynamixel.

No-load experiments - Acceleration

The joint's angular velocity response to a sudden 12 V voltage application is observed. The system of Equation 3.6 reduces to the following differential equation :

$$J \frac{d\dot{q}}{dt} = k_t i_a - \tau_c - K_v \dot{q} \quad \text{where} \quad i_a = \frac{U_a - k'_e \dot{q}}{R_a} \quad (3.8)$$

Simplifying further :

$$J \frac{d\dot{q}}{dt} = \left(\frac{k_t U_a}{R_a} - \tau_c \right) - \left(\frac{k_t k'_e}{R_a} + K_v \right) \dot{q} \quad (3.9)$$

This last differential equation has as general solution :

$$\dot{q} = C_1 e^{-t/\tau_1} + \left(\frac{k_t U_a}{R_a} - \tau_c \right) / K_v \quad (3.10)$$

where :

$$\tau_1 = J / \left(\frac{k'_e k_t}{R_a} + K_v \right). \quad (3.11)$$

The time evolution of the joint's speed following a step input of 12 V is depicted in Figure 3.6.

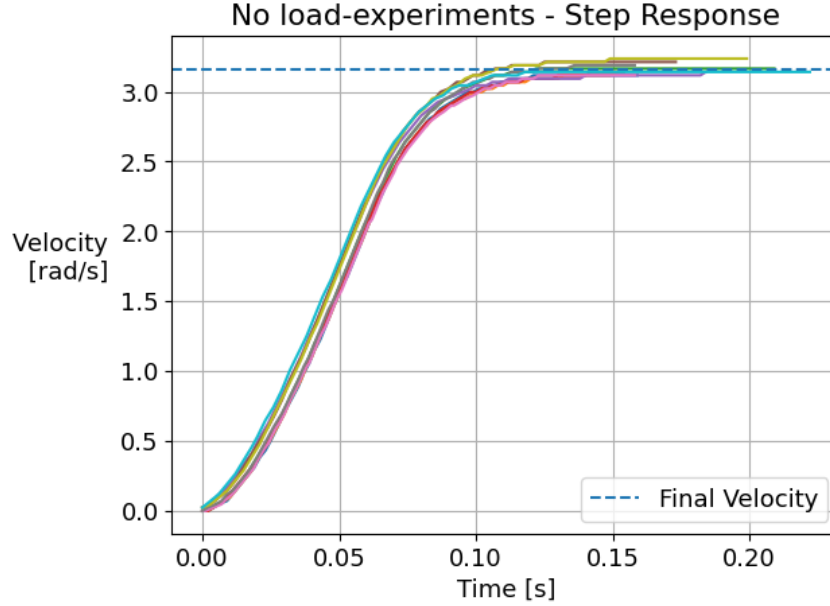


Figure 3.6: Time evolution of joint speed for a 12 V step input, for 10 experiments. The mean of all the final velocities reached is shown in blue dotted lines.

The 63.2% time constant is determined to be $\tau_1 = 0.0586$ s.

No-load experiments - Deceleration

As second experiment, the joint's speed time evolution is observed in Figure 3.7 when disconnecting the DC voltage source of the motor. This figure illustrates the joint's deceleration curve, highlighting the effects of friction. In this particular scenario, the Differential Equation 3.8 simplifies to :

$$J \frac{d\dot{q}}{dt} = -\tau_c - K_v \dot{q} \quad (3.12)$$

Where the general solution is:

$$\dot{q} = C_2 e^{-t/\tau_2} - \tau_c / K_v \quad (3.13)$$

with:

$$\tau_2 = \frac{J}{K_v}. \quad (3.14)$$

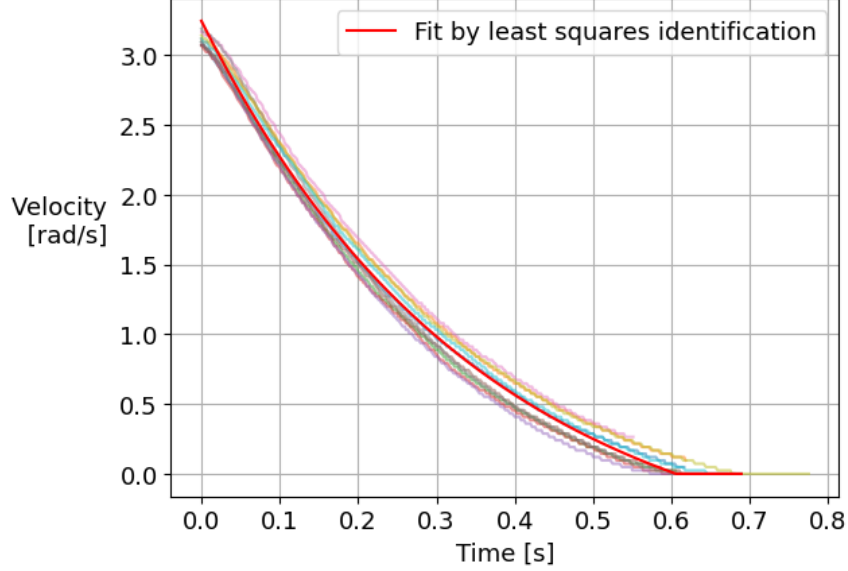


Figure 3.7: Observed joint speed during deceleration without voltage input. The data from 10 experiments are fitted with a model (red line) using least squares identification, which closely matches the experimental results ($R^2 = 0.991$).

To estimate the parameters, the `scipy.optimize.curve_fit` algorithm in Python is used, performing a least-squares fit on the experimental data. This process leads to the following parameter values:

- The integration constant : $C_2 = 3.85 \text{ [rad/s]}$
- The time constant : $\tau_2 = 0.332 \text{ [s]}$
- The ratio : $\tau_c/K_v = 0.583 \text{ [rad/s]}$

No-load experiments - System's solution

The system's mechanical parameters are determined by solving the Equation 3.11 of τ_1 , Equation 3.14 of τ_2 , and $\tau_c/K_v = 0.583$. The derived numerical values are as follows:

$$\begin{cases} J = J_{\text{motor}} = \frac{k'_e \tau_2}{R_a(1-\tau_2/\tau_1)} = 0.07305 \text{ [kg.m}^2\text{]} \\ K_v = \frac{J}{\tau_1} = 0.22 \text{ [N.m.s/rad]} \\ \tau_c = 0.583 \cdot K_v = 0.128 \text{ [Nm]} \end{cases} \quad (3.15)$$

3.2 Shift from current control to PWM control

Now that the parameters of the friction model have been identified, we can add friction into the model and perform new simulations to assess the improvements. The exact same methodology is followed as in Section 2.3.1, but now friction is taken into account during simulation. To recall, the experiments are conducted on Dynamixel units mounted on the real robot. In these experiments, the robot's knee is fixed, there is no ground interaction, and the input signal is applied to the hip motor in current control. The dynamical model is:

$$\begin{aligned} \ddot{q} &= M(q)^{-1}(\tau - \tau_f - C(q, \dot{q})) \\ \Leftrightarrow \ddot{q} &= \frac{k_t i_a - (\tau_c + K_v \dot{q}) - C(q, \dot{q})}{J_{\text{external}} + J_{\text{motor}}} \end{aligned} \quad (3.16)$$

where J_{external} and $C(q, \dot{q})$ are both computed using `RigidBodyDynamics` package given the *URDF* model of the robot [19]. k_t , τ_c , K_v , and J_{motor} are the four parameters we have identified. The Dynamixel is controlled in current, and the same input is applied on the model. The three current references used are the ones already presented in green in Figure 2.3 of Section 2.3.1. Figure 3.8 presents the results obtained.

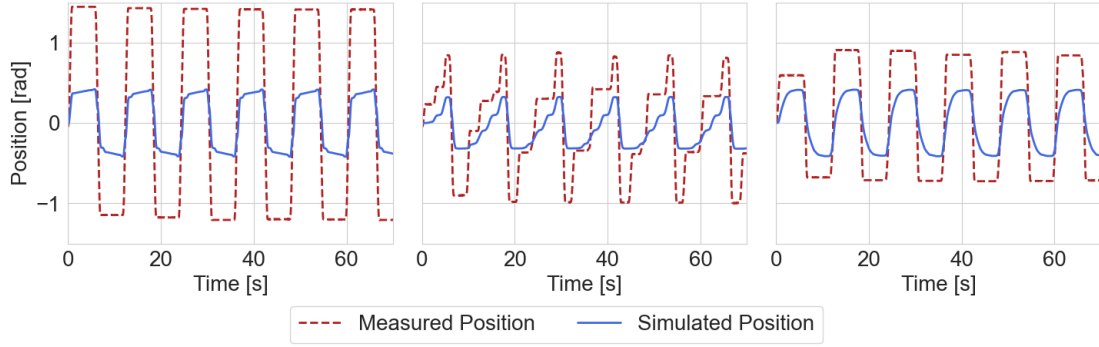


Figure 3.8: Comparison between real position evolution in dotted red with simulation results in blue for current (i_a) control and with friction compensation. The system is simplified to a single-pendulum.

We do not observe oscillations anymore, as was the case without friction compensation. To assess the fit of the model, the average R-squared is 53.29% and the average NRMSE is 0.6814, which is not satisfactory.

However, if we now use the measured value for U_a to simulate the system results are more convincing. This shift requires establishing a new relationship between torque and voltage. By isolating the current in the system of equations 3.6:

$$i_a = \frac{U_a - k'_e \dot{q}}{R_a}$$

we can rewrite the model 3.16 as:

$$\ddot{q} = \frac{k'_t U_a - (\tau_c + K'_v \dot{q}) - C(q, \dot{q})}{J_{\text{external}} + J_{\text{motor}}} \quad (3.17)$$

where the following two constants are introduced for simplification:

$$k'_t = \frac{k_t}{R_a} \quad \text{and} \quad K'_v = \left(K_v + \frac{k_t k'_e}{R_a} \right).$$

Using the measured U_a value from the experiments to simulate the model, the results are shown in Figure 3.9.

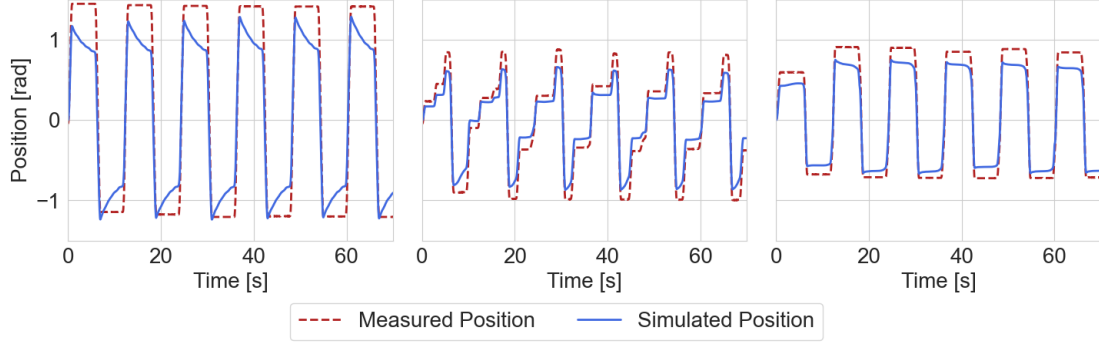


Figure 3.9: Same experiment as in Figure 3.8 except that the simulation uses the measured voltage U_a to simulate the real system.

By using the measured value of the voltage instead of the current to simulate the model, the results are significantly improved, with an average R-squared value of 94.11% and an average value of NRMSE of 0.2412.

Such strange behaviours of current control was also observed by the authors in [23]. The current measurements and control were found to be unreliable, motivating the shift towards a model with torque control through the voltage.

3.3 Optimization-based parameter calibration

Building on the fundamental insights obtained from the previous methodology, this section moves towards a more direct approach for parameter identification. It uses optimization algorithms to calibrate the model parameters using experimental data from the actual robot.

3.3.1 Experimental setup and data collection

The experiments are conducted on Dynamixel units mounted on the robot reduced to a single-pendulum configuration, such as previously. The model is given by Equation 3.17, where k'_t , τ_c , K'_v , and J_{motor} are four parameters to identify.

Compared to the previous methodology, we no longer use the Wizard 2.0 tool for control and data collection. Instead, the LabVIEW code is used, which allows to upload a file with the desired inputs for the Dynamixel at each time step. The sampling rate is now 20 ms.

3.3.2 Frequency analysis

Determining the appropriate reference signal for identifying the model parameters raises an initial question. A signal that adequately excites all the system states and accurately describes its behaviour is crucial. The aim of this frequency analysis is to characterize the system's frequency response.

This is achieved by applying an input signal consisting of a sequence of sinusoidal waves, each at a different frequency. The frequencies used range from 0.01 Hz to 25 Hz, with each frequency being applied for a sufficiently long duration to cover at least two full periods (for lower frequencies) or a minimum of five seconds (for higher frequencies). Moreover, each frequency segment starts and ends at a zero crossing to keep continuity in the signal. The sinusoidal signal for each frequency f and for a constant amplitude of $A = 2\text{ V}$ is simply: $s(t) = A \sin(2\pi ft)$.

As the experimental setup's sampling rate is 20 ms, corresponding to a sampling rate of 50 Hz, the maximum frequency that can be accurately sampled is 25 Hz, according to the Nyquist criterion. To prevent aliasing and to ensure accurate measurements, the input frequencies are limited to this value. The input signal and its corresponding response are illustrated in blue and in orange in Figure 3.10.

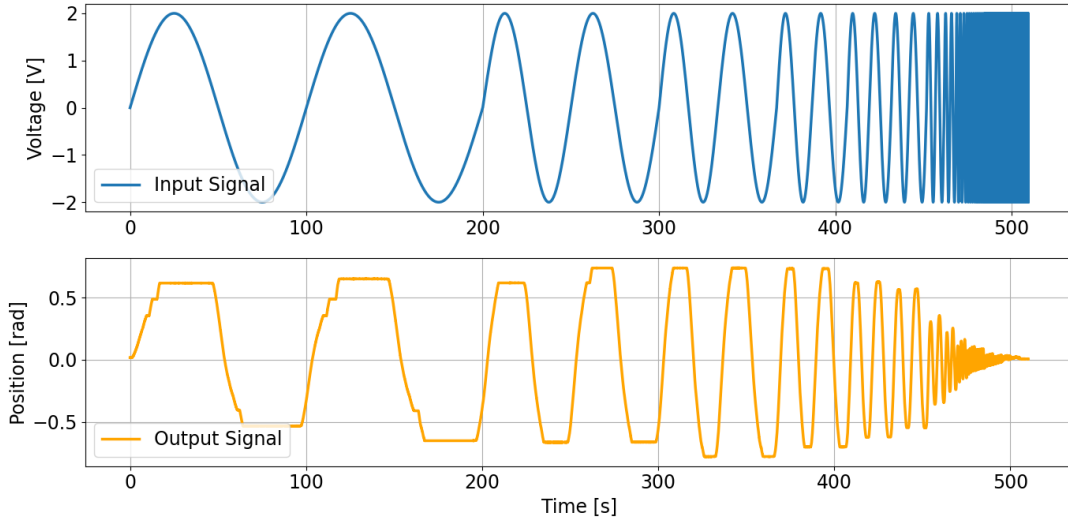


Figure 3.10: Sequence of sinusoidal waves in blue (input signal to the hip motor) and the position evolution in orange of the real system, simplified to a single-pendulum.

A subsequent Bode analysis of the system's response, normalized with a measured DC magnitude from a constant input experiment, is depicted in Figure 3.11. This analysis shows that the mechanical system behaves similarly to a low-pass filter. The Bode plot shows a cutoff frequency (where the magnitude decreases by 3 dB) approximately around 0.1 Hz, illustrating the system's attenuation of higher frequencies.

3.3.3 Reference signal construction

The previous analysis enables the construction of a suitable reference signal for calibrating the model's parameters. The signal is defined as follows:

$$S(t) = A(t) \cdot (\sin(2\pi ft) + \sin(2\pi 10ft) + \Delta(t)) \quad (3.18)$$

where:

$$\Delta(t) = \begin{cases} 0 & \text{for intervals with only two sine terms,} \\ \sin(2\pi 100ft) & \text{for intervals with three sine terms,} \end{cases}$$

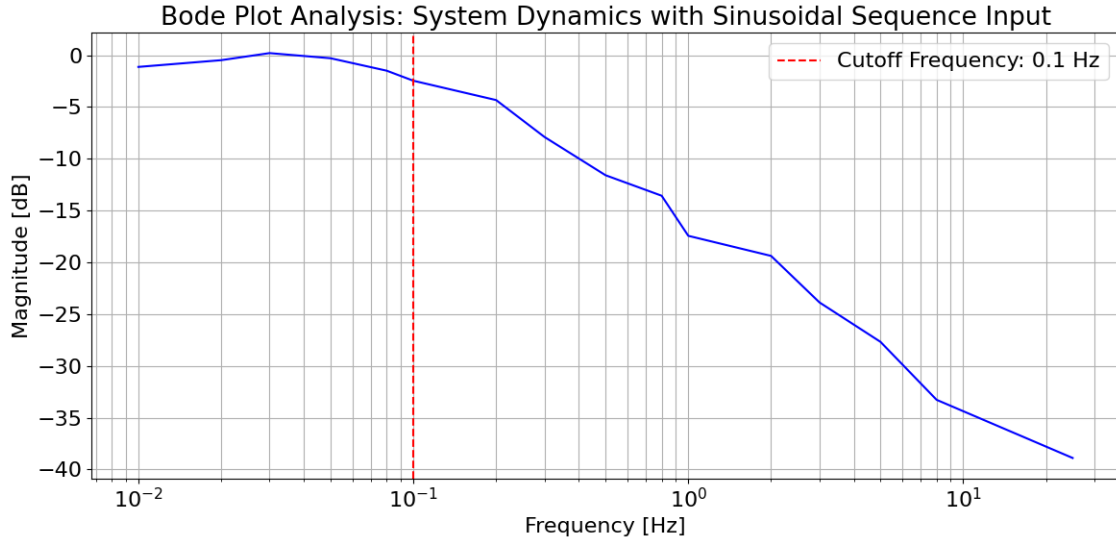


Figure 3.11: Bode plot illustrating the system's response as a low-pass filter with a cutoff frequency around 0.1 Hz. The normalization of the magnitude is done using a DC experiment, with an output of 0.74 [rad].

The amplitude $A(t)$ increases from 1V to 3V over the duration of the experiment. The frequency f is chosen to be 0.03 Hz, resulting in sine waves with frequencies of 0.03 Hz, 0.3 Hz, and 3 Hz. This choice is a balance between an adequate excitation of the system's states and a manageable experiment duration. The system is thus excited with sinusoids of low frequency, near cutoff, and above cutoff, keeping the system within its primary operating range and around the point where attenuation begins.

The resulting signal illustrated in blue in Figure 3.12 lasts 200 seconds and includes intervals with either two or three simultaneous frequencies, and three different amplitude levels. The corresponding response of the real system is in orange.

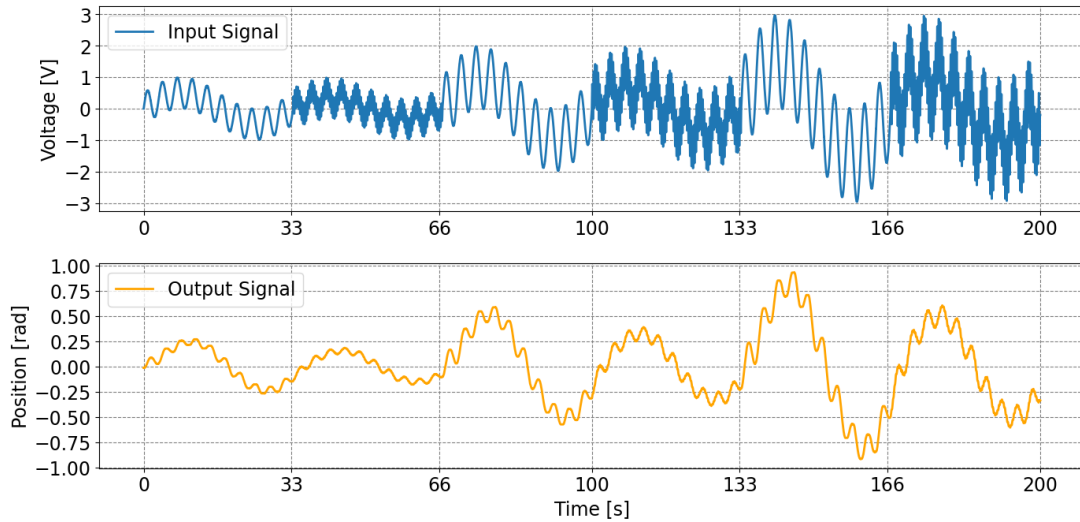


Figure 3.12: Voltage input signal (top) and position evolution for this input signal (bottom) of the real system.

3.3.4 Parameters calibration

The calibration of the model's parameters of Equation 3.17 was done using the `JuliaSim ModelOptimizer` package, a component of the `JuliaSim` suite developed by JuliaHub Inc [44]. This tool allows to solve inverse problems and to perform efficient model calibration. The model is defined in the `ModelingToolkit` framework, which will be further discussed in Chapter 5. The Julia code for parameter calibration is available in Annex B.

The estimated parameters value obtained with the previous methodology allows us to identify the order of magnitude of the parameters, and define a targeted search space, which is presented below:

$$\text{search_space} = \begin{cases} \tau_c & \rightarrow (0., 0.2), \\ K'_v & \rightarrow (0., 2.), \\ k'_t & \rightarrow (0., 1.), \\ J_{\text{motor}} & \rightarrow (0., 0.1) \end{cases} \quad (3.19)$$

Model calibration identify the parameters that minimize the cost function, the difference between observed position evolution and simulated one. The `calibrate` function within `JuliaSimModelOptimizer` automates this task.

The chosen optimization algorithm is the `IpoptOptimizer` (`Ipopt` - Interior Point OPTimizer), designed for large-scale nonlinear optimization problems, from the `Optimization` package [37]. The optimization problem is formulated as follows:

$$\begin{aligned} \min_x \quad & f(x) \\ \text{s.t.} \quad & g_L \leq g(x) \leq g_U \\ & x_L \leq x \leq x_U \end{aligned}$$

where x represents the vector of decision variables, $f(x)$ is the objective function to minimize, $g(x)$ the constraints vector, with g_L , g_U , x_L , x_U are the vectors of lower and upper bounds on the constraints and variables, respectively.

Applying this methodology to the experiment of Figure 3.12 and model 3.17 results in the optimization algorithm determining values for four parameters. The numerical values are in Table 4.1 for the previous methodology, denoted as Parameter Set 1, and for the calibration result, denoted as Parameter Set 2.

Parameter Set	τ_c (Nm)	K'_v (N·m/(rad/s))	k'_t (Nm/V)	J_{motor} (kg·m ²)
1	0.128	1.247	0.284	0.073
2	0.065	1.589	0.395	0.009

Table 3.1: Parameter values for the parameter set 1 and 2.

3.4 Conclusion

This chapter explored the identification of robot actuator parameters, looking at both the theoretical behaviour of DC motor and the practical challenges of real-world experimentation. Two distinct methodologies were presented.

The first approach, motivated by the need for deeper understanding of the Dynamixels, involved a series of experiments designed to isolate and identify specific motor parameters. This method, though less direct, proved satisfactory results, but moreover, allows us to identify a lack of consistency in current measurements.

Then, the optimization-based calibration approach provided a more direct identification of parameters. Frequency analysis and the choice of design for reference signal further showed the importance of understanding system behaviour in the selection of appropriate experimental conditions.

The next chapter will present the validation of the model developed in this chapter. This essential step will allow us to assess the accuracy and reliability of the model by comparing the prediction across several non-trivial experiments with measured data. This validation process will not only confirm the suitability of the models for control and simulation purposes, but also present areas for improvements.

Chapter 4

Model Validation

This chapter presents the validation of the dynamic model developed in the previous chapters. The goal is to determine the accuracy and reliability of the model predictions compared to results from real-world experiments. Model validation is critical for designing effective control systems, and particularly important for the next development of abstraction-based controllers, which rely on precise error bounds derived from the model.

The chapter is organized as follows: First, we describe the experimental methodology, detailing the series of tests with various signal waveforms and amplitudes. Then, the section introduces the concepts of simulation and prediction, defines the evaluation metrics (R-squared and NRMSE), and presents a comparative analysis of model performance using two parameter sets. A bound on the error is determined for the model with the best-performing parameter set. Finally, we explore potential model improvement by adding the Stribeck effect and evaluating a new parameter set.

4.1 Experimental setup

The experiments are conducted on one leg of the robot with the knee fixed, as previously presented in Sections 2.3.1 and 3.2. The Dynamixel of the hip is controlled in torque mode using PWM control. Eight non-trivial reference signals are chosen to validate the model:

- Three types of reference waveforms: square, saw-tooth, and triangle waves, as previously introduced in Section 2.3.1. Each waveform is tested at two different amplitudes: 1.5 V for lower amplitude movements and 3 V for full amplitude movements.
- Two walking gaits: recorded input signals from a walking gait applied to the robot. One gait is for slow walking and the other is for faster motion, to achieve different amplitudes. The walking gaits were identified by a previous master student in [36].

These eight reference signals are shown in Figure 4.1. They are applied to both the real system and the model. The final objective is to assess the error obtained with the model in predicting the state of the system, therefore the evolution of the joint's position and velocity are measured and will be compared with simulations. The Dynamixel allows a precision of 1.5×10^{-3} radians in position measurements and a precision of 0.0239 rad/s in velocity measurements.

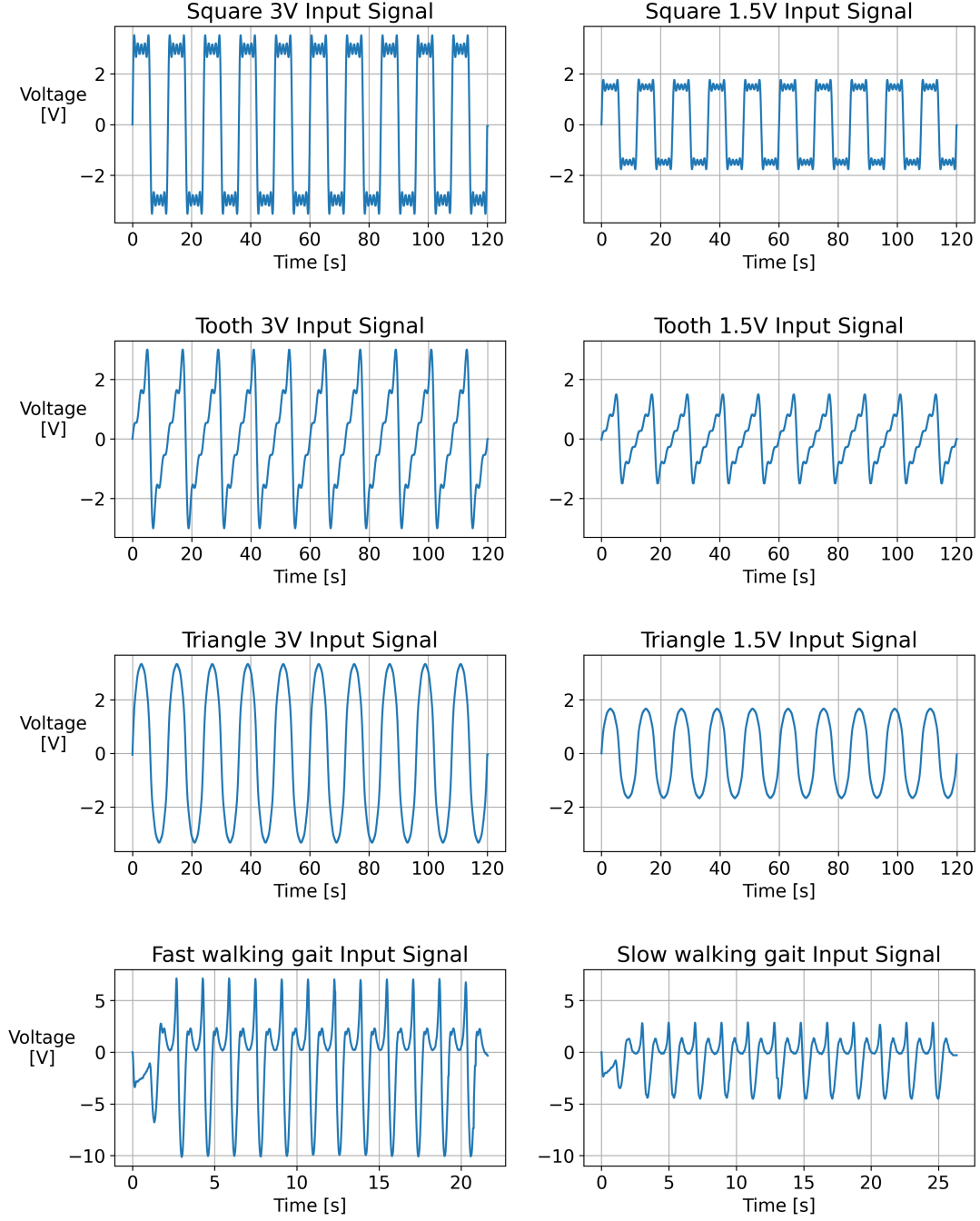


Figure 4.1: Reference input signals used in the experiments. From top left to bottom right, we have the square, tooth and triangle signals of amplitude 3V and 1.5V, and the fast and slow walking gaits.

4.2 Model performance evaluation

4.2.1 Comparison of predictor and simulator

First, let's define the difference between a simulator and a predictor, as MathWorks has defined it in [41]. In the context of system modeling, both simulation and prediction

are used to understand how a system behaves under given inputs, but they differ in the following sense:

- **Simulator:** computes the model's response using given input data and initial conditions over a specific period of time.
- **Predictor:** projects the model response into the future using current and past values of the measured input and output, as well as the initial conditions. It typically involves a prediction horizon, which defines how far into the future the prediction extends.

The key distinction is that simulation uses only the input data and initial conditions to generate outputs, while prediction uses past measurements of both inputs and outputs to predict future states. In a sense, a predictor with infinite prediction horizon tends toward a simulator.

These concepts are visually represented in the figure below:

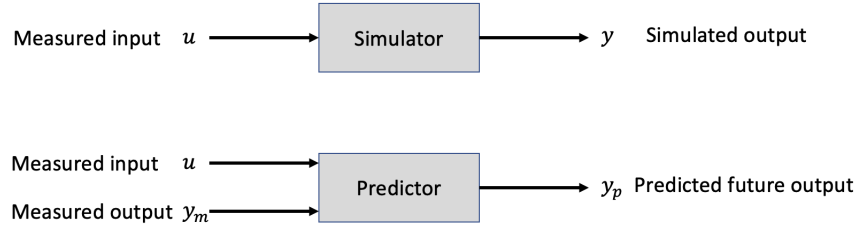


Figure 4.2: The first block diagram represents a simulator that takes measured input u to produce simulated output y . The second block diagram represents a predictor which uses both measured input u and measured output y_m to generate a predicted future output y_p .

4.2.2 Evaluation metrics

The performance of the model is assessed using two metrics: the R-squared (coefficient of determination) and the Normalized Root Mean Square Error (NRMSE). The first one, R-squared, quantifies the proportion of variance in the observed data that is explained by the model. An R-squared value close to 1 indicates that the model captures most of the variability in the data, whereas a value close to 0 suggests poor model fit. On the other hand, the NRMSE is calculated by dividing the RMSE by the standard deviation of the measured data. This normalization allows the RMSE to be interpreted in terms of the data's variability, allowing comparison across different datasets or models. A lower NRMSE indicates better predictive accuracy of the model.

For each parameter set, the following methodology is followed:

1. The model is used as a predictor with increasing prediction horizons for each of the eight experiments. The prediction horizons, ranging from 1 to 200, are selected as multiples of the sampling rate (20 ms). Simulations are also performed to obtain the simulation results (infinite horizon).
2. For each prediction horizon, the average values of NRMSE and R-squared are computed across the eight experiments for both position and velocity predictions.

The average is taken since no significant outliers were observed in the NRMSE and R-squared results.

4.2.3 Model performance

Parameter Set 1

The performance of the model with the first set of parameters is illustrated in Figure 4.6.

The results for parameter set 1 reveal several key trends and limitations, highlighting its suboptimal performance compared to the subsequent parameter set 2. For position prediction, a clear trend of decreasing accuracy with an increasing prediction horizon is observed. This is logically expected, as predicting further into the future typically increases error. For small prediction horizons, the goodness of fit, indicated by R-squared, remains high for position prediction, reflecting good short-term predictive capability.

On the other hand, the trend for velocity prediction exhibits a fluctuating pattern. This suggests that the model's ability to explain the variance in velocity data is inconsistent across different prediction horizons.

Overall, the results indicate that parameter set 1 leads to a model that tends towards a relatively poor fit as the prediction horizon increases. The fluctuating R-squared and NRMSE values for velocity suggest that this parameter set does not provide an optimal model for accurate predictions.

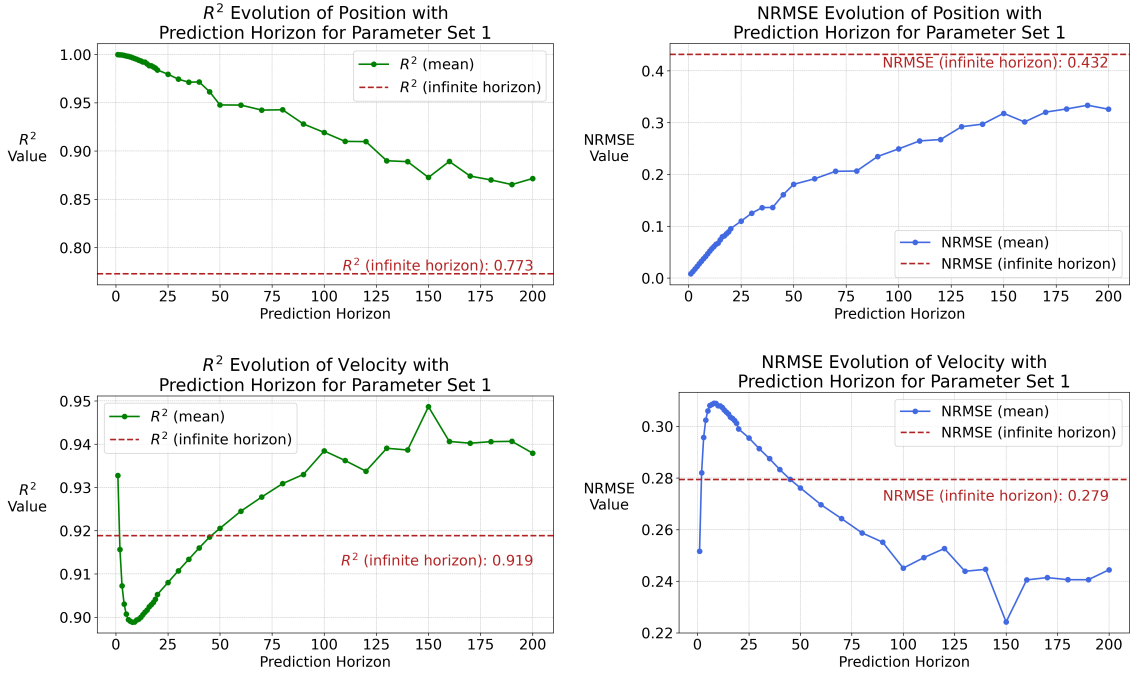


Figure 4.3: Evolution of R-square and NRMSE for position and velocity prediction with parameter set 1 over varying prediction horizons. The red lines is for simulation results.

Parameter Set 2

The results for parameter set 2 (Figure 4.7) show improvement in model performance compared to parameter set 1.

The R-squared values for position remain consistently high (above 0.98) throughout all prediction horizons. For the simulation the fit remains good with an R-squared value of 0.968. This signifies the model’s ability to explain the variance in position data, maintaining accuracy even for long-term predictions.

Similarly, the NRMSE values for position prediction are lower compared to parameter set 1, tending toward 0.157. This indicates a lower prediction error, reflecting improved model reliability for position prediction over varying horizons.

For velocity prediction, the R-squared values exhibit a stable pattern, remaining above 0.975, which indicates a consistent ability to explain the variance in velocity data. The NRMSE values for velocity are also lower and more stable compared to parameter set 1, with values around 0.154.

Given the better performance of the model with parameter set 2 across all evaluation metrics and prediction horizons, it is clear that this parameter set offers a better model for predicting both position and velocity of the system. Therefore, for the remainder of this study, we will use the model with parameter set 2.

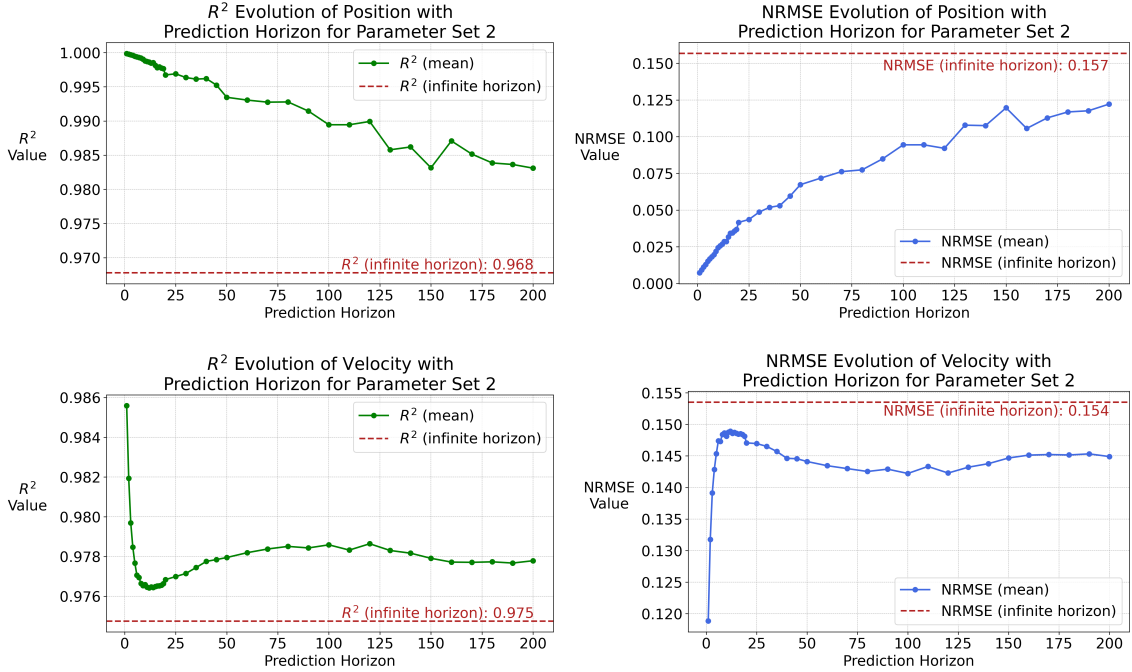


Figure 4.4: Evolution of R-square and NRMSE for position and velocity prediction with parameter set 2 over varying prediction horizons. The red lines is for simulation results.

4.2.4 Error bound

Abstraction-based controllers are correct-by-design controllers but require an estimation of the error obtained with the model. To estimate this error, the 99.9th percentile is used. The error distribution across experiments was observed to be centre around 0.

For instance, for a prediction horizon of 1 (equivalent to 20ms), the error on position is 0.0056 radians, and it linearly increase to 0.027 radians for a prediction horizon of 10, so 200ms in the future. The Dynamixels mounted on the robot are mechanically limited on a range of 180°. These errors correspond to approximately 0.178% to 0.86% of the total range. This indicates that we have some flexibility to increase the controller’s time

step if needed, as the errors remain within an acceptable range.

The figures below illustrate the evolution of the 99.9th percentile of the error for both position and velocity predictions with parameter set 2 over varying prediction horizons. The red lines represent the infinite horizon values from the simulations.

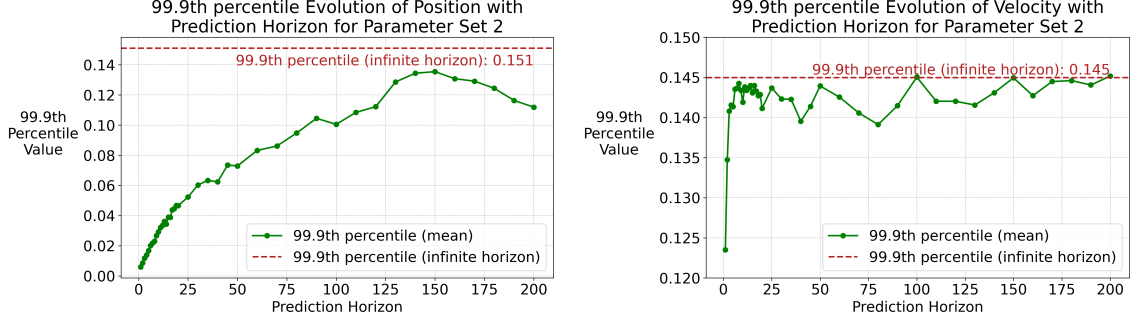


Figure 4.5: 99.9th percentile evolution of position in left figure and velocity in right figure over varying prediction horizon for parameter set 2. The red line represents the infinite horizon value obtained with a simulation.

4.3 Improvement with Stribeck effect

The exact same optimization-based methodology can be followed for a model that takes into account the Stribeck effect, as first introduced in Equation 2.5, and recalled here:

$$\tau_f = \text{sign}(\dot{q}) \left(\tau_c + (\tau_s - \tau_c) e^{-(\dot{q}/\dot{q}_s)^\delta} \right) + K_v \dot{q}$$

The parameter δ can be set to 1 to simplify calculations, a simplification also done in [17] and [23]. Since the Stribeck effect is significant only at very low velocities, and considering that our controller's objective will be to manage the motion of the bipedal robot dynamically at higher velocities, this simplification is justified. Two parameters need to be identified by the optimization algorithm: the static friction torque τ_s and the Stribeck velocity $\dot{q}_s$. The search space is defined as follow:

$$\text{search_space} = \begin{cases} \tau_c & \rightarrow (0., 0.2), \\ K'_v & \rightarrow (0., 2.), \\ k'_t & \rightarrow (0., 1.), \\ \tau_{s-c} & \rightarrow (0., 0.3), \\ \dot{q}_s & \rightarrow (0., 0.1), \\ J_{\text{motor}} & \rightarrow (0., 0.1) \end{cases} \quad (4.1)$$

The parameter $\tau_{s-c} = (\tau_s - \tau_c)$ allows for restricting the optimization algorithm to find values of the static torque τ_s that are greater than the values of the Coulomb torque τ_c , as it is physically expected.

The results of the optimization algorithm with the same reference signal as in Section 3.3.4 gives the Parameter Set 3:

Parameter Set	τ_c (Nm)	K'_v (N·m/(rad/s))	k'_t (Nm/V)	J_{motor} (kg·m ²)	τ_s (Nm)	$\dot{q}_s$ (rad/s)
1	0.128	1.247	0.284	0.073	/	/
2	0.065	1.589	0.395	0.009	/	/
3	0.108	1.354	0.389	0.039	0.11	0.0057

Table 4.1: Parameter values for the parameter set 1 and 2 (recall) and the new parameter set 3 with two new identified parameters.

4.3.1 Stribeck model performance

Incorporating the Stribeck effect into the model, the performance metrics indicate a nuanced improvement over previous iterations. Specifically, the R-squared values for position prediction show a comparable trajectory as observed with Parameter Set 2, except with an improvement of approximately 1.24% in the simulation results. Additionally, the NRMSE for position is slightly better at longer prediction horizons. Similar trends are observed in the velocity predictions, where both the R-squared and NRMSE metrics show improvements.

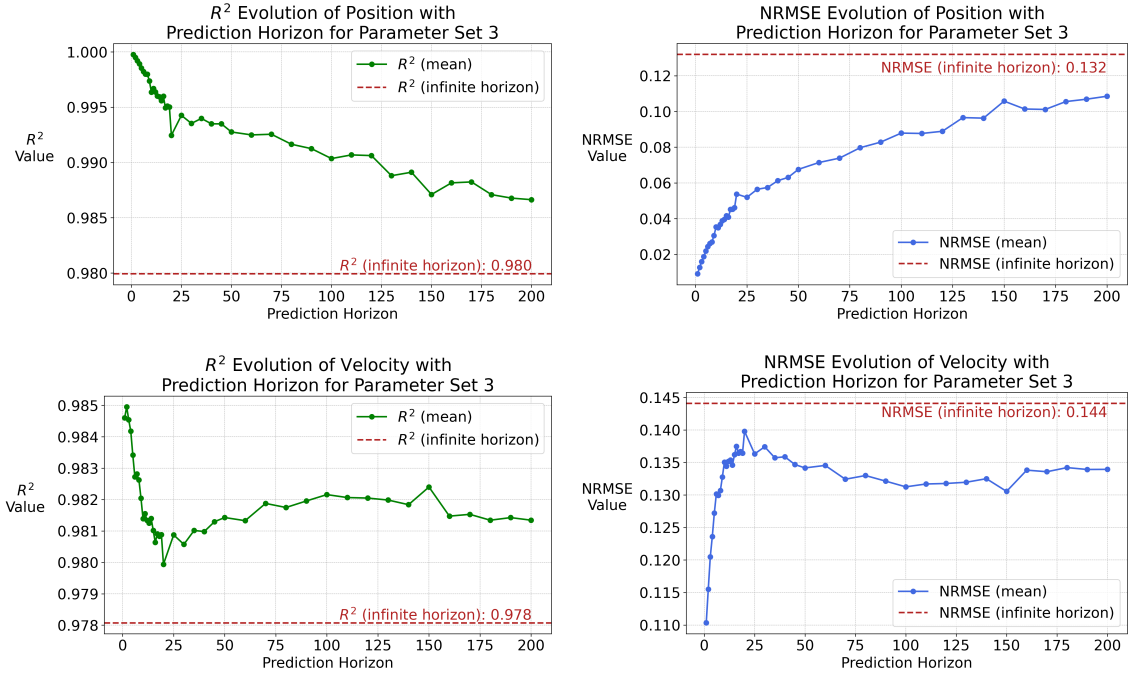


Figure 4.6: Evolution of R-square and NRMSE for position and velocity prediction with parameter set 3 over varying prediction horizons. The red lines is for simulation results.

Despite these improvements, the error analysis for this parameter set raises questions. The 99.9th percentile error for position prediction is 0.007 radians at a one-step prediction horizon and 0.034 radians at a ten-step horizon, which is higher compared to the values from Parameter Set 2. However, the error bounds for velocity prediction are lower than the ones obtained previously. An improvement from 0.123 to 0.09 is observed for a prediction horizon of 1, which represents a percentage decrease of approximately 26.83%. Given the improvements in the model, the Stribeck effect can be integrated into the final

model of the system. This decision is based on the need for a model that consistently provides both accurate and reliable predictions across all metrics.

The maximum velocity reachable by the Dynamixel is 24.5 rad/s. However, this speed is too high for the robot, potentially leading to high velocities of the end-effector. A more reasonable maximum speed can be derived from the maximal observed speed during experiments, which is approximately 2.7 rad/s. We can then set an arbitrary higher threshold at 4 rad/s. The error in velocity ranges from 0.09 for a time horizon of 1 to 0.128 for a time horizon of 10, which corresponds to 2.25% to 3.2% of the total range. These higher values compared to the position prediction may be due to the fact that velocity is the derivative of position, making it inherently noisier and more difficult to predict.

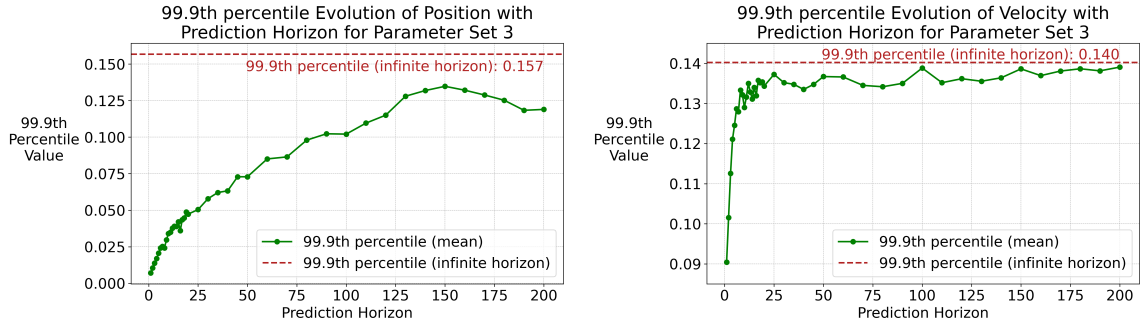


Figure 4.7: 99.9th percentile evolution of position in left figure and velocity in right figure over varying prediction horizon for parameter set 3. The red line represents the infinite horizon value obtained with a simulation.

4.4 Conclusion

In conclusion, this chapter has validated the dynamic model of the system. Through the evaluation of two parameter sets using metrics such as R-squared and NRMSE, we identified that the parameters determined through the optimization-based approach offered better performance. Moreover, the exploration of the Stribeck effect showed potential for further improvements. Iteratively building a model from simple to complex while achieving the required performance is essential.

In the next chapter, we will focus on the actual implementation of the validated model in Julia. We will detail a component-based model, improve the URDF representation of the robot, and refine the workflow for its construction. The challenges introduced by Coulomb friction and its non-smooth properties will also be discussed. Additionally, we will discuss potential future improvements to the model, such as adding a ground contact model and developing a comprehensive hybrid model of the system.

Chapter 5

Implementation in Julia

This chapter details the implementation and simulation of the model developed so far using the Julia language. First, the evolution of the model implementation is presented, along with the motivation for the choice of framework. Then, a workflow is introduced to simplify the process of URDF construction, allowing future mechanical modifications to be easily integrated into the robot's model. Finally, the simulation of dry friction is discussed, which poses challenges for numerical solvers. A finite state machine is implemented to clearly define the system and avoid any numerical instabilities.

5.1 Component-based model

The implementation of the model in Julia evolved during the year, leading to a component-based model using `ModelingToolkit.jl` (MTK) [31]. Initially, a first version of the model was built using `DifferentialEquations.jl` (DE) due to its ease of use in defining and solving systems of differential equations [25]. However, the need for linearization, a requirement for integrating with `Dionysos.jl`, motivated an exploration of alternative frameworks. MTK meets this requirement with automatic linearization of the model, and other advantages compared to DE.

First, MTK is built on top of `Symbolics.jl`, allowing symbolic representation and computation. The model is first written symbolically, then MTK performs symbolic transformations and structural simplifications of the model. Then, a set of ODE's is automatically generated by MTK, allowing the use of numerical solvers for simulations.

Secondly, MTK's component-based architecture allows for the modular development of models, where reusable building blocks (like joints, actuators, sensors) can be easily connected and modified. This modularity improves model organisation, readability, and maintainability, making it easier for future students and researchers to understand, modify, and extend the existing code-base.

Figure 5.1 represents the structure of the developed code-base for this thesis. In green, we have the definition of the model using MTK. It connects five components together. The Input Signal component provides the voltage reference U_a to an open-loop controller (OLController), which converts this voltage to the electromagnetic torque command τ . This controller component can be modified in the future to integrate the actual controller and simulate its behaviour.

The torque then goes through the DynamicsBias, FrictionTorque, and MassMatrix components, leading to the full dynamical equation. The URDF file is parsed with `RigidBodyDynamics.jl` to compute the dynamic bias $C(q, \dot{q})$ and the mass matrix $M(q)$

for a given state $(q, \dot{q})$ of the system.

The model is then converted to an ODEProblem from DE, as shown in blue in the figure, allowing the use of the numerous solvers from this library to solve the problem and compute the time evolution of the system's states.

This model can also be directly used in `JuliaSimModelOptimizer.jl`, as explained in Section 3.3.4, to calibrate the parameters k_t , τ_c , K_v , and J_{motor} using measurements from the real system.

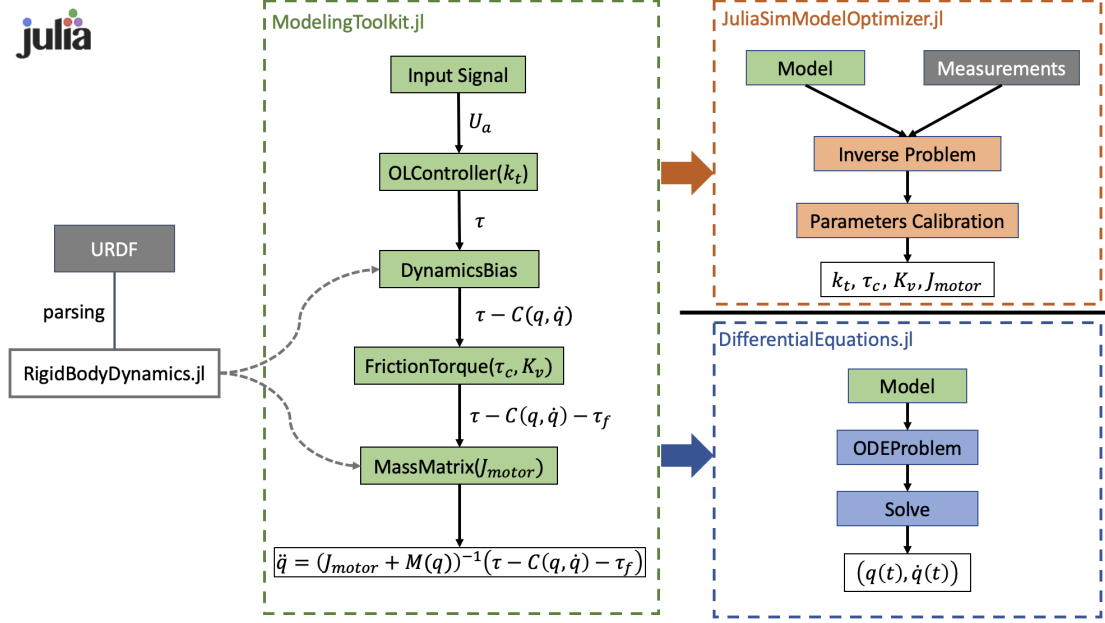


Figure 5.1: The structure of the codebase developed for this thesis. The green section represents the model definition using `ModelingToolkit.jl`, connecting various components. The orange section shows the parameter calibration process using `JuliaSimModelOptimizer.jl`. The blue section indicates the conversion of the model to an ODE problem for solving with `DifferentialEquations.jl`. The `RigidBodyDynamics.jl` package is used for parsing the URDF file to compute dynamic biases and mass matrices.

5.2 Enhanced URDF File Construction

The construction of the *URDF* file transitions from a manual approach using basic shapes to a more advanced method that uses exact geometrical shapes. This section discusses the initial model simplification, the transition to an advanced modeling approach, and the workflow for generating a *URDF* file that accurately represents the robot's structure.

The *URDF* model, developed in the previous year by previous master student, simplifies the robot's structure by representing each body segment with rectangular boxes, each defined by specific dimensions and mass [36]. In experiments focusing on a single leg with a fixed knee joint, this simplification effectively reduces the robot to a single pendulum configuration. This model was manually built using basic shapes, as depicted in Figure 5.2. The model is with flat feet as it was required for Zero Moment Point (ZMP) stability criterion in previous work on this robot.

The process of constructing the *URDF* file was refined through an advanced approach

with the objective to improve the workflow and the fidelity of the robot model. This involves using exact geometrical representations of the robot's components to get a model that closely mirrors the actual shapes and dimensions of the robot. The methodology's primary goal is to produce a *URDF* file with accurately defined joints, enabling precise inertia computation for each element and incorporating visually realistic representations for improved simulation experiences.

The robot is originally designed in FreeCAD. Initial attempts to convert the FreeCAD assembly to a *URDF* file with tools like the **FreeCAD** framework [38] faced problems due to its early development phase and unresolved issues. Then, alternative CAD tools can be used, for instance SolidWorks, with the use of the SolidWorks to URDF exporter [30]. This, however, necessitated extensive manual input, leading to further exploration of other CAD tools.

The discovery of the well-documented **onshape-to-robot** framework [43] led to the adoption of Onshape (a cloud-based CAD system) for this task. The workflow has several key steps, starting with the robot's original design in FreeCAD:

1. Export components into Onshape.
2. Reassemble within Onshape to address compatibility issues and prevent bugs, observed with FreeCAD assemblies imported into Onshape.
3. Define rotational joints in Onshape, following the guidelines available in the documentation [46].
4. Use the onshape-to-robot library in Python to convert the assembly into a *URDF* file.

This workflow simplifies the production of a *URDF* from CAD environments, leading to the generation of the robot's *URDF*, as shown in Figure 5.3.

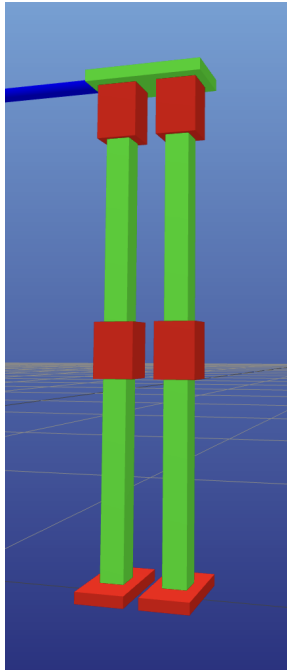


Figure 5.2: Initial URDF Model.

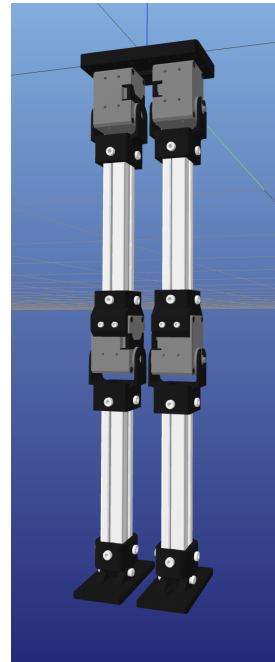


Figure 5.3: Enhanced URDF Model.

5.3 Simulation of Coulomb friction

The Coulomb friction model, due to the use of the ‘sign’ function, introduces discontinuities into the system dynamics. This creates a system that is not everywhere differentiable, posing challenges for numerical solvers that generally rely on smooth differential equations, leading to stability and accuracy issues.

Moreover, for small torque inputs below the Coulomb torque threshold τ_c previously identified, the total applied torque on the joint is less than the friction torque. Intuitively, any initiated movement in one direction is immediately countered by friction, which surpasses other torque sources and leads to system oscillation. This phenomenon results in solver oscillations due to inadequately defined behaviour in the code, as illustrated in Figure 5.4. Addressing this issue is important for stability of the simulation and overall performances.

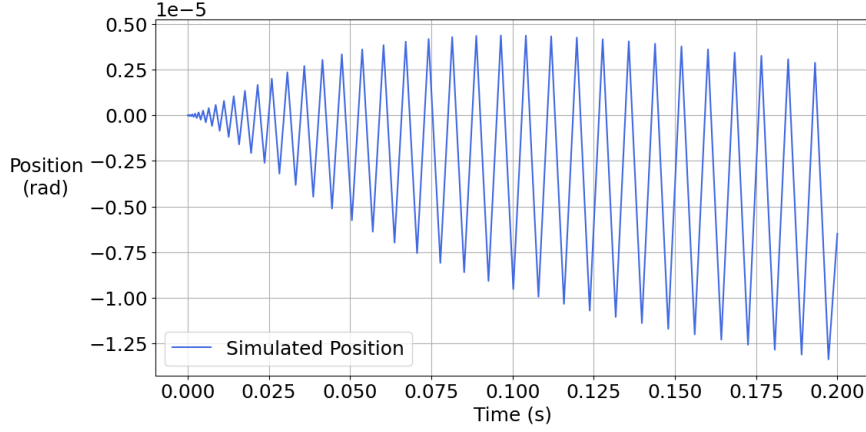


Figure 5.4: Solver oscillations due to Coulomb friction for a small period of time (200 ms). This phenomenon is also observed when the ‘sign’ function is smoothed with the hyperbolic tangent function $\tanh(\dot{q}/\delta)$, where δ is a smoothing parameter. Another approach proposed in the documentation of ModelingToolkit.jl is to define a *continuous_event* within the numerical solver, triggering it when $\dot{q} = 0$. This allows the solver to pause and reassess the system state using callbacks logic, improving its robustness. However, both of these solutions do not fully address the fundamental issue of the system definition in low-torque scenarios.

The methodology involves developing a Finite State Machine (FSM) based on the framework outlined in [45], as depicted in Figure 5.5. The three states are defined as follows:

- **STATIC**: The joint remains static, the angular velocity is zero ($\dot{q} = 0$).
- **MOVING_CCW**: The joint moves with a positive angular velocity, with friction being : $K_v \dot{q} + \tau_c$.
- **MOVING_CW**: The joint moves with a negative angular velocity, with friction being : $K_v \dot{q} - \tau_c$.

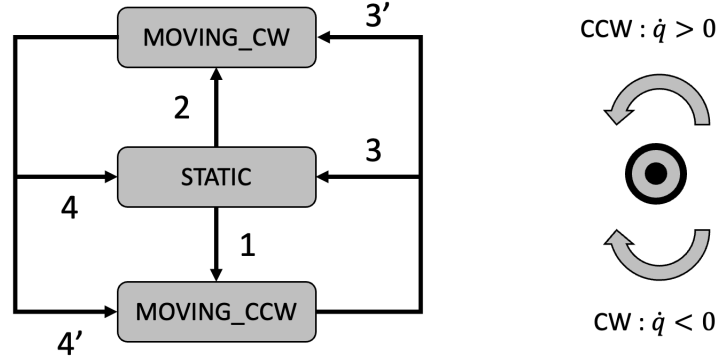


Figure 5.5: Finite State Machine illustrating the three states: STATIC, MOVING_CW, and MOVING_CCW.

Transitions between these states are dependent on the effective torque applied on the joint, τ_{eff} , and the static friction torque, τ_s . The transitions are given by the following conditions:

1. From STATIC to MOVING_CCW (1): Occurs if $\tau_{eff} > \tau_s$, where the joint begins to move with a positive angular velocity ($\dot{q} > 0$).
2. From STATIC to MOVING_CW (2): Takes place if $\tau_{eff} < -\tau_s$, resulting in the joint moving with a negative angular velocity ($\dot{q} < 0$).
3. From MOVING_CW to STATIC: This transition is initiated when the angular velocity becomes positive, indicating a potential change to counter-clockwise movement (3') or to a static state (3), depending on whether the net torque exceeds τ_s .
4. From MOVING_CCW to STATIC: Occurs when the angular velocity turns negative, indicating a transition to clockwise movement (4'), or to static (4) if the net torque is insufficient to overcome τ_s .

This approach ensures that to initiate movement, the effective torque must exceed the static friction torque; otherwise, the system remains static. Currently, the static friction torque (τ_s) is assumed to be equal to the estimated Coulomb's torque (τ_c). Future research could focus on accurately estimating τ_s to improve the model.

Chapter 6

Abstraction-based controller synthesis

A significant challenge with abstraction-based control is the curse of dimensionality, where the difficulty of control exponentially increases with the number of states. To address this, the problems introduced into Dionysos are of increasing complexity. Initially, the focus is on controlling the hip of the robot with the knee fixed, reducing the system to a single-pendulum configuration. Then, a more complex double-pendulum problem is introduced, where the hip and the knee of one robot's leg are controlled. This chapter briefly introduces both problems formulation.

Although the actual synthesis of the controller with Dionysos has not yet been achieved, the groundwork and problem setup are crucial steps towards this goal. This ongoing work involves detailed modeling, state space discretization, and defining appropriate constraints and objectives. Future work will involve implementing the controller synthesis within the Dionysos framework, optimizing the control strategies, and validating the results through simulations and real-world experiments.

6.1 Problem Formulation

For each system to be integrated into `Dionysos`, two primary functions are required:

1. A function of the type $x_{k+1} = f(x_k, u, t_s)$. For a given state x_k , input u , and time span t_s , this function predicts the next state x_{k+1} of the system.
2. A function of the type $\tilde{f}(x_k)$. For a given state x_k (operation point), this function returns the linearization of the system, providing the matrices A and B of a linear state-space representation:

$$\dot{x}(t) = Ax(t) + Bu(t)$$

In particular, for abstraction-based control we are interested in a manner to discretize the state-space of the system.

A possible approach to integrate a system into Dionysos is explained now. We consider a nonlinear continuous-time system defined by:

$$\dot{x}(t) = f(x(t), u(t))$$

The first step is to linearize this system around a chosen operating point, resulting in a linear approximation:

$$\dot{x}(t) = Ax(t) + Bu(t) + \epsilon_l$$

where ϵ_l represents the error introduced by linearization. The next step involves discretizing the linearized system using the matrix exponential for exact discretization:

$$x[k+1] = e^{A\Delta t}x[k] + \int_0^{\Delta t} e^{A(\Delta t-\tau)}Bu[k]d\tau$$

This approach allows for the estimation of error at each step, which is important to meet the requirements for synthesis of correct-by-design controllers.

6.1.1 Single-pendulum problem

The number of state variables in this configuration is the position of the joint and its velocity. This first simple problem allows the synthesis of a first controller with Dionysos, before going ahead with more complex problems.

6.1.2 Double-pendulum problem

The double-pendulum configuration increases the number of states to four: two states for the position of each joint, and two states for their velocities. The control objective is to drive the states of the system from one state to another while avoiding an obstacle.

The abstraction techniques used in Dionysos rely on defining ellipsoids and controllers within each ellipsoid. The discretization of the state space is achieved using ellipsoids. One advantage of this approach is the ability to directly define obstacles in the system's domain. The chosen obstacle, visible in Figure 6.1 in red, is designed to be easily reproducible with 3D printing for real-world experiments. By adjusting the size of the ellipsoid, the size of the obstacle can be increased or decreased, thereby modifying the problem's difficulty for the controller to transition from one state to the goal state. The constraints of the problem are as follows:

- Constraints on input voltage: $[-12, 12]$ V.
- Constraints on joint positions: The mechanical structure of the bipedal robot's legs limits the range to $[-\pi/2, \pi/2]$ rad.
- Constraints on joint speeds: According to the Dynamixel datasheet, the maximum speed range is $[-24.5, 24.5]$ rad/s, but a more reasonable and arbitrary velocity constraint for the joints is $[-4, 4]$ rad/s. This constraint is motivated by the maximum observed velocity value from the walking gaits, which is approximately 2.7 rad/s. This range considers higher values while avoiding unfeasible speeds within the joints' position range.

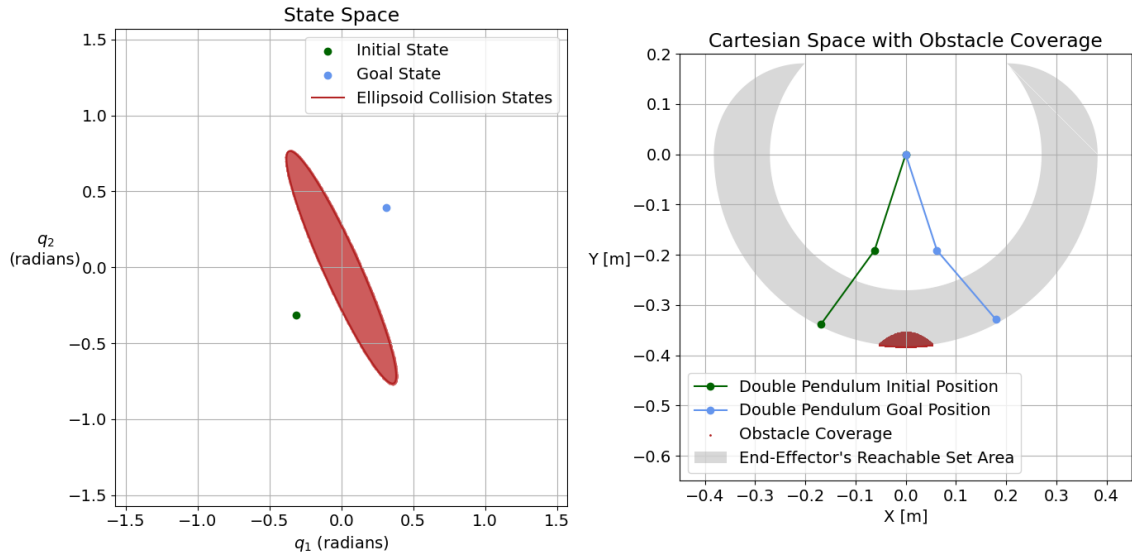


Figure 6.1: Visualization of the double-pendulum problem in both state space and Cartesian space with obstacle coverage. The left plot shows the state space with q_1 and q_2 (joint angles) as the axes. The green dot represents the initial state, the blue dot represents the goal state, and the red ellipsoid represents the collision states with the obstacle. The right plot shows the Cartesian space with X and Y coordinates. The corresponding initial and goal positions of the double pendulum are visible in green and blue. The obstacle is visible in red, and the gray shaded area illustrates the end-effector's reachable set area.

Chapter 7

Conclusion

This thesis has focused on the modeling and identification of a 5-link planar bipedal robot within the context of abstraction-based control. The robot’s dynamic model was derived using the Euler-Lagrange method, a general approach in robotics. The model parameters, particularly those related to friction of the actuators, were identified through two approaches. The model’s validity was confirmed through a series of experiments, demonstrating its capacity to predict the robot’s state under various conditions.

7.1 Summary

The study began with a comprehensive review of the state-of-the-art in bipedal robotics, focusing on the specific case of planar 5-link robot models. This review set the stage by outlining existing knowledge and identifying gaps, particularly in the application of abstraction-based control to real-world scenarios.

Next, the Euler-Lagrange equations of the robot were derived, and it was observed that friction compensation was necessary to control the joints in torque accurately.

To address this, two main approaches were followed to identify model parameters, particularly those related to friction of the actuators. Initially, experimental methods isolated specific parameters for detailed study. Then, an optimization-based method was applied, enhancing the model’s precision and reliability.

Validation of the developed model was confirmed by experimental tests with eight complex input signals. These experiments were essential to demonstrate the model’s predictive capabilities and to define a bound on its error. In addition, we identified and implemented a first improvement to the model, by adding the Stribeck effect.

Finally, the validated model was implemented in the Julia programming language using the ModelingToolkit.jl framework, and discussing the benefits of a component-based modeling approach.

To construct the model, several assumptions were made. The radius of the boom constraining the robot to the sagittal plane was assumed to be infinite, simplifying the robot’s movement to a planar surface. The mass and inertia of the boom were considered negligible. The foot-ground contact during the single support phase was modeled as an ideal pivot point with no sliding. The double support phase was approximated as an instantaneous impact with no rebound or slip. Additionally, the inductive voltage drop in the DC motors was neglected. These simplifications were made to manage the complexity of the model while still capturing the essential dynamics of the robot.

While significant advancements were made in the modeling and validation of the robot's dynamics, the ultimate goal of synthesizing an abstraction-based controller within the Dionysos framework was not achieved. The model developed provides a basis for future implementation and application to the real-world bipedal robot.

7.2 Future Work

Several potential improvements and future research directions can enhance the current model, driven by the need for a more precise model for controlling the robot's full degree of freedom using Dionysos for instance. First, further exploration of friction models, particularly the Generalized Maxwell-Slip (GMS) model, could lead to a more precise friction compensation. Additionally, improving the model of the foot's contact with the ground by incorporating tangential friction and normal constraints—whether soft, allowing for slight deformation, or hard, preventing any penetration—could enhance the overall accuracy.

On the control side, one significant challenge is the "curse of dimensionality" associated with abstraction-based control. To address this, initial steps could focus on controlling simplified models before tackling the full 7 degrees of freedom bipedal robot. This could involve mechanically blocking some degrees of freedom of the boom or applying dimensionality reduction techniques such as hybrid zero dynamics.

7.3 Final Thoughts

The work presented in this thesis has been challenging on several levels, the complexities and challenges encountered have been numerous, but they have also provided invaluable learning opportunities. This final section seeks to clarify the technical challenges encountered, but also highlights the lessons learned from this research experience.

First of all, understanding the behaviour of Dynamixel actuators, which operate like "black boxes" with undocumented internal controllers, was a major obstacle. The idea of switching to voltage-based torque control was not very intuitive, as we learned in electromechanics classes that torque is proportional to current, so naturally, it is controlled via current. Many weeks of fruitless experimentation finally led to trying voltage control, which taught a valuable lesson about maintaining structure and validating key points before implementing a solution. Listing all assumptions from the outset and looking for ways to simplify initial iterations to confirm core assumptions is now a key strategy, underlining the importance of starting with the simplest approach and iterating progressively towards greater complexity.

Implementation in Julia was also a challenge. As it's a fairly young language, libraries are constantly evolving, which often results in libraries that are either poorly documented, so difficult to understand quickly, or whose documentation is no longer up to date. This adds complexity to the basic challenge of learning a new programming language.

Finally, the final process proposed in this work is a synthesis of multiple trials and errors, ultimately leading to a solution that is as technically optimal as possible, whether for future improvements of the current work, or for reproducing the approach on similar cases. It starts with the efficient construction of a URDF model and ends with the definition of the model in a framework that allows parameter identification, simulation and integration with a wide range of other libraries.

Bibliography

- [1] C. Canudas de Wit et al. “A new model for control of systems with friction”. In: *IEEE Transactions on Automatic Control* 40.3 (1995), pp. 419–425. DOI: 10.1109/9.376053.
- [2] H. Olsson et al. “Friction Models and Friction Compensation”. In: *European Journal of Control* 4.3 (1998), pp. 176–195. ISSN: 0947-3580.
- [3] M. Grotjahn, M. Daemi, and B. Heimann. “Friction and rigid body identification of robot dynamics”. In: *International Journal of Solids and Structures* 38.10 (2001), pp. 1889–1902. ISSN: 0020-7683.
- [4] C. Chevallereau et al. “RABBIT: a testbed for advanced control theory”. In: *IEEE Control Systems Magazine* 23.5 (2003), pp. 57–79. DOI: 10.1109/MCS.2003.1234651.
- [5] E.R. Westervelt, J.W. Grizzle, and D.E. Koditschek. “Hybrid zero dynamics of planar biped walkers”. In: *IEEE Transactions on Automatic Control* 48.1 (2003), pp. 42–56. DOI: 10.1109/TAC.2002.806653.
- [6] Chevallereau Christine, Eric Westervelt, and J. Grizzle. “Asymptotically Stable Running for a Five-Link, Four-Actuator, Planar Bipedal Robot”. In: *I. J. Robotic Res.* 24 (June 2005), pp. 431–464.
- [7] T. Tjahjowidodo, F. Al-Bender, and H. Van Brussel. “Friction identification and compensation in a DC motor”. In: *IFAC Proceedings Volumes* 38.1 (2005). 16th IFAC World Congress, pp. 554–559. ISSN: 1474-6670.
- [8] Tegoeh Tjahjowidodo, Farid Al-Bender, and H. Brussel. “Model Based Friction Compensation in a DC Motor”. In: (Jan. 2005).
- [9] R.R. Waiboer, Ronald G.K.M. Aarts, and Jan B. Jonker. “Velocity dependence of joint friction in robotic manipulators with gear transmissions”. Undefined. In: *Proceedings of the ECCOMAS Thematic Conference Multibody Dynamics 2005, Advances in Computational Multibody Dynamics, 21-24 june 2005, Madrid, Spain.* II International Conference on Advances in Computational Multibody Dynamics 2005 ; Conference date: 21-06-2005 Through 24-06-2005. June 2005, pp. 1–19.
- [10] Nasser Sadati and Kaveh Hamed. “CPG Based Controller for a 5-Link Planar Biped Robot”. In: *2007 IEEE International Conference on Mechatronics.* 2007, pp. 1–6. DOI: 10.1109/ICMECH.2007.4280020.
- [11] Eric Westervelt et al. *Feedback Control of Dynamic Bipedal Robot Locomotion.* 2007, pp. 1–41. ISBN: 9781315219424. DOI: 10.1201/9781420053739.
- [12] Vincent Duindam and Stefano Stramigioli. *Modeling and Control for Efficient Bipedal Walking Robots: A Port-Based Approach.* 1st. Springer Publishing Company, Incorporated, 2009, pp. 1–15. ISBN: 3540899170.

- [13] Yuichi Tazaki and Jun-ichi Imura. “Planar bipedal locomotion control based on state discretization”. In: *Robotics and Autonomous Systems* 58.5 (2010), pp. 657–665. ISSN: 0921-8890.
- [14] Gianluca Garofalo, Christian Ott, and Alin Albu-Schäffer. “Walking control of fully actuated robots based on the Bipedal SLIP model”. In: *2012 IEEE International Conference on Robotics and Automation*. 2012, pp. 1456–1463. DOI: 10.1109/ICRA.2012.6225272.
- [15] Shishir Nadubettu Yadukumar, Murali Pasupuleti, and Aaron D. Ames. “Human-inspired underactuated bipedal robotic walking with AMBER on flat-ground, up-slope and uneven terrain”. In: *2012 IEEE/RSJ International Conference on Intelligent Robots and Systems*. 2012, pp. 2478–2483. DOI: 10.1109/IROS.2012.6386182.
- [16] Ivan Virgala and Michal Kelemen. “Experimental Friction Identification of a DC Motor”. In: *International Journal of Mechanics and Applications* 3 (Jan. 2013). DOI: 10.5923/j.mechanics.20130301.04.
- [17] Max Schwarz and Sven Behnke. “Compliant Robot Behavior Using Servo Actuator Models Identified by Iterative Learning Control”. In: *RoboCup 2013: Robot World Cup XVII*. Ed. by Sven Behnke et al. Berlin, Heidelberg: Springer Berlin Heidelberg, 2014, pp. 207–218. ISBN: 978-3-662-44468-9.
- [18] Hui-Hua Zhao et al. “Human-inspired multi-contact locomotion with AMBER2”. In: *2014 ACM/IEEE International Conference on Cyber-Physical Systems (ICCPS)*. 2014, pp. 199–210. DOI: 10.1109/ICCPS.2014.6843723.
- [19] Twan Koolen and contributors. *RigidBodyDynamics.jl*. Accessed: 2023-11-10. 2016. URL: <https://github.com/JuliaRobotics/RigidBodyDynamics.jl>.
- [20] Eric Ambrose et al. “Toward benchmarking locomotion economy across design configurations on the modular robot: AMBER-3M”. In: Aug. 2017. DOI: 10.1109/CCTA.2017.8062633.
- [21] Aaron D. Ames and Ioannis Poulakakis. “Hybrid Zero Dynamics Control of Legged Robots”. In: *Bioinspired Legged Locomotion: Control of Motion and Compliance*. Ed. by Maziar A. Sharbafi and André Seyfarth. Butterworth-Heinemann, 2017. Chap. 4.7, pp. 292–331.
- [22] Aaron D. Ames et al. “First steps toward formal controller synthesis for bipedal robots with experimental implementation”. In: *Nonlinear Analysis: Hybrid Systems* 25 (2017), pp. 155–173. ISSN: 1751-570X.
- [23] Rémi Fabre et al. “Dynaban, an Open-Source Alternative Firmware for Dynamixel Servo-Motors”. In: *RoboCup 2016: Robot World Cup XX*. Ed. by Sven Behnke et al. Cham: Springer International Publishing, 2017, pp. 169–177. ISBN: 978-3-319-68792-6.
- [24] Wen-Loong Ma et al. “Bipedal Robotic Running with DURUS-2D: Bridging the Gap between Theory and Experiment”. In: Apr. 2017. DOI: 10.1145/3049797.3049823.
- [25] Christopher Rackauckas and Qing Nie. “DifferentialEquations.jl—a performant and feature-rich ecosystem for solving differential equations in Julia”. In: *Journal of Open Research Software* 5.1 (2017).
- [26] Huihua Zhao et al. “Multi-contact bipedal robotic locomotion”. In: *Robotica* 35.5 (2017), pp. 1072–1106. DOI: 10.1017/S0263574715000995.

- [27] Martin Fevre, Bill Goodwine, and James P. Schmiedeler. “Design and Experimental Validation of a Velocity Decomposition-Based Controller for Underactuated Planar Biped”. In: *IEEE Robotics and Automation Letters* 3.3 (2018), pp. 1896–1903. DOI: 10.1109/LRA.2018.2806088.
- [28] Murtadha Awoda and Ramzy Al-Waily. “Parameter Estimation of a Permanent Magnetic DC Motor”. In: *Iraqi Journal for Electrical and Electronic Engineering* 15 (June 2019), pp. 28–36. DOI: 10.37917/ijeee.15.1.3.
- [29] Shuuji Kajita. “Linear Inverted Pendulum-Based Gait”. In: Jan. 2019, pp. 905–922. ISBN: 978-94-007-6045-5. DOI: 10.1007/978-94-007-6046-2_42.
- [30] Stephen Brawner and contributors. *SolidWorks to URDF Exporter*. https://github.com/ros/solidworks_urdf_exporter. Accessed: 2024-04-01. 2021.
- [31] Yingbo Ma et al. *ModelingToolkit: A Composable Graph Transformation System For Equation-Based Modeling*. 2021. arXiv: 2103.05244 [cs.MS].
- [32] Rafal Szczepanski et al. “Identification of mechanical parameters in servo-drive system”. In: Apr. 2021. DOI: 10.1109/PEMC48073.2021.9432595.
- [33] Mohamad Nazir Abdullah et al. “Experimental and numerical studies for parameters identification of direct current motor”. In: *Indonesian Journal of Electrical Engineering and Computer Science* 27 (Aug. 2022), pp. 592–600. DOI: 10.11591/ijeecs.v27.i2.pp592-600.
- [34] Min Dai, Xiaobin Xiong, and Aaron Ames. “Bipedal Walking on Constrained Footholds: Momentum Regulation via Vertical COM Control”. In: *2022 International Conference on Robotics and Automation (ICRA)*. 2022, pp. 10435–10441. DOI: 10.1109/ICRA46639.2022.9812247.
- [35] Eben60. *LabVIEW0.jl*. <https://github.com/Eben60/LabVIEW0.jl>. 2022.
- [36] Xing Chen. “Design and Control of a Bipedal Robot”. MA thesis. Ecole polytechnique de Louvain, Université catholique de Louvain, 2023.
- [37] Vaibhav Kumar Dixit and Christopher Rackauckas. *Optimization.jl: A Unified Optimization Package*. Version v3.12.1. Mar. 2023. DOI: 10.5281/zenodo.7738525. URL: <https://doi.org/10.5281/zenodo.7738525>.
- [38] Gaël Écorchard and contributors. *CROSS CAD and ROS, an Open-Source Synergy; ROS workbench for FreeCAD*. <https://github.com/galou/freecad.cross>. Accessed: 2024-04-01. 2023.
- [39] Jian Liao, Fuming Zhou, and Jianbo Zheng. “An Improved Parameter Identification Algorithm for the Friction Model of Electro-Hydraulic Servo Systems”. In: *Sensors* 23.4 (2023). ISSN: 1424-8220.
- [40] Jian Liao, Fuming Zhou, and Jianbo Zheng. “An Improved Parameter Identification Algorithm for the Friction Model of Electro-Hydraulic Servo Systems”. In: *Sensors* 23 (Feb. 2023), p. 2076. DOI: 10.3390/s23042076.
- [41] MathWorks. *Simulate and Predict Identified Model Output*. Accessed: 2024-05-29. 2023. URL: <https://www.mathworks.com/help/ident/ug/definition-simulation-and-prediction.html>.
- [42] Julien Calbert et al. *Dionysos.jl: a Modular Platform for Smart Symbolic Control*. 2024. arXiv: 2404.14114 [eess.SY].

- [43] Rhoban and contributors. *Converting OnShape assembly to robot definition (SDF or URDF) through OnShape API*. <https://github.com/Rhoban/onshape-to-robot>. Accessed: 2024-04-01. 2024.
- [44] JuliaHub Inc. *JuliaSimModelOptimizer: A Julia package for automated solution of complex inverse problems*. *JuliaSim Suite*. Accessed: 29-February-2024. URL: <https://help.juliahub.com/jsmo/stable/>.
- [45] D. Klaff. *Using state machines and callbacks to model friction and limit stops*. https://klaff.github.io/friction_state_machine.jl.html. Accessed: 2024-02-27.
- [46] *Onshape-to-Robot Documentation*. <https://onshape-to-robot.readthedocs.io/en/latest/>. Accessed: 2024-04-01.

Appendix A

LabVIEW to Julia connection

The controller synthesized with `Dionysos.jl` is in Julia, but the actual communication with the robot is done with LabVIEW. An interesting feature would be to integrate LabVIEW with Julia in order to compute control inputs based on the real state of the robot in Julia, and then apply these commands directly to the robot. This approach would leverage LabVIEW for its real-time characteristic and Julia for its computational efficiency. A GitHub repository [35] provides a solution to integrate Julia functions into LabVIEW Virtual Instruments, using ZeroMQ as communication library.

To assess the performance of the solution, systems of linear equations are solved in LabVIEW with two approaches:

- By using directly the built-in solver of LabVIEW
- By calling a Julia function from LabVIEW, solving the system with Julia, and sending back the result to LabVIEW.

The computation time taken by both methods for given matrix size is in Table A.1. The experiments are done on the Windows machine available in the lab.

Size N of the matrix ($N \times N$)	Julia Computation Time [ms]	LabVIEW Computation Time [ms]
7	1.5	0.06
35	1.6	0.4
175	8	6
350	23	26
700	80	120
2000	800	4000

Table A.1: Comparison of computation times between Julia and LabVIEW for different matrix sizes.

From the table, it can be observed that for smaller matrix sizes, LabVIEW's built-in method significantly outperforms the Julia method. However, as the matrix size increases, the gap in computational times between Julia and LabVIEW becomes smaller and, at some point, Julia becomes faster. We can conclude that depending on the task complexity, it can be better to perform computation through the Julia server.

Moreover, for some specific tasks it can be easier in terms of development to use Julia rather than complex Virtual Instruments built in LabVIEW.

Appendix B

GitHub Repository

The complete code used for this thesis, including data preprocessing, model calibration, optimization routines, and analysis scripts, is available in a GitHub repository at the following link: https://github.com/DeligneS/master_thesis. The repository is structured as described in the following section.

B.1 Repository Structure

The repository is organized into the following directories and files:

- **data/**: Contains all datasets used, divided into two folders: one for experiments conducted on the robot and one for simulations.
- **analysis/**: Contains folders for the analysis conducted in each chapter of this thesis.
- **src/**: This folder includes scripts for cleaning and preprocessing data, initial methodologies for parameter identification of the Dynamixels, and a folder with Python scripts for recurrent plots and analysis.
- **param-id/**: Contains the Julia scripts for parameter identification using the optimization-based approach.
- **URDF/**: Contains all the URDF files obtained with the Onshape to URDF workflow, and a folder with Julia scripts for visualization of the URDFs.
- **single_pendulum/**: Contains all the Julia scripts used for modeling the single-pendulum configuration of the robot.
- **double_pendulum/**: Contains the Julia scripts for modeling the double-pendulum system and a Python folder for the construction of the obstacle avoidance problem.
- **utils/**: Contains various utilities used across several Julia scripts, such as loading datasets, loading URDF files, and defining ModelingToolkit components.

Each folder contains README files that explain the purpose of the folder, how to run the scripts, and how to launch the local environments in Python or Julia.

UNIVERSITÉ CATHOLIQUE DE LOUVAIN
École polytechnique de Louvain

Rue Archimède, 1 bte L6.11.01, 1348 Louvain-la-Neuve, Belgique | www.uclouvain.be/epl