UNIVERSITÀ DEGLI STUDI DI PADOVA

DIPARTIMENTO DI INGEGNERIA DELL'INFORMAZIONE

MASTER THESIS IN CONTROL SYSTEM ENGINEERING

Adaptive Locomotion of Planar Snake Robots: Body Shape Control and Path-Following

SUPERVISOR

Prof. Damiano Varagnolo

University of Padua

CANDIDATE

Morselli Alberto

Student ID 2102413

MAIN-SUPERVISOR

Prof. Kristin Y. Pettersen

NTNU - Norwegian University of Science and Technology

CO-SUPERVISORS

PhD Candidate Ivan I. Gushkov

Dr. Amer Orucevic

NTNU - Norwegian University of Science and Technology

ACADEMIC YEAR 2024/2025

October 2025

Abstract

Snake robots constitute a class of *hyper-redundant*, articulated systems inspired by their biological counterpart. Their many degrees of freedom make them challenging to control, but provide unique locomotion capabilities in irregular and challenging environments.

However, the inherent complexity of full order models constrains the design of high-performance *model-based* controllers, especially for advanced tasks such as *path-following*, which is often based on a simplified *control-oriented* model. While this model allows for tractable controller design, it suffers from a structural mismatch with the full system and requires precise parameter knowledge – such as rotational *drag* coefficients – that is not available in practice.

This thesis proposes two adaptive control schemes to address these challenges. The first focuses on *body-shape control* when friction coefficients are partially known, completely unknown, or vary across the environment. The second addresses *path-following* with adaptation of the rotational *drag* coefficients of the simplified model. Combined, they form a unified adaptive approach to snake robot locomotion in practical applications.

The organization of this work is deliberately structured to mirror the hierarchical design process of the control architecture, starting from the inner building blocks and progressively adding more sophisticated layers in a cascaded structure. Each scheme is formally analyzed, and numerical simulation studies demonstrate the *tracking* performance and effective compensation despite parametric uncertainty.

Abstract in italiano

Gli *snake robot* rappresentano una classe avveniristica di sistemi articolati e *iperridondanti* che trae ispirazione diretta dalla loro controparte biologica. I numerosi gradi di libertà di cui dispongono li rendono complessi da controllare, ma al tempo stesso conferiscono loro capacità di locomozione uniche in ambienti irregolari e ostili.

La complessità intrinseca del modello dinamico completo limita tuttavia la progettazione di controllori particolarmente avanzati, specialmente per compiti come il *path-following*, i quali vengono spesso affrontati ricorrendo non al modello completo, bensì a un modello semplificato *control-oriented*. Sebbene quest'ultimo consenta un approccio *model-based*, soffre di una discrepanza strutturale col modello dinamico reale e richiede la conoscenza di alcuni parametri – come i coefficienti di *drag* rotazionale – non disponibili in applicazioni reali.

Questa tesi propone due schemi di controllo adattativo per superare tali limitazioni. Il primo si concentra sul *body-shape control*, considerando scenari in cui i coefficienti di attrito del terreno sono parzialmente noti, completamente ignoti o variabili. Il secondo affronta invece il problema del *path-following*, adattando i coefficienti di *drag* rotazionale del modello semplificato. Combinati, essi rappresentano un approccio adattativo unificato per la locomozione degli *snake robot* in applicazioni pratiche.

L'organizzazione di questo lavoro è volutamente strutturata in modo da rispecchiare il processo gerarchico di progettazione dell'architettura di controllo, partendo dai blocchi elementari interni e aggiungendo progressivamente livelli più sofisticati verso l'esterno, in una struttura a cascata. Ogni schema è analizzato formalmente, e simulazioni numeriche ne dimostrano le prestazioni sia in termini di *tracking* che di compensazione delle incertezze parametriche.

Contents

Figures

Tables

Code Listings

Acronyms

ARTDC Adaptive Robust Time Delayed Control. 10

AUV Autonomous Underwater Vehicle. 2

CFD Computational Fluid Dynamics. 102

CM Center of Mass. 20, 21, 23, 38, 39, 42, 43, 45, 50, 52–54, 58, 64, 66, 73, 75, 82, 84, 94, 97, 99

CO control-oriented. iii, v, 5–7, 9, 10, 51–54, 63–65, 73, 74, 79, 83–86, 93, 98, 101–103, 109

dof degrees of freedom. iii, 1, 23–26, 56

GAS Globally Asymptotically Stable. 14, 17

GUAS Globally Uniformly Asymptotically Stable. 16, 17, 68, 69

GUB Globally Uniformly Bounded. 15

GUUB Globally Uniformly Ultimately Bounded. 15, 89

ISS Input to State Stable. 33, 40, 77

LFC Lyapunov Function Candidate. 34, 38, 68, 70, 77, 86

LOS line-of-sight. 40, 42–44, 48–50, 58, 59, 61, 64, 66, 75, 82, 91, 93, 95, 96, 101

LTI Linear Time Invariant. 17

NTNU Norwegian University of Science and Technology. 102

PD Proportional-Derivative. 40, 44, 48, 49, 51, 57, 63, 64

PE Persistent Excitation. 14, 33, 35, 38, 70, 71, 73, 77, 78, 82, 97

RK4 Runge-Kutta 4. 29

RL Reinforcement Learning. 4

ROV Remotely Operated Vehicle. 1

SMC Sliding Mode Control. 10

STSMC Super Twisting Sliding Mode Control. 10

TDE Time-Delayed Estimation. 10

UB Uniformly Bounded. 15, 16

UGES Uniformly Globally Exponentially Stable. 17, 26, 35, 56, 67, 78

UUB Uniformly Ultimately Bounded. 15, 16

VHC Virtual Holonomic Constraints. 9, 10, 31

Glossary

actuated Directly influenced by one or more control inputs. 20, 23–26, 49, 53, 56

anisotropic Property of a material or system that varies with direction. 20, 53

body-shape control Refer to control of the posture of a robot's body, often to achieve desired locomotion or interaction with the environment.. iii, v, 3, 5, 6

control-oriented Designed to facilitate analysis and control design. 3, 5, 9, 49, 83

controllability Property that indicates whether exists an admissible control inputs to drive the system from any initial state to any desired final state. 20, 52, 53

cybernetic Pertaining to the science of automatic control systems in both machines and living organism. 1

drag Resistive force opposing the motion of an object, which can arise from friction, damping, or other effects.. iii, v, 5, 7, 9–11, 20, 51, 53, 55, 63, 65, 67–70, 73–75, 82, 93, 98, 101–103

eel-like Type of *lateral ondulation*, typically submarine, mimicking the swimming patterns of eels. 28

feed-forward control inputs based on knowledge of the system and desired trajectories, often complementing feedback control as anticipating tracking terms. 44, 65

forward velocity Component of a robot's velocity in the direction of its heading. 21, 41, 58, 60, 61, 74, 82, 97, 98

gait A coordinated pattern of movement or joint actuation to achieve locomotion. ix–xi, 4, 25, 28–30, 33, 35, 40, 45–47, 49, 52, 57, 58, 63, 65, 73, 77, 78, 84, 85, 93, 94, 96, 98, 99, 103

heading Orientation or direction in which a robot is facing. 3–7, 9, 10, 21, 24, 40–42, 44, 45, 48–50

holonomic If all of its motion constraints can be expressed as functions of position only, allowing it to move freely in as many independent directions as its degrees of freedom permit. In robotics, a holonomic robot can instantaneously move in any direction of its workspace.. 2, 5

hyper-redundant Possessing a high number of degrees of freedom–far exceeding what is necessary for basic task execution. iii, v, 1

lateral ondulation Locomotion method using continuous horizontal body waves to generate forward movement by pushing against the environment. xvii, 27–29

model-based Refers to a control or approach that relies on a mathematical model of the system to predict and analyze its behavior.. iii, v, 4, 5, 49, 51, 57, 59, 101, 102

path-following Control task where a robotic system is guided to move along a predefined geometric path in space, without necessarily imposing constraints on the timing or speed. iii, v, ix, 3–7, 9–11, 19, 28, 31, 40, 43, 49, 51, 57, 58, 63, 74, 93, 96, 101

planar Confined on a two-dimensional plane. 5, 9, 19, 101

snake robot A modular or articulated robot composed of serially connected segments, capable of snake-like locomotion.. iii, v

tracking Following a desired trajectory or reference signal over time, minimizing deviation from it.. iii, v, ix, 10, 17, 25, 26, 30, 33, 35, 38, 39, 44–46, 49, 50, 66–68, 70, 71, 73–75, 77, 78, 82, 84, 85, 88, 90, 93–97, 101

unactuated Lacking of direct actuation. It cannot be controlled independently and its motion results solely from passive dynamics or interaction with other actuated parts. 23, 24, 49

under-actuated Having fewer independent actuators than degrees of freedom. 23, 42

unmodeled Refers to dynamics, behaviors, or effects that are not captured in a system's mathematical model, often due to simplifications, assumptions, or unknown parameters. 40, 69, 88, 102, 103

Chapter 1

Introduction

1.1 Background

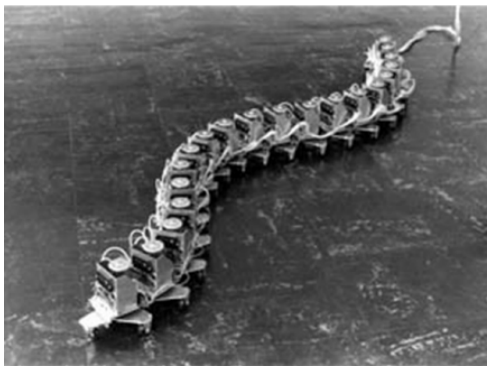
Biological snakes are fascinating and intricately complex creatures, endowed with unparalleled – sometimes unique locomotion capabilities. Among the many different species existing they are able to glide, swim with extraordinary grace, climb vertical surfaces, and, on occasions, appear to defy gravity by jumping and waving their tail to propel through the air [1]. Squeezing through tight, confined spaces, they traverse over irregular and rugged terrains, and maneuver with an agility and dexterity that seem almost magical. It is only natural for us to marvel and ask ourselves whether we, as humans, possess the ability to replicate the gift that *mother nature* has given to them.

A *cybernetic* counterpart of the biological snake would be incredibly revolutionary in life saving search-and-rescue operations [2] in cluttered and confined environments such as rock slides, caves, or post-earthquake collapsed buildings, where humans struggle to move and intrusive equipment could risk even a further collapse. Moreover, this innovative technology excels in underwater scenarios and challenging environments, where traditional *Remotely Operated Vehicle (ROV)s* prove to be too cumbersome or incapable of maneuvering. Beyond rescue operations, it also promises in sub-sea exploration, underwater pipeline maintenance, inspections and other demanding tasks [3].

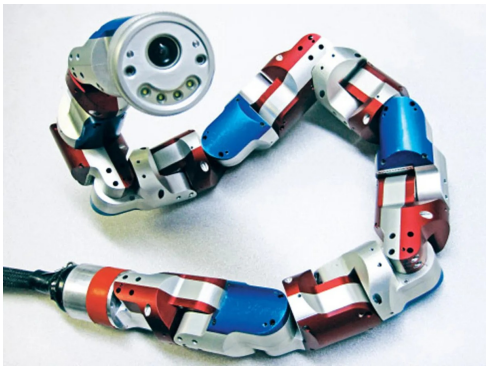
The snake robot's uniquely slim and reconfigurable modular design transforms it into a self-propelled *hyper-redundant* robotic arm, thanks to its many *degrees of freedom (dof)* [3]. Yet for all its promises, this remarkable potential is accompanied by significant challenges. In fact, it is precisely this high number of *degrees of freedom* that introduces substantial complexity into the system's design, interaction and control, making this class of swimming manipulators so interesting. As often observed, the greater the complexity of a challenge, the stronger the scientific interest it attracts.

Indeed, research in the field of snake robots date to the early 70s, with the efforts of Professor Hirose [4] being the earliest documented literature. From then

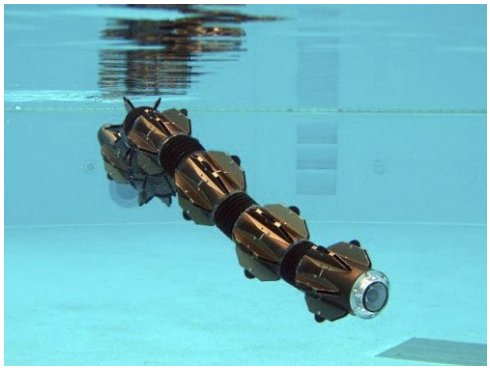
on, research communities around the world became increasingly drawn to the field, developing impressive and agile snake robots in an effort to mimic the motion capabilities of their biological counterparts [1]. Figure 1.1 shows some examples of snake robots developed for research purposes.



(a) *ACM III* from Prof. Hirose in 1972. Courtesy of Tokyo Institute of Technology.



(b) *Uncle Sam* can climb up poles. Courtesy of Carnegie Mellon University.



(c) *ACM R5* is equipped with passive wheels and can also swim underwater. Courtesy of Tokyo Institute of Technology.

(d) *Eelume* M-series modular *Autonomous Underwater Vehicle (AUV)* equipped with thrusters. Courtesy of *Eelume*.

Figure 1.1: Four examples of snake robots developed trough the years.

1.2 Control framework

Aside from their wide range of applications, snake robots also pose a rich and challenging problem from a control perspective. The complexity arises primarily from their high-dimensional, nonlinear dynamics and the *holonomic* constraints associated with their motion. To investigate these challenges, a detailed *revolute-joint* model, referred to also as *complex model*, has been developed. This model structurally and quantitatively captures the physical dynamics of the snake robot, and it's particularly valuable for simulation and for analyzing locomotion mechanisms from a first-principles perspective. For example, it becomes immediately

apparent from the models structure that propulsion arises from frictional interactions between each link of the robot and the surrounding environment, characterized by friction coefficients. These parameters quantify the resistance encountered during sliding contact, and are critical in determining locomotion's properties, or whether it's achievable at all. For certain values – such as in low-friction settings – the robot may lose its ability to generate propulsion altogether. While useful for understanding such structural features, the *revolute-joint* model proves to be difficult for controller design beyond joint-level control.

Yet, if these robots have to perform useful tasks beyond *body-shape* control, which governs the internal joint coordination responsible of generating locomotion, it is essential also to design higher-level control layers dedicated to typical robotics tasks such as path following, velocity and heading control. The ability to steer and orient appropriately is critical for navigating complex or constrained environments, while velocity control allows precise modulation of forward propulsion, enabling synchronization with external events (e.g., reacting to abrupt underwater currents) or coordination with other robotic systems. Together, these control layers allow snake robots to move not only effectively, but purposefully and autonomously – achieving goal-directed motion through cluttered or dynamic environments, which is fundamental for real-world tasks such as search and rescue, inspection, and environmental monitoring.

Figure 1.2: Autonomous locomotion and body-shape reconfiguration for underwater pipeline inspection (on the left) and self-coordination with other robots and the structures around (on the right). Courtesy of *Eelume*.

However, due to the intractability of full-order models like the *complex* one, high-level control designs often rely on a simplified *control-oriented* model, which retain only the essential dynamics of the system [1]. In these settings, control frameworks often adopt a cascaded structure, where the overall system is decomposed into hierarchical control layers – typically, an inner loop responsible for body-shape control, and an outer loop governing path-following and heading control. The layered architecture enables modular controller design and facilitates the stability analysis of the interconnected subsystems. As a result, the stability and performance of the overall locomotion system hinge on the proper coordination of these two layers, making the problems of body-shape and path-following control inherently interconnected.

Alongside these model-based approaches two paradigms have emerged and grew into their own mature fields of control engineering: adaptive control and, more recently, learning based methods. Adaptive control emerged not only to compensate for parametric uncertainty within system models, but also to address a critical gap in the literature at the time – namely, the lack of online adaptation mechanisms for environment-dependent parameters, which were traditionally handled using robust control. In parallel, learning-based methods have grown in prominence due to their ability to cope with complex and poorly modeled interactions between the robot and its environment. Techniques such as *Reinforcement Learning (RL)* and *Imitation Learning* have been employed to discover locomotion strategies and adapt *gaits* to diverse terrains without reliance on explicit models. However, despite their flexibility, learning-based methods lacks of formal stability guarantees, limiting their use in safety-critical or dangerous scenarios. As a result, they are frequently considered as supplementary to model-based methods, rather than as substitutes. These limitations have kept traditional model-based control predominant, as it offers robustness, formal stability guarantees, and interpretability.

1.3 Motivation

In this work, we focus on employing adaptive control techniques to address two distinct but related problems: (i) body-shape adaptive control for locomotion under unknown or varying friction coefficients, and (ii) adaptive heading control for use in a cascaded path-following system. Each of these is tackled through two conceptually different – but structurally related – modeling frameworks and control strategies.

The first problem stems from the observation that locomotion is critically dependent on the frictional properties of each link in contact with the environment. The friction coefficients, however, are rarely known precisely and often vary across space and time – especially in underwater or unstructured terrestrial settings. Friction coefficients are not only essential for accurate modeling and related control synthesis to achieve proper locomotion, but also for *informed decision-making* in uncertain conditions: when entering a region where the current *motion pattern* may cease to be effective due to unpredicted friction loss, timely identification of these parameters becomes critical for adjusting locomotion strategies, for instance by switching *gaits* or reconfiguring body posture, in order to maintain locomotion. As these parameters play a direct role in determining the effectiveness of locomotion, we adopt a classical adaptive control approach that compensates for such uncertainty online, enabling the robot to maintain effective propulsion even in dynamically changing environments.

The second problem, path-following, constitutes the outermost layer of the locomotion control hierarchy. Propulsion alone is not sufficient: the ability to follow a desired trajectory is what transforms undirected locomotion into goal-oriented

navigation, which is essential for accomplishing the complex tasks previously mentioned. A common challenge in the literature has been the difficulty of designing high-level controllers directly on the full *revolute-joint* model, due to its complexity and tightly coupled dynamic. To resolve this, we employ a *control-oriented* model, which abstracts the kinematics of the snake to a series of interconnected translational joints, while preserving the essential external dynamics. A key insight is that the structure of the *control-oriented (CO) model* is cast in a form that lends itself naturally to hierarchical decomposition – which enables us to design and implement cascaded control strategies. In this formulation, body-shape control forms the inner subsystem that perturbs and drives the outer, higher-level path-following layer. However, this abstraction comes at the cost of introducing *artificial* parameters – most notably, two rotational drag coefficients that allow for rotational dynamics to emerge in a model that only uses translational joints.

Thus, a key contribution of this work lies in developing adaptive strategies to estimate and regulate these artificial coefficients, so that the simplified *CO model* provides a good match to the real, physical system. This is crucial for the success of *CO-based* controller designs in practical implementations, especially when deploying controllers tuned in simulation onto physical robots in real-world scenarios. The problem of matching the *CO model* to the *complex model* has so far remained largely unsolved in the literature, as they were traditionally treated as control gains requiring tuning. Our adaptive approach offers a way to close this gap, by effectively applying adaptive control to artificial coefficients that exist only within the abstraction of the *CO model*.

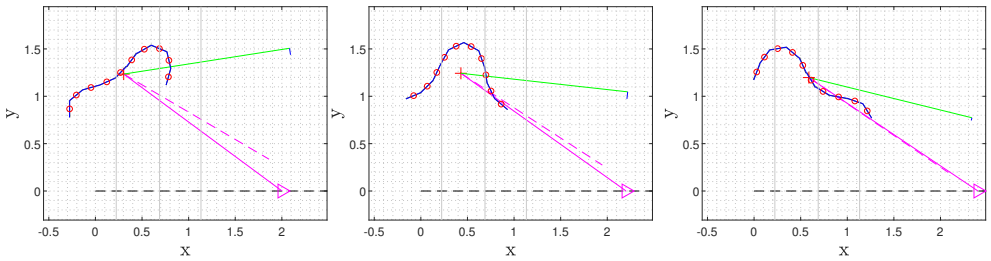
Through this two-pronged approach, we aim to combine the simplicity and suitability of the *CO model-based* heading controllers with the accuracy and applicability of the *complex model*-based body-shape controllers, with the goal of addressing the locomotion problem for snake robots in practical applications where perfect knowledge of model and environment-dependent parameters is not available.

1.4 Contributions and outline

In this work, we investigate adaptive control strategies for *planar holonomic* snake robots, addressing both body-shape control and path-following. The main contributions include adaptive extensions of existing controllers from the literature as well as the development of novel adaptive schemes, and can be summarized as follows:

1. We propose a friction-adaptive scheme for body-shape control, suitable for scenarios with varying or unknown friction coefficients. The design follows a methodology similar to Chitikena *et al.* [5], but builds upon the partial feedback linearization controller introduced by Liljebäck *et al.* [1] for the *complex model* of the snake robot. Stability is formally proven, and performance is validated through numerical simulations.

2. We propose a novel adaptive heading controller for path-following, designed on the simplified *CO-model* of the snake robot, within a cascaded control architecture. Stability is formally proven, and the controller is validated through numerical simulations.
3. We extend the adaptive heading controller by introducing a novel friction-adaptive scheme for body-shape control also for the *CO-model*, demonstrating that two adaptive control schemes can operate concurrently within the cascaded architecture. Stability is formally proven, and performance is confirmed through simulations.
4. We adjust the adaptive heading controller from point 2 – originally designed for the *CO-model* – to account for the structural differences of the *complex model* and integrate it with the friction-adaptive body-shape controller from point 1. This results in a complete adaptive locomotion control architecture for the *complex model* of the snake robot. Again, we prove stability and validate it through numerical simulations.
5. We provide a tool for analyzing the model-matching between the *complex model* and the *CO-model* of the snake robot, using the adaptive heading controller as a parameter estimation tool.
6. We develop a comprehensive `MATLAB` simulator for validation, analysis, and experimental evaluation of the proposed models and controllers. The simulator allows for versatile choice of control architecture and provides efficient visualization (see Figure 1.3).



(a) $t = 4$ [s] **(b)** $t = 5$ [s] **(c)** $t = 6$ [s]

Figure 1.3: Three snapshots from a simulation

The remainder of this thesis is organized as follows: Chapter 2 contextualizes this work in the existing research field, with a particular emphasis on existing adaptive control approaches to the problems herein faced. Chapter 3 presents the fundamental tools and theorems used for the stability analysis throughout this work.

Chapter 4 introduce the existing work in modeling and control of the *complex model* of the snake robot, to afterwards present an adaptive body shape controller suitable for scenarios characterized by unknown or variable friction coefficients within the environment. Subsequently, we move our attention to the path-following problem, highlighting the limitations associated with designing a

suitable heading controller for the *complex model*, given its inherent complexity.

Chapter 5 presents the simplified model, which is more suitable for control and path-following purposes. Here we introduce a heading controller proposed by [1] designed with a cascaded approach, from which we take inspiration to design an adaptive heading controller estimating the rotational drag coefficients. The subsequent sections propose a body-shape controller also for the *CO-model* demonstrating the parallel execution of two adaptive schemes.

Chapter 6 concludes by examining the challenges involved in transferring the adaptive heading controller to the *complex model*, and presents corresponding modifications and analysis to the control structure. The performance of the final control architecture is then validated on the *complex model*.

Finally, Chapter 7 concludes with a summary of the work conducted and potential directions for future research.

1.5 Methodology

To emphasize the practical relevance of the proposed methods, this thesis is complemented and enriched by carefully selected code snippets from the simulation environment, which was both developed and utilized throughout this work. As each component of the overall final control architecture is systematically introduced, the corresponding `MATLAB` examples are included to support theoretical development and provide reproducibility. These snippets not only illustrate the practical feasibility of the approach, but also serve as a foundation for future extensions and experimental deployment.

Chapter 2

Literature Review

Background. Research on snake robots has a long-standing history, beginning in the '40s with empirical and theoretical studies of snake locomotion by Gray [6], and continuing in 1972 with Professor Hiroses development of the first robotic implementation [4]. Over the past twenty years, the field has grown considerably, driven by the increasing interest in their potential applications, with numerous studies focusing on their modeling, design, and control. A comprehensive and detailed overview of the state of the art in bio-inspired snake robots is provided by Liljebäck *et al.* [7] and the references therein. Here, we narrow the focus to the field of *adaptive control*.

Cascaded control. A major challenge in the literature has been the difficulty of applying high-level controllers directly to the full *revolute-joint* model. This has motivated the introduction of a *control-oriented* model, which replaces the revolute joints with prismatic ones, considerably simplifying the structure of the dynamics while preserving the essential external behavior. The key advantage of this abstraction is that it naturally supports hierarchical decomposition, enabling *cascaded* control strategies – particularly for *path-following* – at the expense of introducing two *artificial* rotational drag parameters. In this direction, Liljebäck *et al.* [8] investigated *path-following* through cascaded stability arguments, though under the restrictive assumption that both the ground friction coefficients and the CO-model drag parameters are perfectly known.

Manouvering control. More recently, *maneuvering control* of planar snake robots has been investigated in Mohammadi *et al.* [9] within the framework of Virtual Holonomic Constraints (VHC), which implements a hierarchical control chain for velocity, heading, and body-shape. In this scheme, *dynamic compensators* are introduced to allow for control design on, respectively, velocity and head-angle in the *complex model* of the snake robot. While this approach offers powerful control capabilities, it critically relies on the assumption that the robots friction coefficients are perfectly known. In practice, friction is uncertain and environment-dependent, varying both across terrains for the same robot and across different robots on the same terrain. Consequently, control strategies that can cope with unknown or time-varying friction parameters are of greater practical importance.

SMC with uncertainties. Building on the hierarchical structure of the VHC-based maneuvering controller from [9], several robust strategies have been proposed to handle uncertainty in the friction coefficients without explicitly estimating them. For instance, the Sliding Mode Control (SMC) framework of [10] leverages the inherent robustness of SMC to achieve *finite-time stability* on head-angle and velocity tracking errors, provided that an upper bound on the uncertainties is known.

However, the discontinuous nature of SMC can induce chattering and large torques to remain on the sliding surface, potentially stressing mechanical components. To mitigate this, [11] proposed an *adaptive* switching gain that dynamically adjusts based on the deviation from the sliding surface, eliminating the need for prior knowledge on the uncertainty bounds and reducing the average control effort by approximately 33% [11].

On underwater snake robots. Other studies extend this line of work to underwater snake robots. Patel and Dwivedy [12] mitigated chattering effects by employing a Super Twisting Sliding Mode Control (STSMC) framework for an underwater snake robot model subject to linear and *nonlinear* drag forces, assuming negligible *added mass* effects. In their work the drag coefficients (equivalent to the ground friction coefficients) and fluidic torque parameters are explicitly estimated online, rather than just compensated. Zhang *et al.* [13] instead proposed a different control scheme: an $\mathcal{L}_1$ adaptive controller to explicitly address both matched and unmatched uncertainties in a reduced-order hydrodynamic model structurally similar to the ground-friction CO-model.

Time-Delayed Estimation (TDE). Alternative robust schemes have also been explored for ground friction models. For instance, in [14, 15] an Adaptive Robust Time Delayed Control (ARTDC) framework is applied to both the heading and body-shape layers of the hierarchical control architecture. This approach employs TDE to estimate and cancel model uncertainties at each timestep, under the assumption that these uncertainties remain approximately constant between successive timesteps.

Adaptive control. To move beyond robustness toward explicit parameter adaptation, Chitikena *et al.* [5] extended the VHC-based maneuvering controller of [9], providing an explicit regressor-based parametrization of the friction coefficients. This allowed the use of a classical adaptive control framework. To our knowledge, this represents the first major effort in which the term adaptive refers to the explicit online estimation of the physical friction coefficients for the *complex model* of the snake robot.

Adaptive control on the CO-model. On the other hand, Wang *et al.* [16] proposed a *backstepping*-based *path-following* controller for the CO-model, simultaneously adapting both the friction coefficients and the rotational drag parameters, but stopping short of extending the analysis to the complex model. While their method was validated on a physical snake robot, parameter drift is noticed in the estimated drag coefficients.

In this work, we aim to bridge these two lines of research. We design a control architecture that avoids the complexity of the hierarchical structure in [9], favoring the simplicity of the cascaded *path-following* architecture of [16], while embedding an adaptive body-shape controller designed for the *complex model* that makes use of the estimation structure of [5]. Unlike Wang *et al.* [16], our approach is afterwards extended to the *complex model* and showed to resolve the parameter drift noticed in the rotational drag coefficients. The result is a simple and reliable adaptive control framework suitable for underactuated, bio-inspired snake robots operating on uncertain ground conditions, where velocity control is of secondary importance.

Chapter 3

Stability Concepts

In this section, we summarize several fundamental results from nonlinear systems theory and Lyapunov stability analysis, which will be employed in the subsequent developments.

To assess the closed-loop stability of an equilibrium, typically the origin – whether in the sense of *Lyapunov stability* (trajectories starting sufficiently close remain close to the equilibrium) or *asymptotic stability* (Lyapunov stability holds and trajectories asymptotically *converge* to the equilibrium) – we will often employ Lyapunov's Direct Method.

Theorem 1 (Lyapunov's Direct Method, see *Theorem 4.1* from Khalil [17])**.** *Let $x = 0$ be an equilibrium point for $\dot{x} = f(x)$, where $f : D \subset \mathbb{R}^n \to \mathbb{R}^n$ is locally Lipschitz and D a domain containing $x = 0$. Let $V(x) : D \to \mathbb{R}$ be a continuously differentiable function such that*

$$V(0) = 0 \quad \text{and} \quad \mathrm{V(x) > 0} \quad \text{in} \quad \mathrm{D - \{0\}} \tag{3.1a}$$

$$\dot{V}(x) = \frac{\partial V}{\partial x} f(x) \leq 0 \quad \text{in} \quad \mathrm{D} \tag{3.1b}$$

*then the origin $x = 0$ is **Lyapunov stable**. Moreover, if*

$$\dot{V}(x) < 0 \quad \text{in} \quad \mathrm{D - \{0\}} \tag{3.2}$$

*then the origin $x = 0$ is **asymptotically stable**.*

Theorem 1 only guarantees *local* asymptotic stability. Theorem 2 extends this property to *global asymptotic stability*, ensuring convergence to the equilibrium from any initial state.

Theorem 2 (Barbashin-Krasovskij, see *Theorem 4.2* from Khalil [17])**.**
Let $x = 0$ be an equilibrium point for $\dot{x} = f(x)$, where $f : \mathbb{R}^n \to \mathbb{R}^n$ is locally Lipschitz. Let $V(x) : \mathbb{R}^n \to \mathbb{R}$ be a continuously differentiable function such that

$$V(0) = 0 \quad \text{and} \quad \mathrm{V(x)} > 0, \quad \forall \mathrm{x} \neq 0 \tag{3.3a}$$

$$\|x\| \to \infty \quad \Rightarrow \quad V(x) \to \infty \tag{3.3b}$$

$$\dot{V}(x) < 0, \quad \forall x \neq 0 \tag{3.3c}$$

*then $x = 0$ is **Globally Asymptotically Stable (GAS)**.*

When dealing with *adaptive control*, we're often only able to show negative *semi*-definiteness of $\dot{V}$, which by Theorem 1 only proves stability, not convergence. Theorem 3 is often used in control theory to demonstrate that certain desired states converge to an equilibrium point from the weaker condition of $\dot{V}(x) \leq 0$.

Theorem 3 (Barbalat's Lemma, see *Lemma 8.2* from Khalil [17])**.**
Let $\phi : \mathbb{R} \to \mathbb{R}$ be a uniformly continuous function on $[0, \infty)$. Suppose that $\lim_{t\to\infty} \int_0^t \phi(\tau)\, d\tau$ exists and is finite. Then,

$$\phi(t) \to 0 \quad as \quad t \to \infty \tag{3.4}$$

To prove convergence of the estimation error to 0, we need the regressor matrix to be sufficiently rich, as detailed in Theorem 4. This richness originates from the closed loop system trajectories, which themselves inherit richness from the reference trajectories.

Theorem 4 (Parameter convergence, see *Equation 8.31* from Slotine [18])**.**
Generalize the scalar adaptive law from Slotine and Li to the form

$$\dot{\hat{\mathbf{a}}} = -\Gamma Y^T \mathbf{x} \tag{3.5}$$

*where $\mathbf{x} \in \mathbb{R}^n$ is the error variable to be driven to $\mathbf{0}$, $\Gamma = \Gamma^T > 0$ is the **positive definite symmetric** matrix containing the adaptive gains, $\hat{\mathbf{a}} \in \mathbb{R}^m$ is the vector of parameter estimates and $Y \in \mathbb{R}^{m\times n}$ is some known non-linear function of the signals driving the adaptation.*

In order for the parameter estimates $\hat{\mathbf{a}}$ to converge to their ideal values $\mathbf{a}$, the regressor matrix Y must satisfy the Persistent Excitation (PE) condition; i.e,
$\forall t \geq 0$ there exist positive constants α_1 and T such that

$$\int_t^{t+T} Y(\tau)^T Y(\tau) d\tau \geq \alpha_1 \mathbf{I}_m \tag{3.6}$$

For linear systems, a sufficient condition for the convergent estimation of m unknown parameters is that the reference signals contains at least $m/2$ distinct

sinusoidal frequencies. However, for nonlinear systems, this simple relationship is generally not valid. The conditions for persistent excitation are far more complex and depend on the specific nonlinearities present in the system dynamics.

Sometimes it is difficult, or even not possible, to prove that $\dot{V} < 0$ over the entire domain; instead, negativity can only be ensured outside a neighborhood of the equilibrium point. Definition 1 introduces a slightly relaxed form of stability, wherein the state does not necessarily converge to the origin but is guaranteed to enter and remain within a small, acceptable bound. Theorem 5 provides Lyapunov-like conditions that formally establish this property.

Definition 1 (Uniformly Boundedness, see *Definition 4.6* from Khalil [17])**.** The solutions of $\dot{x} = f(x,t)$ are:

- **Uniformly Bounded (UB)** if there exists a positive constant c, independent of $t_0 \geq 0$, and for every $a \in (0,c)$, there is $\beta = \beta(a) > 0$, independent of t_0, such that

$$\|x(t_0)\| \leq a \implies \|x(t)\| \leq \beta, \quad \forall t \geq t_0 \tag{3.7}$$

- **Globally Uniformly Bounded (GUB)** if (3.7) holds for arbitrarily large a.
- **Uniformly Ultimately Bounded (UUB)** with ultimate bound b if there exist positive constants b and c, independent of $t_0 \geq 0$, and for every $a \in (0,c)$, there is $T = T(a,b) \geq 0$, independent of t_0, such that

$$\|x(t_0)\| \leq a \implies \|x(t)\| \leq b, \quad \forall t \geq t_0 + T \tag{3.8}$$

- **Globally Uniformly Ultimately Bounded (GUUB)** if (3.8) holds for arbitrarily large a.

Theorem 5 (UB by Lyapunov, see *Theorem 4.18* from Khalil [17]).
Let $D \subset \mathbb{R}^n$ be a domain that contains the origin and $V : [0, \infty) \times D \to \mathbb{R}$ be a continuously differentiable function such that

$$\alpha_1(\|x\|) \leq V(x,t) \leq \alpha_2(\|x\|) \tag{3.9a}$$

$$\frac{\partial V}{\partial t} + \frac{\partial V}{\partial x} f(x,t) \leq -W_3(x), \quad \forall \|x\| \geq \mu > 0 \tag{3.9b}$$

$\forall t \geq 0$ and $\forall x \in D$, where α_1 and α_2 are class $\mathcal{K}$ functions and $W_3(x)$ is a continuous positive definite function. Take $r > 0$ such that $B_r \subset D$ and suppose that

$$\mu < \alpha_2^{-1}(\alpha_1(r)) \tag{3.10}$$

Then, there exists a class $\mathcal{KL}$ function β and for every initial state $x(t_0)$, satisfying $\|x(t_0)\| \leq \alpha_2^{-1}(\alpha_1(r))$, there is $T \geq 0$ (dependent on $x(t_0)$ and μ) such that the solution of $\dot{x} = f(x,t)$ satisfies

$$\|x(t)\| \leq \beta(\|x(t_0)\|, t - t_0), \quad \forall t_0 \leq t \leq t_0 + T \tag{3.11a}$$

$$\|x(t)\| \leq \alpha_1^{-1}(\alpha_2(\mu)), \quad \forall t \geq t_0 + T \tag{3.11b}$$

Moreover, if $D = \mathbb{R}^n$ and α_1 belongs to class $\mathcal{K}_\infty$, then (3.11a) *and* (3.11b) *hold for any initial state $x(t_0)$, with no restriction on how large μ is.*

Inequalities (3.11a) and (3.11b) show that $x(t)$ is *UB* for all $t \geq t_0$ and *UUB* with the ultimate bound $\alpha_1^{-1}(\alpha_2(\mu))$. The ultimate bound is a class $\mathcal{K}$ function of μ; hence, the smaller the value of μ, the smaller the ultimate bound. As $\mu \to 0$, the ultimate bound approaches zero.

Theorem 6 (GUAS, see *Theorem 4.8 and 4-9* from Khalil [17]).
Let $x = 0$ be an equilibrium point for $\dot{x} = f(x,t)$, where $f : [0, \infty) \times \mathbb{R}^n \to \mathbb{R}^n$ is piecewise continuous in t and locally Lipschitz in x on $[0, \infty) \times \mathbb{R}^n$. Let $V : [0, \infty) \times \mathbb{R}^n \to \mathbb{R}$ be a continuously differentiable function such that

$$W_1(x) \leq V(x,t) \leq W_2(x) \tag{3.12a}$$

$$\dot{V}(x,t) = \frac{\partial V}{\partial t} + \frac{\partial V}{\partial x} f(x,t) \leq -W_3(x) \tag{3.12b}$$

*$\forall t \geq 0$ and $\forall x \in \mathbb{R}^n$, where $W_1(x), W_2(x)$ and $W_3(x)$ are continuous **positive definite** functions on $\mathbb{R}^n$ and W_1 is **radially unbounded** on $\mathbb{R}^n$. Then, the origin $x = 0$ is **Globally Uniformly Asymptotically Stable (GUAS)**. Moreover, every trajectory satisfies*

$$\|x(t)\| \leq \beta(\|x(t_0)\|, t - t_0), \quad \forall t \geq t_0 \geq 0 \tag{3.13}$$

for some class $\mathcal{KL}$ function β.

While Theorem 2 proves *GAS* ensuring that all trajectories ultimately converge to the equilibrium, Theorem 6 strengthens this result by ensuring *uniform* convergence, guaranteeing a $\mathcal{KL}$ bound on the convergence of the error. This property is particularly valuable in tracking scenarios, as opposed to pure regulation problems.

Proposition 1 instead extends the notion of *GUAS* for Linear Time Invariant (LTI) second order systems by establishing a much stronger property where convergence to the equilibrium occurs not only uniformly but also at an exponential rate.

Proposition 1 (UGES)**.**
Consider the second-order system

$$\ddot{\mathbf{x}} + k_d\dot{\mathbf{x}} + k_p\mathbf{x} = 0, \tag{3.14}$$

with $x \in \mathbb{R}^n$ and $k_d, k_p > 0$ scalar positive values. Hence, the origin of the system is **Uniformly Globally Exponentially Stable (UGES)**.

Proof. We can rewrite (3.14) in a state-space form as

$$\underbrace{\begin{bmatrix}\ddot{\mathbf{x}}\\ \dot{\mathbf{x}}\end{bmatrix}}_{\dot{\mathbf{z}}} = \underbrace{\begin{bmatrix}-k_d I_{N-1} & -k_p I_{N-1}\\ I_{N-1} & \mathbf{0}\end{bmatrix}}_{A}\underbrace{\begin{bmatrix}\dot{\mathbf{x}}\\ \mathbf{x}\end{bmatrix}}_{\mathbf{z}} \tag{3.15a}$$

$$\dot{\mathbf{z}} = A\mathbf{z}. \tag{3.15b}$$

where $\mathbf{z} = [\dot{\mathbf{x}}, \mathbf{x}]^T$. The *characteristic polynomial* of A is $\Delta = (\lambda^2 + k_d\lambda + k_p)^{N-1}$, hence the *eigenvalues* of A are the two roots of the scalar quadratic equation

$$\lambda^2 + k_d\lambda + k_p = 0, \tag{3.16}$$

each one with *algebraic multiplicity* $N-1$. Since $k_d, k_p > 0$ (3.16) has two roots with strictly negative real part, hence $A < 0$ is *Hurwitz*. Since (3.15b) is also *LTI* this implies there exist $M, \alpha > 0$ such that $\|\mathbf{z}(t)\| \leq Me^{-\alpha(t-t_0)}\|\mathbf{z}(t_0)\| \;\; \forall t \geq t_0$, and since this is valid for every initial $\mathbf{z}(t_0)$ the origin is *UGES*. □

Theorem 7 (Lyapunov Equation, see *Theorem 4.6* from Khalil [17])**.**
*A matrix $A \in \mathbb{R}^{n\times n}$ is **Hurwitz**, i.e. $\Re e(\lambda_i) < 0$ for all eigenvalues of A, if and only if for any given **positive definite** matrix $Q \in \mathbb{R}^{n\times n}$ there exists a **positive definite symmetric** matrix $P = P^T \in \mathbb{R}^{n\times n}$ that satisfies the **Lyapunov equation***

$$A^T P + PA = -Q. \tag{3.17}$$

*Moreover, if A is **Hurwitz**, then P is unique solution of* (3.17).

Chapter 4

A Complex Model for the Snake Robot

In this chapter we present the accurate mathematical model for a planar snake robot as derived in Liljebäck *et al.* [1], for which we're able to design an exponentially stable joint controller by partial feedback linearization of the model under complete knowledge of the model parameters *and terrain parameters*. The model considers the robot as a collection of rigid links, connected together by motorized revolute joints, which we can apply control torques to. We then present a methodology for transitioning towards an adaptive body-shape controller to address partial or complete lack of knowledge regarding the surrounding environment, specifically ground friction coefficients, inspired by the approach proposed in [5]. Lastly, we tackle the problem of path-following control, showing the limitations of the control purposes of this *complex* model, which motivates the need of a simpler *control-oriented* model, which is then introduced in the next chapter.

4.1 Modeling

This section provides a brief overview of the dynamical model considered in this work, which we refer to as the *complex model*. Among the various alternatives explored in the literature (see, e.g., [1, 3]), we model the ground friction forces as viscous. While this choice is less accurate from a physical perspective – since ground friction is more accurately captured by a Coulomb-type model – it offers important advantages. First, viscous friction leads to mathematically simpler expressions, which facilitates both simulation and controller design [1, Section 2.5]. Second, it is structurally closer to the hydrodynamic resistive forces encountered in underwater locomotion, making it more general and suitable for our case. Moreover, mobile robots, especially snake-like ones, frequently deviate from conventional kinematic models due to side-slip constraints, and their considerable surface contact area with the ground leads to significant velocity-dependent interactions, further supporting the viscous approximation [5].

We also assume that the friction forces are *anisotropic*, a necessary, but not sufficient, condition for *controllability*, and more specifically that the friction coefficient describing the force in the tangential direction of the link, which we'll denote as c_t, is smaller than the one describing the force in the direction normal to the link, denoted as c_n. This is also necessary to achieve forward propulsion, as motivated by [1, Chapter 4], and it's a common characteristic observed in biological snakes enabled by the scales along their bodies. Artificially, this can be effectively replicated through the attachment of passive wheels along the body of the snake robot, by artificial scales or by employing a variety of alternative methods. The same is also valid in underwater scenarios where the robot can have a completely smooth outer surface, and it will still have this anisotropic drag property due to his prolonged shape which produces higher drag forces in the normal direction of each link compared to the tangential link direction [19].

The material presented in the following sections is an exposition of snake robot modeling results from the literature, and is sufficient background for the reader to follow the rest of the thesis. Detailed derivations are, however, omitted as they are not original contributions of the thesis. For additional details regarding the derivation of the model, we refer the reader to [1].

4.1.1 Model Parameters and Kinematic

The *complex model* of the snake robot consists of N rigid links, each one of length $2l$ interconnected by $N-1$ rotational actuated joints. All links has the same mass m and moment of inertia $J = \frac{1}{3}ml^2$. We assume that the mass and the inertia of the actuators is negligible with respect to the one of the links, and we also don't consider the width of the links. We also assume that the mass is uniformly distributed across the links, such that his Center of Mass (CM) coincides with his geometrical center.

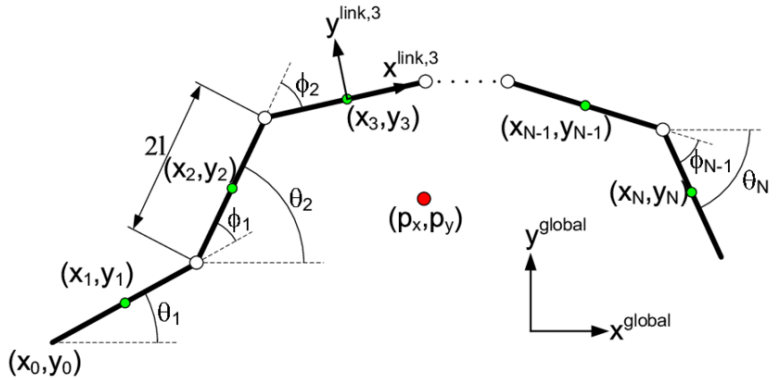


Figure 4.1: The kinematic parameters of the snake robot. Illustration from [1].

We call θ_i the N absolute angles that the links $i \in [1,\ldots,N]$ forms with the global x-axis and we define ϕ_i the $N-1$ relative angles between adjacent links

(i.e. the angles of the joints) as

$$\phi_i = \theta_i - \theta_{i+1}. \tag{4.1}$$

The CM of the link i is located in (x_i, y_i), while the overall center of mass of the snake robot is located in (p_x, p_y), both expressed in global coordinates. For a more compact notation, we will frequently assemble the angles in a single vector defining $\boldsymbol{\theta} = [\theta_1, \ldots, \theta_N] \in \mathbb{R}^N$ and $\boldsymbol{\phi} = [\phi_1, \ldots, \phi_{N-1}] \in \mathbb{R}^{N-1}$ the vector of the absolute link angles and relative joint angles, respectively. We also define as *heading* $\overline{\theta}$ the average of the link angles θ_i, as

$$\overline{\theta} = \frac{1}{N} \sum_{i=1}^{N} \theta_i, \tag{4.2}$$

and as forward velocity $\overline{v}_t$ and normal velocity $\overline{v}_n$, respectively, the component of the CM velocity $\dot{\boldsymbol{p}}$ along the current direction of heading $\overline{\theta}$ and its normal direction, as

$$\overline{v}_t = \dot{p}_x \cos\overline{\theta} + \dot{p}_y \sin\overline{\theta}, \tag{4.3a}$$

$$\overline{v}_n = -\dot{p}_x \sin\overline{\theta} + \dot{p}_y \cos\overline{\theta}. \tag{4.3b}$$

4.1.2 Notation

Here we present some matrices and vectors that will be frequently used throughout this work.

$$\mathbf{A} = \begin{bmatrix} 1 & 1 & & \\ & \ddots & \ddots & \\ & & 1 & 1 \end{bmatrix} \in \mathbb{R}^{(N-1)\times N}, \quad \mathbf{D} = \begin{bmatrix} 1 & -1 & & \\ & \ddots & \ddots & \\ & & 1 & -1 \end{bmatrix} \in \mathbb{R}^{(N-1)\times N},$$

$$\mathbf{e} = [1, \ldots, 1] \in \mathbb{R}^N, \qquad \mathbf{E} = \begin{bmatrix} \mathbf{e} & \mathbf{0}_{N\times 1} \\ \mathbf{0}_{N\times 1} & \mathbf{e} \end{bmatrix} \in \mathbb{R}^{2N\times 2},$$

$$\mathbf{H} = \begin{bmatrix} 1 & 1 & \ldots & 1 \\ & 1 & \ldots & 1 \\ & & \ddots & \vdots \\ & & & 1 \end{bmatrix} \in \mathbb{R}^{N\times N}, \tag{4.4}$$

where $\mathbf{A}$ and $\mathbf{D}$ represents, respectively, an addition and difference matrix, used frequently to add and subtract adjacent elements of a vector, $\mathbf{H}$ reflects a cumulative addition, while $\mathbf{e}$ a summation vector, used to sum-up all the elements of a vector.

In addition, we also define $\sin\boldsymbol{\theta} = [\sin(\theta_1), \ldots, \sin(\theta_N)]^T \in \mathbb{R}^N$ the vector that performs element-wise the operation $\sin(\cdot)$ on the elements of $\boldsymbol{\theta}$, and $\cos\boldsymbol{\theta} = [\cos(\theta_1), \ldots, \cos(\theta_N)]^T \in \mathbb{R}^N$ and $\dot{\boldsymbol{\theta}}^2 = [\dot{\theta}_1^2, \ldots, \dot{\theta}_N^2]^T \in \mathbb{R}^N$ in the same way.

4.1.3 Dynamical Model

The model derivations are omitted here for brevity, and we refer the curious reader to Liljebäck *et al.* [1, Section 2.7] for a detailed derivation of the model. Following [1, Chapter 2] the complete dynamical model of the snake robots can be written as

$$\boxed{\begin{aligned}\mathbf{M}_\theta\ddot{\boldsymbol{\theta}} + \mathbf{W}\dot{\boldsymbol{\theta}}^2 - l\mathbf{S}_\theta\mathbf{K}\mathbf{f}_{R,x} + l\mathbf{C}_\theta\mathbf{K}\mathbf{f}_{R,y} &= \mathbf{D}^T\mathbf{u}, && \text{(4.5a)}\\ Nm\ddot{\boldsymbol{p}} = Nm\begin{bmatrix}\ddot{p}_x\\ \ddot{p}_y\end{bmatrix} = \begin{bmatrix}\mathbf{e}^T\mathbf{f}_{R,x}\\ \mathbf{e}^T\mathbf{f}_{R,y}\end{bmatrix} &= \mathbf{E}^T\mathbf{f}_R, && \text{(4.5b)}\end{aligned}}$$

where $\mathbf{u}$ is the vector of the actuation torques, $\mathbf{f}_R$ is the vector of the viscous friction forces acting as the coupling term between the two equations, which is the hidden underlying mechanism that allows forward motion of the snake, and it's given by

$$\mathbf{f}_R = \begin{bmatrix}\mathbf{f}_{R,x}\\ \mathbf{f}_{R,y}\end{bmatrix} = -\begin{bmatrix}c_t(\mathbf{C}_\theta)^2 + c_n(\mathbf{S}_\theta)^2 & (c_t - c_n)\mathbf{S}_\theta\mathbf{C}_\theta\\ (c_t - c_n)\mathbf{S}_\theta\mathbf{C}_\theta & c_t(\mathbf{S}_\theta)^2 + c_n(\mathbf{C}_\theta)^2\end{bmatrix}\begin{bmatrix}\dot{\mathbf{X}}\\ \dot{\mathbf{Y}}\end{bmatrix} \in \mathbb{R}^{2N}, \tag{4.6}$$

for which the forward kinematic problem of $\dot{\mathbf{X}}$ and $\dot{\mathbf{Y}}$ can be solved as

$$\dot{\mathbf{X}} = l\mathbf{K}^T\mathbf{S}_\theta\dot{\boldsymbol{\theta}} + \mathbf{e}\dot{p}_x, \tag{4.7a}$$
$$\dot{\mathbf{Y}} = -l\mathbf{K}^T\mathbf{C}_\theta\dot{\boldsymbol{\theta}} + \mathbf{e}\dot{p}_y, \tag{4.7b}$$

and the remaining quantities are defined as

$$\mathbf{S}_\theta = \operatorname{diag}(\sin\boldsymbol{\theta}) \in \mathbb{R}^{N\times N}, \tag{4.8a}$$
$$\mathbf{C}_\theta = \operatorname{diag}(\cos\boldsymbol{\theta}) \in \mathbb{R}^{N\times N}, \tag{4.8b}$$
$$\mathbf{M}_\theta = J\mathbf{I}_N + ml^2\mathbf{S}_\theta\mathbf{V}\mathbf{S}_\theta + ml^2\mathbf{C}_\theta\mathbf{V}\mathbf{C}_\theta, \tag{4.8c}$$
$$\mathbf{W} = ml^2\mathbf{S}_\theta\mathbf{V}\mathbf{C}_\theta - ml^2\mathbf{C}_\theta\mathbf{V}\mathbf{S}_\theta, \tag{4.8d}$$
$$\mathbf{V} = \mathbf{A}^T(\mathbf{D}\mathbf{D}^T)^{-1}\mathbf{A}, \tag{4.8e}$$
$$\mathbf{K} = \mathbf{A}^T(\mathbf{D}\mathbf{D}^T)^{-1}\mathbf{D}. \tag{4.8f}$$

By defining the state variable $\mathbf{x} = [\boldsymbol{\theta}^T, \boldsymbol{p}^T, \dot{\boldsymbol{\theta}}^T, \dot{\boldsymbol{p}}^T]^T \in \mathbb{R}^{2N+4}$ and solving (4.5) for $\ddot{\boldsymbol{\theta}}$ and $\ddot{\boldsymbol{p}}$ we can find the integration law to be used in the `MATLAB` simulator, whose implementation is shown in Code listing 4.1.

Code listing 4.1: Dynamics of the complex model of the snake robot: `dyn233`

```
%% dynamics using (2.33) model

function dx = dyn233(x,u)
    global Nl l m J ct cn;
    global nx e D K V;

    theta = x(1:Nl); thetadot = x(Nl+3:end-2);
```

```
    pxdot = x(end-1); pydot = x(end);

    %(2.12)
    St = diag(sin(theta));
    Ct = diag(cos(theta));
    Xd = l*K'*St*thetadot + e*pxdot;
    Yd = -l*K'*Ct*thetadot + e*pydot;

    %(2.34)
    Mt = J*eye(Nl) + m*(l^2)*St*V*St + m*(l^2)*Ct*V*Ct;
    Mtinv = inv(Mt);
    Wt = m*(l^2)*St*V*Ct - m*(l^2)*Ct*V*St;

    %(2.25)
    Fx = -((ct*(Ct.^2) + cn*(St.^2))*Xd + (ct-cn)*(St*Ct)*Yd);
    Fy = -((ct-cn)*St*Ct*Xd + (ct*(St.^2) + cn*(Ct.^2))*Yd);
    Fr = [Fx;Fy];

    %(2.33)
    dx = zeros(nx,1);
    dx(1:Nl) = thetadot;
    dx(Nl+1:Nl+2) = [pxdot;pydot];
    dx(Nl+3:end-2) = Mtinv*(-Wt*(thetadot.^2) + l*St*K*Fx - l*Ct*K*Fy + D'*u);
    dx(end-1) = e' * Fx / (Nl*m);
    dx(end) = e' * Fy / (Nl*m);
end
```

4.1.4 Separating Actuated and Unactuated Dynamics

The system itself has $N+2$ configurations but only $N-1$ actuators, resulting in a *under-actuated* system for which the controls $\mathbf{u}$ has no direct effect on the CM dynamics, leaving 3 *unactuated* degrees of freedom (position and orientation). We therefore seek a transformation that allows us to write the model in a form that separates the *actuated* and *unactuated degrees of freedom* of the model, to facilitate the design of a controller, exploiting a method called *partial feedback linearisation* (see Spong [20] and Gu and Xu [21]) that consists of linearising the dynamics of the *actuated* part.

Again, the model derivations are omitted here for brevity, and we refer the curious reader to Liljebäck *et al.* [1, Section 2.7] for a detailed derivation of the partitioned model. The complete model of the snake robot is

$$\overline{\mathbf{M}}(\overline{\boldsymbol{\phi}})\ddot{\mathbf{q}}_{\boldsymbol{\phi}} + \overline{\mathbf{W}}(\overline{\boldsymbol{\phi}}, \dot{\overline{\boldsymbol{\phi}}}) + \overline{\mathbf{G}}(\overline{\boldsymbol{\phi}})\mathbf{f}_R(\overline{\boldsymbol{\phi}}, \dot{\overline{\boldsymbol{\phi}}}, \dot{\mathbf{p}}) = \overline{\mathbf{B}}\mathbf{u}, \tag{4.9}$$

where

$$\mathbf{q}_{\boldsymbol{\phi}} = \begin{bmatrix} \overline{\boldsymbol{\phi}} \\ \mathbf{p} \end{bmatrix}, \qquad \overline{\boldsymbol{\phi}} = [\phi_1, \ldots, \phi_{N-1}, \theta_N]^T \in \mathbb{R}^N, \tag{4.10}$$

and

$$\overline{\mathbf{M}}(\overline{\boldsymbol{\phi}}) = \begin{bmatrix} \mathbf{H}^T \mathbf{M}_{\boldsymbol{\theta}} \mathbf{H} & \mathbf{0}_{N\times 2} \\ \mathbf{0}_{2\times N} & Nm\mathbf{I}_2 \end{bmatrix}, \tag{4.11a}$$

$$\overline{\mathbf{W}}(\overline{\boldsymbol{\phi}}, \dot{\overline{\boldsymbol{\phi}}}) = \begin{bmatrix} \mathbf{H}^T \mathbf{W}\ \mathrm{diag}(\mathbf{H}\dot{\overline{\boldsymbol{\phi}}})\mathbf{H}\dot{\overline{\boldsymbol{\phi}}} \\ \mathbf{0}_{2\times 1} \end{bmatrix}, \tag{4.11b}$$

$$\overline{\mathbf{G}}(\overline{\boldsymbol{\phi}}) = \begin{bmatrix} -l\mathbf{H}^T \mathbf{S}_{\boldsymbol{\theta}} \mathbf{K} & l\mathbf{H}^T \mathbf{C}_{\boldsymbol{\theta}} \mathbf{K} \\ -\mathbf{e}^T & \mathbf{0}_{1\times N} \\ \mathbf{0}_{1\times N} & -\mathbf{e}^T \end{bmatrix}, \tag{4.11c}$$

$$\overline{\mathbf{B}} = \begin{bmatrix} \mathbf{I}_{N-1} \\ \mathbf{0}_{3\times(N-1)} \end{bmatrix}. \tag{4.11d}$$

The first $N-1$ equations of (4.9) reflects the dynamics of the *actuated* degrees of freedom, while the last three represents the *unactuated* ones, i.e. position in the global coordinates and orientation of the head. Note that *orientation of the head* θ_N is different from the previously defined *heading* $\overline{\theta}$. θ_N is in fact the last link we informally call head link. We can then partition the model as

$$\boxed{\overline{\mathbf{M}}_{11}\ddot{\mathbf{q}}_a + \overline{\mathbf{M}}_{12}\ddot{\mathbf{q}}_u + \overline{\mathbf{W}}_1 + \overline{\mathbf{G}}_1 \mathbf{f}_R = \mathbf{u},} \tag{4.12a}$$

$$\boxed{\overline{\mathbf{M}}_{21}\ddot{\mathbf{q}}_a + \overline{\mathbf{M}}_{22}\ddot{\mathbf{q}}_u + \overline{\mathbf{W}}_2 + \overline{\mathbf{G}}_2 \mathbf{f}_R = \mathbf{0}_{3\times 1},} \tag{4.12b}$$

where $\mathbf{q}_a = [\phi_1, \ldots, \phi_{N-1}]^T \in \mathbb{R}^{N-1}$ represents the $N-1$ *actuated degrees of freedom* (i.e. the dynamics of the relative joint angles), while $\mathbf{q}_u = [\theta_N, p_x, p_y]^T \in \mathbb{R}^3$ represents the 3 *unactuated degrees of freedom* (i.e. absolute orientation and position). Moreover $\overline{\mathbf{M}}_{11} \in \mathbb{R}^{(N-1)\times(N-1)}$, $\overline{\mathbf{M}}_{12} \in \mathbb{R}^{(N-1)\times 3}$, $\overline{\mathbf{M}}_{21} \in \mathbb{R}^{3\times(N-1)}$, $\overline{\mathbf{M}}_{22} \in \mathbb{R}^{3\times 3}$, $\overline{\mathbf{W}}_1 \in \mathbb{R}^{N-1}$, $\overline{\mathbf{W}}_2 \in \mathbb{R}^3$, $\overline{\mathbf{G}}_1 \in \mathbb{R}^{(N-1)\times 2N}$, and $\overline{\mathbf{G}}_2 \in \mathbb{R}^{3\times 2N}$.

From the partitioned model in Equations (4.12a) and (4.12b), following the approach presented in Reyhanoglu *et al.* [22] we can begin by solving (4.12b) for $\ddot{\mathbf{q}}_u$, as

$$\ddot{\mathbf{q}}_u = -\overline{\mathbf{M}}_{22}^{-1}\left(\overline{\mathbf{M}}_{21}\ddot{\mathbf{q}}_a + \overline{\mathbf{W}}_2 + \overline{\mathbf{G}}_2 \mathbf{f}_R\right), \tag{4.13}$$

which inserted in (4.12a) gives

$$\ddot{\mathbf{q}}_a = \left(\overline{\mathbf{M}}_{11} - \overline{\mathbf{M}}_{12}\overline{\mathbf{M}}_{22}^{-1}\overline{\mathbf{M}}_{21}\right)^{-1}\left[\overline{\mathbf{M}}_{12}\overline{\mathbf{M}}_{22}^{-1}\overline{\mathbf{W}}_2 - \overline{\mathbf{W}}_1 + \left(\overline{\mathbf{M}}_{12}\overline{\mathbf{M}}_{22}^{-1}\overline{\mathbf{G}}_2 - \overline{\mathbf{G}}_1\right)\mathbf{f}_R + \mathbf{u}\right]. \tag{4.14}$$

Equations (4.13) and (4.14), along with Equations (4.6) and (4.7), are the ones used in the simulator to solve for the dynamics when choosing `dyn241` instead of `dyn233`. Code listing 4.2 shows a basic implementation, where it's been used the identity $\boldsymbol{\theta} = \mathbf{H}\overline{\boldsymbol{\phi}} = \mathbf{H} \cdot [\mathbf{q}_a^T, \theta_N]^T$ to keep consistency with the previous representation of the snake robot state as $\mathbf{x} = [\boldsymbol{\theta}^T, \boldsymbol{p}^T, \dot{\boldsymbol{\theta}}^T, \dot{\boldsymbol{p}}^T]^T$.

Code listing 4.2: Dynamics of the complex model of the snake robot: `dyn241`

```
%% dynamics using (2.41) partitioned model

function dx = dyn241(x,u)
    global Nl l m J ct cn;
    global nx e K V H;

    % ...
    % from theta to Fr of {lst:dyn233} : lines [6--23]
    % ...

    %(2.40)
    M_bar = [H'*Mt*H, zeros(Nl,2);
             zeros(2,Nl), Nl*m*eye(2)];
    W_bar = [H'*Wt*(thetadot.^2);
             zeros(2,1)];
    G_bar = [-l*H'*St*K, l*H'*Ct*K;
             -e', zeros(1,Nl);
             zeros(1,Nl), -e'];

    M11 = M_bar(1:Nl-1, 1:Nl-1);
    M12 = M_bar(1:Nl-1, Nl:end);
    M21 = M_bar(Nl:end, 1:Nl-1);
    M22 = M_bar(Nl:end, Nl:end);
    M22inv = inv(M22);
    W1 = W_bar(1:Nl-1);
    W2 = W_bar(Nl:end);
    G1 = G_bar(1:Nl-1,:);
    G2 = G_bar(Nl:end,:);

    %(2.41-2.42)
    dx = zeros(nx,1);
    dx(1:Nl) = thetadot;
    dx(Nl+1:Nl+2) = [pxdot; pydot];
    qaddot = inv(M11 - M12*M22inv*M21) * ...
             (M12*M22inv*W2 - W1 + (M12*M22inv*G2 - G1)*Fr + u);
    quddot = -M22inv*(M21*qaddot + W2 + G2*Fr);
    dx(Nl+3:end-2) = H*[qaddot; quddot(1)];
    dx(end-1) = quddot(2);
    dx(end) = quddot(3);
end
```

4.2 Body Shape Control

We now proceed to introduce the building blocks of the body-shape controller, designed to achieve tracking of the $N-1$ *actuated degrees of freedom* ϕ_i. The controller consists of two main components: a *gait pattern* generator, which produces the reference trajectories, and a state-feedback controller, which requires knowledge of the robots state variables $\boldsymbol{\theta}, \dot{\boldsymbol{\theta}}, \dot{p}_x, \dot{p}_y$ as well as the environment parameters (c_t, c_n). The latter will be later made adaptive with respect to the friction coefficients, as detailed in Section 4.3. The overall control architecture is illustrated in Figure 4.2.

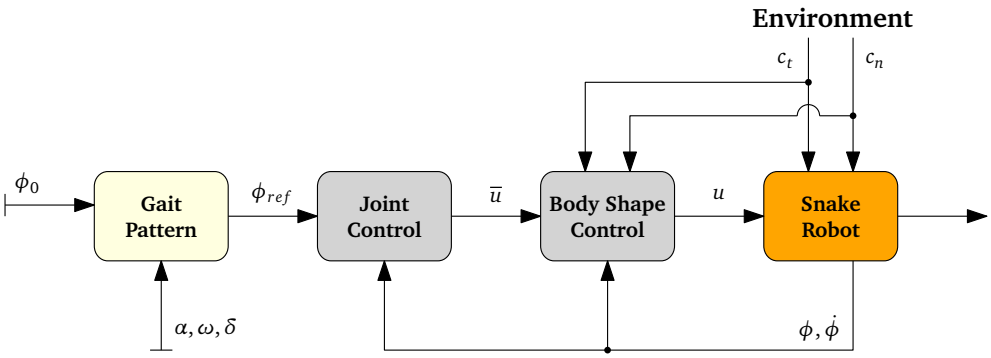


Figure 4.2: Partial feedback linearization body-shape controller

4.2.1 Partial Feedback Linearization of the Model

Based on the partitioned model in Equations (4.13) and (4.14) it's possible to linearize the *actuated degrees of freedom* (see Spong [20] and Gu and Xu [21]) using the following control law:

$$\mathbf{u} = \underbrace{\left(\overline{\mathbf{M}}_{11} - \overline{\mathbf{M}}_{12}\overline{\mathbf{M}}_{22}^{-1}\overline{\mathbf{M}}_{21}\right)}_{\text{linearizing term } \mathbf{M_u}} \overline{\mathbf{u}} + \underbrace{\overline{\mathbf{W}}_1 - \overline{\mathbf{M}}_{12}\overline{\mathbf{M}}_{22}^{-1}\overline{\mathbf{W}}_2}_{\text{velocity term } \mathbf{W_u}} + \underbrace{\left(\overline{\mathbf{G}}_1 - \overline{\mathbf{M}}_{12}\overline{\mathbf{M}}_{22}^{-1}\overline{\mathbf{G}}_2\right)}_{\text{friction term } \mathbf{G_u}} \mathbf{f}_R \quad (4.15)$$

where $\overline{\boldsymbol{u}}$ is a new set of control inputs defined as

$$\overline{\mathbf{u}} = \ddot{\boldsymbol{\phi}}_{ref} - k_d(\dot{\boldsymbol{\phi}} - \dot{\boldsymbol{\phi}}_{ref}) - k_p(\boldsymbol{\phi} - \boldsymbol{\phi}_{ref}), \quad (4.16)$$

with $k_p > 0, k_d > 0$ scalar controller gains.

Proposition 2. The controller in (4.15) linearize the dynamics of the *actuated dof* in (4.12a) with respect to the new set of control inputs $\overline{\mathbf{u}}$, defined in (4.16). This results in joint error dynamics whose origin is *UGES*.

Proof. By simply inserting (4.15) into (4.14), the dynamics reduce to $\ddot{\boldsymbol{q}}_a = \overline{\mathbf{u}}$, which is linear in the new control input $\overline{\mathbf{u}}$. Replacing $\overline{\mathbf{u}}$ with the joint control law in (4.16) yields the closed-loop error dynamics:

$$(\ddot{\boldsymbol{\phi}} - \ddot{\boldsymbol{\phi}}_{ref}) + k_d(\dot{\boldsymbol{\phi}} - \dot{\boldsymbol{\phi}}_{ref}) + k_p(\boldsymbol{\phi} - \boldsymbol{\phi}_{ref}) = \mathbf{0}, \quad (4.17a)$$

$$\ddot{\widetilde{\boldsymbol{\phi}}} + k_d\dot{\widetilde{\boldsymbol{\phi}}} + k_p\widetilde{\boldsymbol{\phi}} = \mathbf{0}, \quad (4.17b)$$

where $\widetilde{\boldsymbol{\phi}} = \boldsymbol{\phi} - \boldsymbol{\phi}_{ref}$ denotes the joint tracking error. Since the controller gains satisfy $k_p > 0$ and $k_d > 0$, by Proposition 1, (4.17) describes a system which has a *UGES* origin in the error variable. This concludes the proof. □

Code listing 4.3 shows the actual implementation in the MATLAB Simulator, where it's been used the identity $\boldsymbol{\phi} = \mathbf{D}\boldsymbol{\theta}$ being the state vector defined with respect to the link angles and not the joint angles.

Code listing 4.3: Partial feedback linearization controller: `fblin`

```
%% fb linearizing controller using (2.44)

function u = fblin(x, phi_ref, phi_ref_dot, phi_ref_ddot)
    global Nl l m J K V H D e ct cn

    %TUNING ---------
    Kp = 30; Kd = 37;
    %----------------

    % ...
    % from theta to Fr of {lst:dyn233} : lines [6--23]
    % ...

    %theta->phi
    phi = D*theta;
    phi_dot = D*thetadot;

    % ...
    % from M_bar to G2 of {lst:dyn241} : lines [12--28]
    % ...

    M22inv = inv(M22);

    %controller matrix terms
    linearizing_term = M11 - M12*M22inv*M21;
    friction_term = G1 - M12*M22inv*G2;
    velocity_term = W1 - M12*M22inv*W2;

    %fb linearizing control law
    u_bar = phi_ref_ddot - Kd * (phi_dot - phi_ref_dot) - Kp * (phi - phi_ref);
    u = linearizing_term*u_bar + velocity_term + friction_term*Fr;
end
```

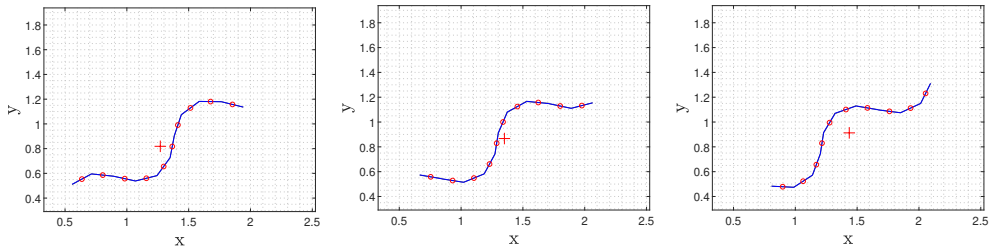
4.2.2 Gait Pattern

It only remains to address how to choose and obtain the references. One of the most common form of locomotion through which snakes propel forward is called *lateral ondulation*. During this motion the snake exhibits rhythmic undulations that resemble a wave propagating backward along the body from head to tail [3]. In accordance with Hirose [4] the movement can be described by the well-known mathematical curve called *serpenoid curve* which can be further well-approximated (since the snake-robot is a discrete approximation of biological snakes) by imposing the sinusoidal reference signal for the i^{th} joint angle

$$\boxed{\phi_{i,ref} = g(i,N)\ \alpha \sin(\omega t + (i-1)\delta) + \phi_0,} \tag{4.18}$$

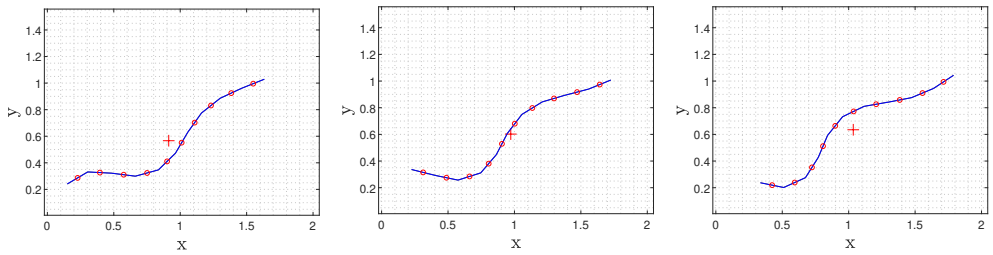
where $i \in [1,\dots,N-1]$, α and ω are, respectively, amplitude and angular frequency of the sinusoidal joint motion, δ determines the phase shift between successive joints and ϕ_0 can be used as an overall offset, identical for all the joints, useful to steer the direction of the movement as will be introduced in Section 4.4.1. The function $g(i,N) : \mathbb{R} \mapsto [0,1]$ scales the amplitude of the joints giving a different shade to the motion. We mainly use two version of it:

- $g(i,N) = 1$ gives the traditional motion pattern called *lateral ondulation*, regular throughout the body of the snake (see Figure 4.3);
- $g(i,N) = \frac{N-i}{N+i}$ gives instead what is called *eel-like* motion, distributing the ondulation predominantly across the tail, while maintaining a relatively straight orientation of the head, mimicking the swimming patterns of *eels* (see Figure 4.4).



(a) $t = 10$ [s] **(b)** $t = 10.5$ [s] **(c)** $t = 11$ [s]

Figure 4.3: *lateral ondulation* using $\alpha = 40°,\ \omega = 120°/s,\ \delta = 50°,\ \phi_0 = 0°$.



(a) $t = 10$ [s] **(b)** $t = 10.5$ [s] **(c)** $t = 11$ [s]

Figure 4.4: *eel-like gait pattern* using $\alpha = 40°,\ \omega = 120°/s,\ \delta = 50°,\ \phi_0 = 0°$.

We refer to [1–3, 9, 23] for detailed studies on propulsion synthesis and motion of biological snakes. Liljebäck *et al.* [1] proposes a method to choose the optimal δ to maximize the forward propulsion based on number of links N, while Mohammadi *et al.* [9] shows how to achieve velocity control by use of state compensators on α and ω.

In our work we treat α, ω, δ as constant fixed parameters, characteristics of the *gait pattern*, and we assume $\phi_0 = 0$ where not explicitly mentioned, i.e. before targeting the path-following problem in Section 4.4.1. If ϕ_0 is assumed a constant offset we can easily obtain $\dot{\boldsymbol{\phi}}_{ref}$ and $\ddot{\boldsymbol{\phi}}_{ref}$ analytically by deriving (4.18) as

$$\dot{\phi}_{i,ref} = g(i,N)\ \alpha\omega\cos(\omega t + (i-1)\delta), \tag{4.19a}$$

$$\ddot{\phi}_{i,ref} = -g(i,N)\ \alpha\omega^2\sin(\omega t + (i-1)\delta), \tag{4.19b}$$

Code listing 4.4 shows an implementation of the two motion pattern. Please note that they don't include the addition of ϕ_0, since that is done in an external stage for reasons that would be clearer later.

Code listing 4.4: Gait pattern generator: `lat_ond, eel_like`

```
%% create gait swimming reference

function [phi_ref, phi_ref_dot, phi_ref_ddot] = lat_ond(t)
    global alpha omega delta Nl;

    ii = (1:Nl-1)';
    phi_ref = alpha*sin(omega*t + (ii-1)*delta);
    phi_ref_dot = omega*alpha*cos(omega*t + (ii-1)*delta);
    phi_ref_ddot = -(omega^2)*alpha*sin(omega*t + (ii-1)*delta);
end

function [phi_ref, phi_ref_dot, phi_ref_ddot] = eel_like(t)
    global alpha omega delta Nl;

    ii = (1:Nl-1)';
    g = (Nl-ii)/(Nl+1);
    phi_ref = g .* (alpha*sin(omega*t + (ii-1)*delta));
    phi_ref_dot = g .* (omega*alpha*cos(omega*t + (ii-1)*delta));
    phi_ref_ddot = g .* (-(omega^2)*alpha*sin(omega*t + (ii-1)*delta));
end
```

4.2.3 Simulation setup and results

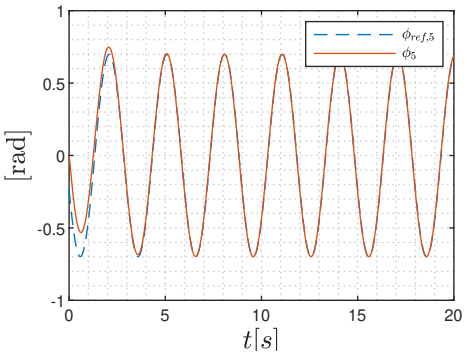
The model in (4.5) was implemented in `MATLAB R2023b` by the simulator proposed , where the dynamics of the model is solved by use of `dyn233` (see Code listing 4.1), or equivalently, `dyn241` (see Code listing 4.2), using a *Runge-Kutta 4 (RK4)* solver (see Code listing 4.5) for integration with a fixed time-step of $dt = 0.01$ $[s]$. We consider a snake robot with $N = 10$ links of length $2l = 0.18$ [m], mass $m = 1.56$ [kg], and moment of inertia $J = 0.0042$ [kgm^2]. The *lateral ondulation* ($g(i,N) = 1$) gait pattern parameters set as $\alpha = 40\pi/180$ [rad], $\omega = 120\pi/180$ [rad/s] and $\delta = 50\pi/180$ [rad]. The snake's initial conditions are $\boldsymbol{\theta}(0) = [0,\ \ldots,\ 0]^T$ [rad], $\dot{\boldsymbol{\theta}}(0) = [0,\ \ldots,\ 0]^T$ [rad/s], $\mathbf{p}(0) = [0,0]^T$ [m], $\dot{\mathbf{p}}(0) = [0,0]^T$ [m/s]. The here outlined setup will be maintained for the whole work presented here.

Code listing 4.5: RK4 integration method: `RK4`

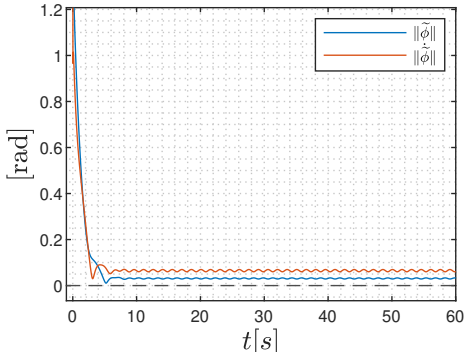
```
%% integration formulas

function x_next = euler_step(x,u,law,dt)
    x_next = x + dt*law(x,u);
end

function x_next = RK4(x,u,law,dt)
    k1 = law(x, u);
    k2 = law(x+dt/2*k1, u);
    k3 = law(x+dt/2*k2, u);
    k4 = law(x+dt*k3, u);

    x_next = x + dt/6*(k1+2*k2+2*k3+k4);
```

```
14 end
```

Figure 4.5 shows how the controller described in Equations (4.15) and (4.16) effectively achieves exponential convergence of the joint angles to the references obtained from the gait pattern generator in (4.18).



(a) Tracking of its reference for joint angle ϕ_5 between $T = [0-20][s]$.



(b) Exponential convergence of $\|\tilde{\boldsymbol{\phi}}\|$ and $\|\dot{\tilde{\boldsymbol{\phi}}}\|$ tracking errors to 0.

Figure 4.5: Joint tracking with the controllers in Equations (4.15) and (4.16). Terrain parameters are set for the current experiment as $c_t = 0.5$ [Nsm^{-1}] and $c_n = 3$ [Nsm^{-1}], while controller gains are set as $k_p = 30$ and $k_d = 37$.

4.3 Body Shape Adaptive Control

In this section, we present a reformulation of the ground friction forces to enable the subsequent design of an adaptive control scheme. Classical adaptive control techniques require that the parameters to be estimated enter the system equations linearly–i.e., that the friction forces in (4.6) admit a regressor-based linear parametrization. However, in their original form, the friction forces are non-linearly linked with the systems state variables, requiring them to get isolated for an adaptation scenario. Moreover, directly adapting the friction force vector is impractical, as it is state-dependent and evolves at the same rate as the system dynamics. This would violate the typical assumption in adaptive control where parameter variations should occur on a slower time scale. For these reasons, adapting the friction coefficients – rather than the forces themselves – appears to be a more suitable and tractable approach.

4.3.1 Approach

The following derivation takes inspiration from [5] and [1], combining their strength to develop a robust and reliable adaptive framework, addressing the challenge of designing a friction-adaptive controller based on the partial feedback linearized body-shape controller previously introduced.

More specifically, we follow [5] to rewrite the viscous friction force vector in a regression-based form suitable for an adaptive control scenario, from which we also adopt a very similar Lyapunov-based design for the adaptive laws but we adapt it to the body-shape controller proposed by [1] instead of the *VHC* controller introduced by Mohammadi *et al.* [9], which is the one adopted by Chitikena *et al.* [5]. This is reasonably motivated by the planned extension of the control architecture to achieve path-following, rather than manoeuvring control, which allows us to rely on a simpler control structure. The block diagram of the control architecture illustrated previously in Figure 4.2 extends now to what's shown in Figure 4.6.

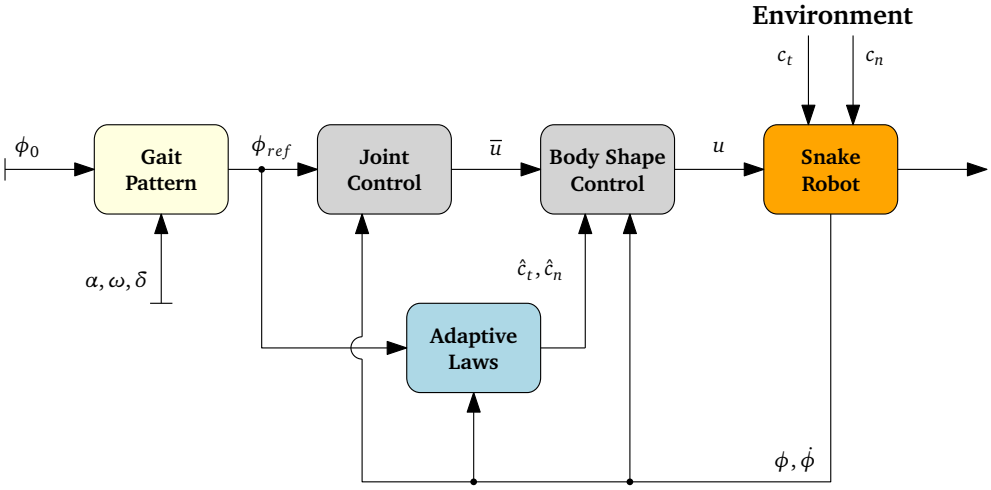


Figure 4.6: Block-diagram of the partial feedback linearized adaptive body-shape controller

4.3.2 Method

We begin by showing how the friction force vector $\mathbf{f}_R$ can be expressed in a regression-based form onto which adaptive control techniques can be invoked, following [5].

Proposition 3. The friction force vector can be expressed linearly with respect to the friction coefficients as $\mathbf{f}_R = -\mathbf{G_{cs}}(Bx)\cdot\mathbf{c}$, where $\mathbf{c} = [c_t, c_n]^T$ and $\mathbf{G_{cs}}(Bx)$ contains solely terms involving the robot's state variables.

Proof. Recalling from Equation (4.6), the friction force vector is expressed as

$$\mathbf{f}_R = -\begin{bmatrix} c_t(\mathbf{C}_\theta)^2 + c_n(\mathbf{S}_\theta)^2 & (c_t - c_n)\mathbf{S}_\theta\mathbf{C}_\theta \\ (c_t - c_n)\mathbf{S}_\theta\mathbf{C}_\theta & c_t(\mathbf{S}_\theta)^2 + c_n(\mathbf{C}_\theta)^2 \end{bmatrix}\begin{bmatrix} \dot{\mathbf{X}} \\ \dot{\mathbf{Y}} \end{bmatrix}, \tag{4.20}$$

The goal is to rewrite the expression in a linear way with respect to the friction coefficients c_t, c_n, as opposed to the link velocities $\dot{\mathbf{X}}$ and $\dot{\mathbf{Y}}$. We can proceed to

isolate the friction coefficients c_t, c_n from the state-dependent expressions as

$$\mathbf{f}_R = -\underbrace{\begin{bmatrix} C_{\boldsymbol{\theta}} & -S_{\boldsymbol{\theta}} \\ S_{\boldsymbol{\theta}} & C_{\boldsymbol{\theta}} \end{bmatrix}}_{\mathrm{R}_{\boldsymbol{\theta}}} \begin{bmatrix} c_t I_N & 0 \\ 0 & c_n I_N \end{bmatrix} \begin{bmatrix} C_{\boldsymbol{\theta}} & S_{\boldsymbol{\theta}} \\ -S_{\boldsymbol{\theta}} & C_{\boldsymbol{\theta}} \end{bmatrix} \underbrace{\begin{bmatrix} \dot{\mathbf{X}} \\ \dot{\mathbf{Y}} \end{bmatrix}}_{\mathbf{v}^{\text{global}}} \tag{4.21a}$$

$$= -\mathrm{R}_{\boldsymbol{\theta}} \begin{bmatrix} c_t I_N & 0 \\ 0 & c_n I_N \end{bmatrix} \mathrm{R}_{\boldsymbol{\theta}}^{\top} \mathbf{v}^{\text{global}} \tag{4.21b}$$

$$= -\mathrm{R}_{\boldsymbol{\theta}} \Bigg(c_t \underbrace{\begin{bmatrix} I_N & 0 \\ 0 & 0 \end{bmatrix}}_{I_{\mathrm{t}}} + c_n \underbrace{\begin{bmatrix} 0 & 0 \\ 0 & I_N \end{bmatrix}}_{I_{\mathrm{n}}} \Bigg) \mathrm{R}_{\boldsymbol{\theta}}^{\top} \mathbf{v}^{\text{global}} \tag{4.21c}$$

$$= -c_t \mathrm{R}_\theta I_{\mathrm{t}} \mathrm{R}_\theta^{\top} \mathbf{v}^{\text{global}} + c_n \mathrm{R}_\theta I_{\mathrm{n}} \mathrm{R}_\theta^{\top} \mathbf{v}^{\text{global}} \tag{4.21d}$$

$$= -\begin{bmatrix} \mathrm{R}_\theta I_{\mathrm{t}} \mathrm{R}_\theta^{\top} \mathbf{v}^{\text{global}} & \mathrm{R}_\theta I_{\mathrm{n}} \mathrm{R}_\theta^{\top} \mathbf{v}^{\text{global}} \end{bmatrix} \begin{bmatrix} c_t \\ c_n \end{bmatrix} \tag{4.21e}$$

$$= -\underbrace{\begin{bmatrix} C_\theta \left(C_\theta \dot{\mathbf{X}} + S_\theta \dot{\mathbf{Y}} \right) & -S_\theta \left(-S_\theta \dot{\mathbf{X}} + C_\theta \dot{\mathbf{Y}} \right) \\ S_\theta \left(C_\theta \dot{\mathbf{X}} + S_\theta \dot{\mathbf{Y}} \right) & C_\theta \left(-S_\theta \dot{\mathbf{X}} + C_\theta \dot{\mathbf{Y}} \right) \end{bmatrix}}_{\triangleq\, \mathbf{G_{cs}}(\mathbf{x})} \underbrace{\begin{bmatrix} c_t \\ c_n \end{bmatrix}}_{\mathbf{c}} \tag{4.21f}$$

$$= -\mathbf{G_{cs}}(\mathbf{x}) \cdot \mathbf{c} \tag{4.21g}$$

assembling the friction coefficients, for a more compact notation, into the vector $\mathbf{c}$. This concludes the proof. □

We now proceed to show how using an estimate of the friction coefficients, rather than their true values, affects the closed-loop dynamics by employing the linear form just introduced. This step allows us to explicitly see how the system depends on the friction coefficients, setting the stage for designing an adaptive control law to update their estimates. We start with the feedback linearizing control law from Equation (4.15), namely

$$\mathbf{u} = \underbrace{\left(\overline{\mathbf{M}}_{11} - \overline{\mathbf{M}}_{12} \overline{\mathbf{M}}_{22}^{-1} \overline{\mathbf{M}}_{21} \right)}_{\text{linearizing term } \mathbf{M_u}} \overline{\mathbf{u}} + \underbrace{\overline{\mathbf{W}}_1 - \overline{\mathbf{M}}_{12} \overline{\mathbf{M}}_{22}^{-1} \overline{\mathbf{W}}_2}_{\text{velocity term } \mathbf{W_u}} + \underbrace{\left(\overline{\mathbf{G}}_1 - \overline{\mathbf{M}}_{12} \overline{\mathbf{M}}_{22}^{-1} \overline{\mathbf{G}}_2 \right)}_{\text{friction term } \mathbf{G_u}} \hat{\mathbf{f}}_R \tag{4.22}$$

where now in the in the friction compensation term we use an estimate of the friction forces $\hat{\mathbf{f}}_R = -\mathbf{G_{cs}}(x) \cdot \hat{\mathbf{c}}$, obtained from an estimate of the friction coefficients $\hat{\mathbf{c}}$ using the linear expression just introduced in (4.21).

By substituting this modified control law into the partitioned system dynamics in Equation (4.14), we obtain the closed-loop dynamics of the joint angles:

$$\ddot{\boldsymbol{\phi}} = \overline{\mathbf{u}} + \underbrace{\left(\overline{\mathbf{M}}_{11} - \overline{\mathbf{M}}_{12} \overline{\mathbf{M}}_{22}^{-1} \overline{\mathbf{M}}_{21} \right)^{-1}}_{\text{linearizing term } \mathbf{M_u}} \cdot \underbrace{\left(\overline{\mathbf{G}}_1 - \overline{\mathbf{M}}_{12} \overline{\mathbf{M}}_{22}^{-1} \overline{\mathbf{G}}_2 \right)}_{\text{friction term } \mathbf{G_u}} \underbrace{\left(\hat{\mathbf{f}}_R - \mathbf{f}_R \right)}_{-\widetilde{\mathbf{f}}_R} \tag{4.23a}$$

$$= \overline{\mathbf{u}} + \underbrace{\mathbf{M_u}^{-1} \mathbf{G_u}}_{\triangleq\, \mathbf{C_f}} \cdot (-\widetilde{\mathbf{f}}_R) \tag{4.23b}$$

$$= \overline{\mathbf{u}} + \mathbf{C_f} \mathbf{G_{cs}}(\mathbf{x}) \cdot \widetilde{\mathbf{c}}, \tag{4.23c}$$

where, recalling that $\widetilde{\mathbf{f}}_R = -\mathbf{G_{cs}}(\mathbf{x}) \cdot \widetilde{\mathbf{c}}$, we have denoted the estimation error as $\widetilde{\mathbf{c}} = \mathbf{c} - \hat{\mathbf{c}}$. Substituting now for the linearized control input $\overline{\mathbf{u}}$ in (4.16), namely

$$\overline{\mathbf{u}} = \ddot{\boldsymbol{\phi}}_{ref} - k_d \dot{\widetilde{\boldsymbol{\phi}}} - k_p \widetilde{\boldsymbol{\phi}},$$

chosen to stabilize the tracking error dynamics, leads to the closed loop error dynamics:

$$\ddot{\boldsymbol{\phi}} = \ddot{\boldsymbol{\phi}}_{ref} - k_d \dot{\widetilde{\boldsymbol{\phi}}} - k_p \widetilde{\boldsymbol{\phi}} + \mathbf{C_f G_{cs}}(\mathbf{x}) \cdot \widetilde{\mathbf{c}} \tag{4.24a}$$

$$\ddot{\widetilde{\boldsymbol{\phi}}} + k_d \dot{\widetilde{\boldsymbol{\phi}}} + k_p \widetilde{\boldsymbol{\phi}} = \mathbf{C_f G_{cs}}(\mathbf{x}) \cdot \widetilde{\mathbf{c}}. \tag{4.24b}$$

This second-order system can be written more compactly in state-space form by introducing the augmented error vector $\mathbf{z} = [\dot{\widetilde{\boldsymbol{\phi}}},\ \widetilde{\boldsymbol{\phi}}]^T$:

$$\underbrace{\begin{bmatrix} \ddot{\widetilde{\boldsymbol{\phi}}} \\ \dot{\widetilde{\boldsymbol{\phi}}} \end{bmatrix}}_{\dot{\mathbf{z}}} = \underbrace{\begin{bmatrix} -k_d I_{N-1} & -k_p I_{N-1} \\ I_{N-1} & \mathbf{0} \end{bmatrix}}_{A<0} \underbrace{\begin{bmatrix} \dot{\widetilde{\boldsymbol{\phi}}} \\ \widetilde{\boldsymbol{\phi}} \end{bmatrix}}_{\mathbf{z}} + \underbrace{\begin{bmatrix} \mathbf{C_f G_{cs}}(\mathbf{x}) \\ \mathbf{0}_{(N-1)\times 2} \end{bmatrix}}_{F(\mathbf{x})} \widetilde{\mathbf{c}} \tag{4.25a}$$

$$\dot{\mathbf{z}} = A\mathbf{z} + F(\mathbf{x}) \cdot \widetilde{\mathbf{c}}. \tag{4.25b}$$

Here $A < 0$ is a *Hurwitz matrix* (i.e., the real part of all its *eigenvalues* is strictly negative), so that in the absence of estimation error ($\widetilde{\mathbf{c}} = \mathbf{0}$) the system is *exponentially stable*. The estimation error $\widetilde{\mathbf{c}}$ therefore acts as a disturbance input through the matrix $F(\mathbf{x})$, which depends only on the robot state variables. Since the true coefficients are assumed constant (or slowly varying compared to the system dynamics), we have $\dot{\mathbf{c}} = 0$, which implies

$$\dot{\widetilde{\mathbf{c}}} = -\dot{\hat{\mathbf{c}}}. \tag{4.26}$$

The resulting overall system is therefore a cascade:

$$\boxed{\begin{aligned} \dot{\mathbf{z}} &= A\mathbf{z} + F(\mathbf{x}) \cdot \widetilde{\mathbf{c}}, \\ \dot{\widetilde{\mathbf{c}}} &= -\dot{\hat{\mathbf{c}}}. \end{aligned}} \tag{4.27a, 4.27b}$$

This representation clearly shows that the closed-loop error dynamics is *Input to State Stable (ISS)* with respect to the estimation error. The next step is to design the adaptation law for $\hat{\mathbf{c}}$ so as to make $\widetilde{\mathbf{c}}$ stable, ensuring stability of the overall cascaded system.

Proposition 4. The origin of the cascaded system in (4.27) remains Lyapunov stable by selecting the adaptive law $\dot{\hat{\mathbf{c}}} = 2\Gamma F(\mathbf{x})^T P \mathbf{z}$, and thus the system's solutions remains bounded. Moreover, the gait tracking error $\mathbf{z} \to \mathbf{0}$ as $t \to \infty$. Further, under *PE* the entire state variable $(\mathbf{z}, \widetilde{\mathbf{c}}) \to 0$ as $t \to \infty$.

Proof. Consider the *Lyapunov Function Candidate (LFC)* for the system in (4.27):

$$\boxed{V(\mathbf{z},\widetilde{\mathbf{c}}) = \mathbf{z}^T P \mathbf{z} + \frac{1}{2}\widetilde{\mathbf{c}}^T \Gamma^{-1} \widetilde{\mathbf{c}},} \tag{4.28}$$

where $\Gamma = \Gamma^T > 0$ is a positive definite matrix (i.e. always invertible) and $P = P^T > 0$ is a symmetric, positive definite and constant matrix satisfying the Lyapunov equation:

$$A^T P + PA = -Q \text{ for some } Q > 0, \tag{4.29}$$

and $A < 0$ is the stable matrix in (4.25). Q can be chosen arbitrarily provided that the positiveness condition is respected. Following [5] we propose it to be

$$Q = \begin{bmatrix} k_d I_{N-1} & \mathbf{0} \\ \mathbf{0} & k_p I_{N-1} \end{bmatrix}. \tag{4.30}$$

Then, by Theorem 7 (4.29) has a unique constant solution P, which can be found analytically due to the form of A and the one chosen for Q, which is

$$P = \begin{bmatrix} \frac{1}{2}\left(\frac{1+k_d}{k_d}\right) I_{N-1} & \frac{1}{2} I_{N-1} \\ \frac{1}{2} I_{N-1} & \frac{1}{2}\left(k_p\left(\frac{1+k_d}{k_d}\right) + k_d\right) I_{N-1} \end{bmatrix}. \tag{4.31}$$

Differentiating V along the trajectories of the closed-loop dynamical system provided in (4.27) yields

$$\dot{V}(\mathbf{z},\widetilde{\mathbf{c}}) = \mathbf{z}^T P \dot{\mathbf{z}} + \dot{\mathbf{z}}^T P \mathbf{z} + \widetilde{\mathbf{c}}^T \Gamma^{-1} \dot{\widetilde{\mathbf{c}}} \tag{4.32a}$$

$$= \mathbf{z}^T P \left(A\mathbf{z} + F(\mathbf{x}) \cdot \widetilde{\mathbf{c}} \right) + \left(A\mathbf{z} + F(\mathbf{x}) \cdot \widetilde{\mathbf{c}} \right)^T P \mathbf{z} + \widetilde{\mathbf{c}}^T \Gamma^{-1} \dot{\widetilde{\mathbf{c}}} \tag{4.32b}$$

$$= \mathbf{z}^T \underbrace{(PA + A^T P)}_{-Q} \mathbf{z} + 2\widetilde{\mathbf{c}}^T F^T(\mathbf{x}) P \mathbf{z} + \widetilde{\mathbf{c}}^T \Gamma^{-1} \dot{\widetilde{\mathbf{c}}} \tag{4.32c}$$

$$= -\mathbf{z}^T Q \mathbf{z} + 2\widetilde{\mathbf{c}}^T F^T(\mathbf{x}) P \mathbf{z} - \widetilde{\mathbf{c}}^T \Gamma^{-1} \dot{\hat{\mathbf{c}}}. \tag{4.32d}$$

Selecting now the adaptation law

$$\boxed{\dot{\hat{\mathbf{c}}} = 2\Gamma F^T(\mathbf{x}) P \mathbf{z}} \tag{4.33}$$

and inserting it in (4.32d) leads to

$$\dot{V}(\mathbf{z},\mathbf{c}) = -\mathbf{z}^T Q \mathbf{z} + 2\widetilde{\mathbf{c}}^T F^T(\mathbf{x}) P \mathbf{z} - \widetilde{\mathbf{c}}^T \Gamma^{-1} \cdot 2\Gamma F^T(\mathbf{x}) P \mathbf{z} \tag{4.34a}$$

$$= -\mathbf{z}^T Q \mathbf{z} \leq 0. \tag{4.34b}$$

The *Lyapunov function* in (4.28) is positive definite, therefore lower bounded, and its derivative in (4.34b) is negative semi-definite. This implies that $V(t) \leq V(0)\ \forall t > 0$, therefore its arguments, $\mathbf{z}$ and $\widetilde{\mathbf{c}}$, are bounded, or more formally,

that $\mathbf{z}(\cdot) \in \mathcal{L}_\infty$ and also the estimation error $\widetilde{\mathbf{c}}(\cdot) \in \mathcal{L}_\infty$. This is sufficient to prove that the cascaded system in (4.27) is *Lyapunov stable* (see Theorem 1) but not that the dynamics of the *augmented error vector* $\mathbf{z}$, and so of the gait tracking errors, converges to 0. Integrating (4.34b),

$$0 \leq V(t) \leq V(0) - \int_0^t \mathbf{z}(\tau)^T Q \mathbf{z}(\tau)\, d\tau. \tag{4.35}$$

Since $\mathbf{z}(\cdot) \in \mathcal{L}_\infty$ and $Q > 0$ then the quantity $\mathbf{z}(\tau)^T Q \mathbf{z}(\tau)$ in the integral is always ≥ 0, and equal to 0 if and only if $\mathbf{z}(\tau) = 0$. Due to the lower-boundedness this only means that

$$\exists \lim_{t\to\infty} \int_0^t \mathbf{z}(\tau)^T Q \mathbf{z}(\tau)\, d\tau \geq 0 \tag{4.36}$$

and by Barbalat's Lemma (Theorem 3),

$$\lim_{t\to\infty} \mathbf{z}(t)^T Q \mathbf{z}(t) = \mathbf{0} \implies \lim_{t\to\infty} \mathbf{z}(t) = \mathbf{0} \tag{4.37}$$

since $Q > 0$. Furthermore, if the *PE* condition is satisfied, by Theorem 4 also the estimates $\hat{\mathbf{c}}$ converges to their true values $\mathbf{c}$, implying that $\widetilde{\mathbf{c}} \to 0$, which concludes the proof. □

Remark: The tracking error dynamics becomes UGES when the estimation error is $\widetilde{\mathbf{c}} = \mathbf{0}$ (see Proposition 1). This constitutes a remarkable advantage over the weaker *asymptotic convergence* to 0 of the joint tracking error achieved during adaptation of the estimation parameters, owing to the specific formulation in which we set up the adaptive problem. This is a remarkable result, highlighting the importance of having *PE* in our system to achieve faster convergence of the joint angles. Fortunately, achieving PE for our adaptive scheme is fairly easy, thanks to the linearized input and the sinusoidal reference gait.

We summarize here below the full adaptive scheme.

control law
$$\mathbf{u} = \left(\overline{\mathbf{M}}_{11} - \overline{\mathbf{M}}_{12}\overline{\mathbf{M}}_{22}^{-1}\overline{\mathbf{M}}_{21}\right)\overline{\mathbf{u}} + \overline{\mathbf{W}}_1 - \overline{\mathbf{M}}_{12}\overline{\mathbf{M}}_{22}^{-1}\overline{\mathbf{W}}_2 + \left(\overline{\mathbf{G}}_1 - \overline{\mathbf{M}}_{12}\overline{\mathbf{M}}_{22}^{-1}\overline{\mathbf{G}}_2\right)\hat{\mathbf{f}}_R \tag{4.38a}$$

stabilizing law
$$\overline{\mathbf{u}} = \ddot{\boldsymbol{\phi}}_{ref} - k_d\dot{\tilde{\boldsymbol{\phi}}} - k_p\tilde{\boldsymbol{\phi}} \tag{4.38b}$$

friction forces estimate
$$\hat{\mathbf{f}}_R = -\mathbf{G}_{\mathbf{cs}}(\mathbf{x})\cdot\hat{\mathbf{c}} \tag{4.38c}$$

$$\mathbf{G}_{\mathbf{cs}}(\mathbf{x}) = \begin{bmatrix} C_\theta\left(C_\theta\dot{\mathbf{X}} + S_\theta\dot{\mathbf{Y}}\right) & -S_\theta\left(-S_\theta\dot{\mathbf{X}} + C_\theta\dot{\mathbf{Y}}\right) \\ S_\theta\left(C_\theta\dot{\mathbf{X}} + S_\theta\dot{\mathbf{Y}}\right) & C_\theta\left(-S_\theta\dot{\mathbf{X}} + C_\theta\dot{\mathbf{Y}}\right) \end{bmatrix} \tag{4.38d}$$

control gains
$$k_p > 0,\ k_d > 0, \tag{4.38e}$$

adaptive law
$$\dot{\hat{\mathbf{c}}} = 2\Gamma F(\mathbf{x})^T P\mathbf{z}, \quad \hat{\mathbf{c}} = [\hat{c}_t\ \hat{c}_n]^T, \quad \mathbf{z} = [\dot{\tilde{\boldsymbol{\phi}}},\ \tilde{\boldsymbol{\phi}}]^T \tag{4.38f}$$

$$F(\mathbf{x}) = \begin{bmatrix} \mathbf{C}_\mathbf{f}\mathbf{G}_{\mathbf{cs}}(\mathbf{x}) \\ \mathbf{0}_{(N-1)\times 2} \end{bmatrix} \tag{4.38g}$$

$$\mathbf{C}_\mathbf{f} = \left(\overline{\mathbf{M}}_{11} - \overline{\mathbf{M}}_{12}\overline{\mathbf{M}}_{22}^{-1}\overline{\mathbf{M}}_{21}\right)^{-1}\cdot\left(\overline{\mathbf{G}}_1 - \overline{\mathbf{M}}_{12}\overline{\mathbf{M}}_{22}^{-1}\overline{\mathbf{G}}_2\right) \tag{4.38h}$$

lyapunov solution
$$P = \begin{bmatrix} \frac{1}{2}\left(\frac{1+k_d}{k_d}\right)I_{N-1} & \frac{1}{2}I_{N-1} \\ \frac{1}{2}I_{N-1} & \frac{1}{2}\left(k_p\left(\frac{1+k_d}{k_d}\right)+k_d\right)I_{N-1} \end{bmatrix} \tag{4.38i}$$

adaptation gain
$$\Gamma = \Gamma^T > 0. \tag{4.38j}$$

4.3.3 Simulation results

The implementation of the adaptive controller in (4.38) is shown in Code listing 4.6.

Code listing 4.6: Adaptive body-shape controller: `adap`

```
%% adaptive fb-lin controller ---------------------------------

function u = adap(x, phi_ref, phi_ref_dot, phi_ref_ddot)
    global Nl l m J K V H D e;
    global dt int_method ct cn c_hat maxc minc;

    %TUNING ---------
    Kp = 30; Kd = 37;
    Gamma = diag([12,15]);
    %----------------

    %(14b-21-22)
    Az = [-Kd*eye(Nl-1), -Kp*eye(Nl-1);
        eye(Nl-1), zeros(Nl-1, Nl-1)];
    Q = [Kd*eye(Nl-1), zeros(Nl-1,Nl-1);
         zeros(Nl-1,Nl-1), Kp*eye(Nl-1)];
    P = lyap(Az', Q);

```

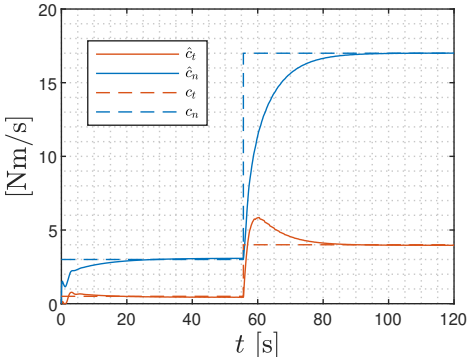
```
    %some checks
    assert(isequal(P,P'), 'P is not symmetric');
    assert(isequal(Q,Q'), 'Q is not symmetric');
    assert(all(eig(P)>0), 'P is not positive def');
    assert(all(eig(Q)>0), 'Q is not positive def');
    assert(all(abs(Az'*P+P*Az - -Q) < 1e-12, 'all'), ...
        'riccati equation is not satisfied');

    % ...
    % from theta to velocity_term of {lst:fblin} : lines [10--27]
    % ...

    % adaptive laws
    z = [phi_dot - phi_ref_dot; phi - phi_ref];
    Gcs = [Ct*(Ct*Xd + St*Yd), -St*(-St*Xd + Ct*Yd);
           St*(Ct*Xd + St*Yd), Ct*(-St*Xd + Ct*Yd)];
    Cf = inv(linearizing_term)*friction_term;
    F = [Cf*Gcs; zeros(Nl-1,2)];

    adap_law = (@(c,z) 2*Gamma*F'*P*z);
    c_hat = int_method(c_hat, z, adap_law, dt); %RK4

    % parameter projection
    c_hat(1) = max(minc, min(c_hat(1), maxc)); %ct
    c_hat(2) = max(minc, min(c_hat(2), maxc)); %cn

    % lyapunov analysis
    global v vdot;
    c_tilde = [ct;cn] - c_hat;
    v(1) = z'*P*z + 0.5*c_tilde'*inv(Gamma)*c_tilde;
    vdot(1) = -z'*Q*z;

    %estimate of the friction forces
    Fr_hat = -Gcs*c_hat;

    %fb linearizing control law
    u_bar = phi_ref_ddot - Kd * (phi_dot - phi_ref_dot) - Kp * (phi - phi_ref);
    u = linearizing_term*u_bar + velocity_term + friction_term*Fr_hat;
end
```

We perform two numerical experiments to effectively validate the performances of the adaptive controller. We keep the same base setup outlined in Section 4.2.3, except for the following modifications. The controller gains are still set as $k_p = 30$ and $k_d = 37$, and, in addition, $\Gamma = \text{diag}([12, 15])$. Since the friction coefficients c_t and c_n are strictly nonnegative physical quantities, the adaptive law is augmented with simple saturation bounds. This prevents the estimates from drifting into nonphysical negative values during transients or in presence of disturbances, which could otherwise compromise the controller's stability and physical interpretability. The clamping values for the friction coefficients are set as $c_{min} = 0.01$ [Ns/m] and $c_{max} = 100$ [Ns/m]. The initial guess of the estimates are set as $\hat{c}_t, \hat{c}_n = [c_{min}, c_{min}]$ [Ns/m]. The parameters here detailed will be maintained also across all the following simulations.

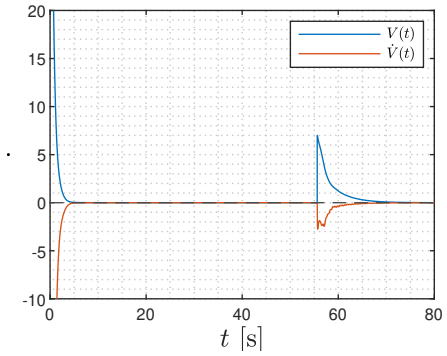
In the first experiment we run the simulation for $T = 120$ [s]. We simulate on two different terrains to enhance the validation of the setup, changing the values

of the friction coefficients. The true friction coefficients for the first terrain are set to $[c_t, c_n] = [0.5, 3]$ [Ns/m], whereas for the second terrain they are $[c_t, c_n] = [4, 17]$ [Ns/m]. For simplicity – and consistency with the model structure – the terrain switch occurs at $p_x = 10$ m and occurs simultaneously to all the links, rather than one by one.

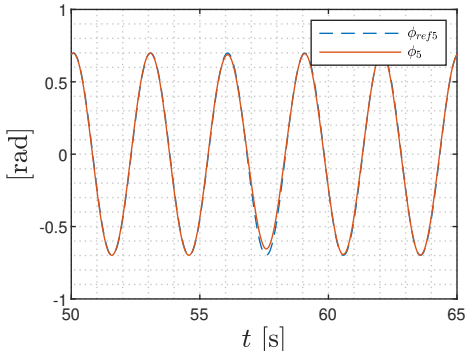
Figure 4.7a shows the time evolution of the estimates, highlighting convergence towards their true values due to the presence of *PE*. The simulation setup allows us to also carry out an analysis of the *Lyapunov Function* in (4.28), whose time-evolution is shown in Figure 4.7b. As final validation element we show the tracking performances for the 5^{th} joint in Figure 4.7c, as done previously for the Partial Feedback Linearized Body-Shape Controller in Figure 4.5a.



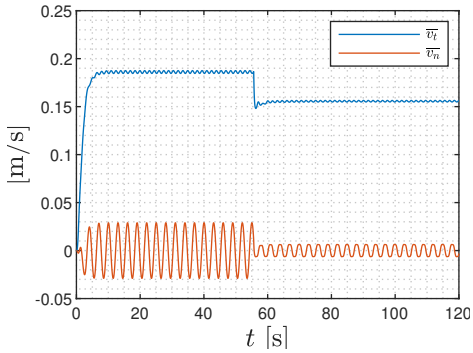
(a) Time evolution of the estimation parameters $\hat{c}_t, \hat{c}_n$ vs true values c_t, c_n.



(b) Time evolution of the *LFC* and its derivative. Zoom in $t \in [0-80]$ [s].



(c) Tracking performances on the 5^{th} joint angle. Zoom in $t \in [50-65]$ [s] to highlight the the change of terrain occurring roughly at $t = 57$ [s].



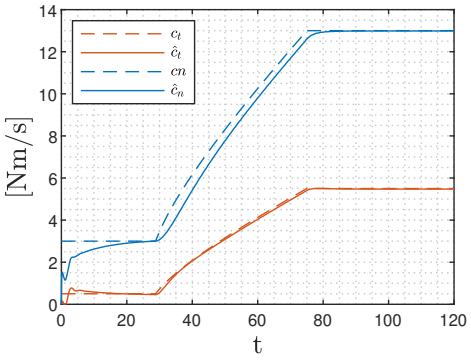
(d) Forward velocity $\overline{v}_t$ and normal velocity $\overline{v}_n$ of the *CM* of the snake as defined in Equation (4.3).

Figure 4.7: Simulation results for the first experiment

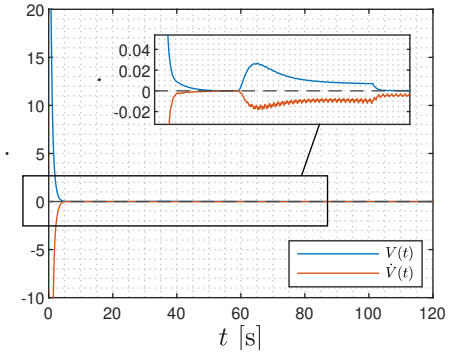
In the second experiment we also run the simulation for $T = 120$ [s], maintaining the same setup and starting with the first terrain ($c_t, c_n = [0.5, 3]$ [Ns/m]) for $x \leq 5$ [m]. Then we start inducing a slow variation of the friction coefficients with respect to the terrain in $5 < x \leq 10$ [m], characterized by $\frac{dc_t}{dx} = 1$ [Nm/s] per

meter and $\frac{dc_n}{dx} = 2\,[\mathrm{Nm/s}]$ per meter. It is crucial to ensure that forward propulsion is sustained by consistently maintaining the normal coefficient at a value greater than the tangential coefficient, which is ensured by the initial values of $c_n < c_t$ and the condition ($\frac{dc_t}{dx} \leq \frac{dc_n}{dx}$).

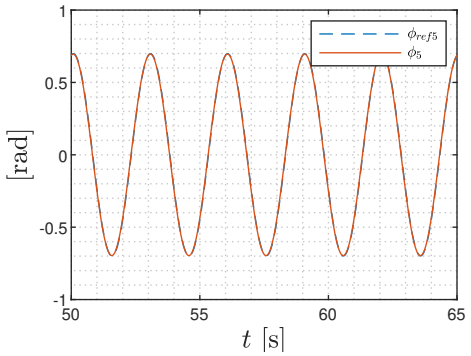
In this way we test the controller performance in a more challenging scenario, demonstrating its robust properties also in a varying friction scenario. Figure 4.8 shows the same elements previously analyzed during the first experiment.



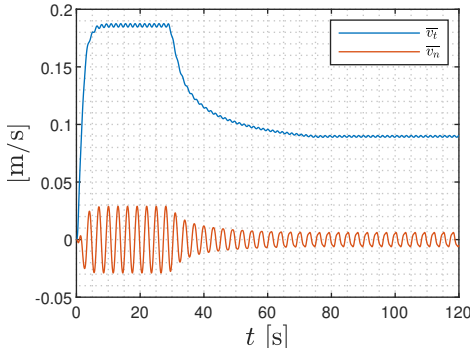
(a) Time evolution of the estimation parameters $\hat{c}_t, \hat{c}_n$ vs true values c_t, c_n.



(b) Time evolution of the Lyapunov function and its derivative. Zoom in $t \in [0-90]$ [s] to highlight the period of varying friction.



(c) Tracking performances on the 5^{th} joint angle. Zoom in $t \in [50-65]$ [s], during the change of friction. In presence of slow changes the losing of tracking is almost un-noticeable.



(d) Forward velocity $\overline{v}_t$ and normal velocity $\overline{v}_n$ of the *CM* of the snake. Please note the damping of both forward propulsion than normal oscillation due to the change of ratio on c_t and c_n.

Figure 4.8: Simulation results for the second experiment

4.3.4 Discussion

Chitikena *et al.* [5] slightly differs regarding the derivation of the Lyapunov-based adaptive law by including an additional term $-\epsilon\hat{\mathbf{c}}$, commonly referred in the literature as *leakage term*. We have chosen not to include it, as simulation results

demonstrated satisfactory performance, and we found no convincing reason to complicate a design that already works effectively. This omitted term is typically used to enforce practical stability of the parameter estimates by shaping their dynamics as a virtual cascaded *ISS* system itself driven by the adaptive law as input signal. Such a modification is particularly useful in the presence of *unmodeled* dynamics or disturbances, as it prevents the estimates from drifting indefinitely or diverging due to model mismatch. However, this comes at the cost of introducing a steady-state error in the parameter estimates, which we liked to avoid.

4.4 Path-following Control

In this section we introduce a strategy to steer the motion of the snake robot through the *gait* parameter ϕ_0, and analyze whereas this can be used for implementing a path-following controller. The proposed approach consists of two main components: a *line-of-sight (LOS)* guidance scheme, detailed in Section 4.4.3, and a heading controller that drives the robots heading towards the reference angle provided by the guidance scheme. To account for the – now – dynamic behavior of ϕ_0 we make use of reference filters as introduced in Section 4.4.5. The overall control architecture is then again evaluated through simulation, whose results are presented in Section 4.4.6. We conclude by highlighting how the complexity of the model dynamics limits the design of heading controllers to relatively simple structures, such as PD control. This motivates the introduction of a *simplified* model of the snake robot, which is presented in the following chapter.

4.4.1 Turning Motion

As previously introduced the remaining parameter ϕ_0 in the *gait pattern* reference is, in fact, a *degree of freedom* and can be exploited to actively induce turning motion in the robot. The reason behind this is illustrated in Figure 4.9.

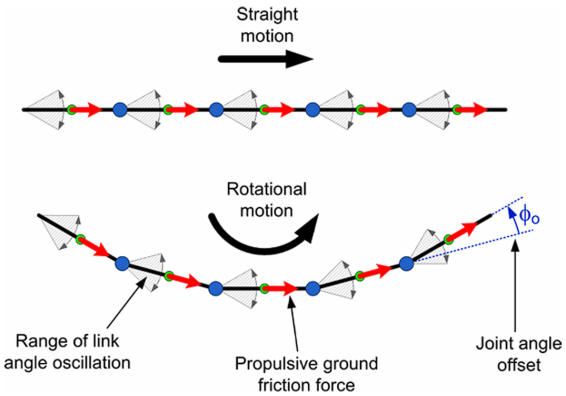
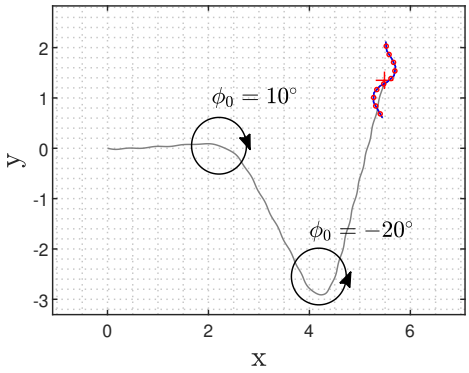


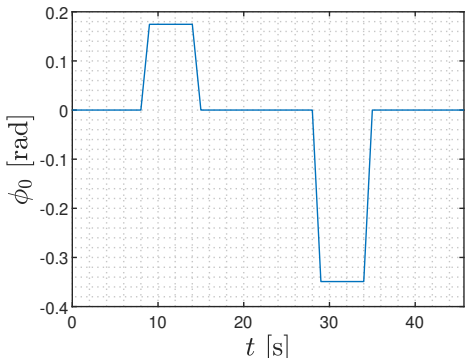
Figure 4.9: Each link angle oscillates about a straight line (i.e., the forward direction) which can be indirectly steered when ϕ_0 is non-zero. Illustration from [1].

To investigate how ϕ_0 affects the forward motion direction of the snake we conduct a simple experiment. We simulate for $T = 45$ [s]. For $t \in [10-15]$ [s] ϕ_0 is kept at the fixed value of $10\pi/180$ [rad], while for $t \in [30-35]$ [s] to $\phi_0 = -20\pi/180$ [rad]. Its value is maintained to 0 outside these time ranges. As a result of it, as can be seen from Figure 4.10, a positive ϕ_0 induces a clockwise motion, whereas a negative ϕ_0 a counter-clockwise one.

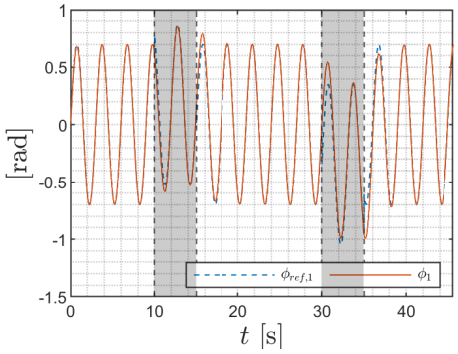
The resulting turning radius is affected both by the value of ϕ_0 (the greater, the tighter) but also from the forward velocity $\overline{v}_t$, since the faster the snake, the wider the radius. This is in accordance with the heading controller proposed by [1] for the *simplified* model that will be introduced in Chapter 5, which also takes in account the forward velocity.



(a) Trace of (p_x, p_y) during the simulation. The joint angle offset is maintained to 0 except for $t \in [10, 15]$ s where $\phi_0 = 10°$ and for $t \in [30, 35]$ s where $\phi_0 = -20°$.



(b) $\phi_0(t)$ joint offset across the simulation.



(c) Joint angle ϕ_1 vs reference $\phi_{ref,1}$ during the simulation. Highlighted in gray the periods where $\phi_0 \neq 0$.

Figure 4.10: Simulation results of turning motion

This is also supported from a simple observation on the conventions adopted for the kinematics of the snake robot (see Section 4.1.1). Defining the joint angles as $\phi_i = \theta_i - \theta_{i+1}$ and ϕ_0 with respect to the *joint* angles, implies that when ϕ_i is positive it creates a *negative* angle between adjacent links, i.e., the next link appear clockwise-r than then previous one. And due to the fact that we named links going

from tail to head, a clockwise angle between successive links stretches the robot in a, indeed, clockwise shape (see Figure 4.11). ***Please note:*** this is counter-intuitive with respect to the traditional definition of angles and the sign-mismatch has to kept in account when designing heading controllers.

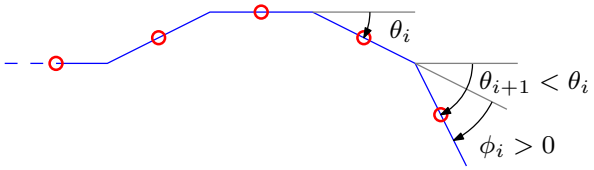


Figure 4.11: Effect of ϕ_0 on *joint* angles due to the conventions adopted.

4.4.2 Control Objective

As observed from experiments, the *CM* position $\mathbf{p}$ does not follow a straight trajectory during locomotion, but instead oscillates periodically around the straight line pointing in the forward direction of the robot. A similar behavior is expected for the heading $\overline{\theta}$ (as defined in (4.2)) as well. Given the *under-actuated* nature of the overall dynamics, it makes no sense to attempt regulation of these states to stationary values along a desired straight path [1]. We therefore relax the control objective onto regulation of the CM of the robot such that p_y oscillates about y_{ref} and the heading $\overline{\theta}$ about a reference angle $\overline{\theta}_{ref}$, rather than converge to it. A formal analysis of the resulting closed-loop behavior is presented in [1, Chapter 5] by making use of a *Poincaré map* to demonstrates that the system state variables trace out an *exponentially stable* periodic orbit. We remind to that for further details.

4.4.3 LOS guidance

In order to steer the robot towards the desired straight path (i.e. y_{ref}), we employ a *LOS* guidance law from [1], defining the heading reference angle according to

$$\boxed{\overline{\theta}_{ref} = -\arctan\left(\frac{\overline{p_y}}{\Delta}\right),} \tag{4.39}$$

where $\overline{p_y} = p_y - y_{ref}$ denotes the cross-track error and Δ is a design parameter known as *look-ahead distance* which influences the rate of convergence to the desired path. A common choice in the literature is to set Δ to at least twice the length of the vehicle. However, given that the snake robot tends to remain curled during forward motion, this rule of thumb may be relaxed. We propose selecting it as the fully outstretched length of the snake (i.e. $\Delta = N \cdot 2l$). Figure 4.12 illustrates the geometry behind the guidance scheme.

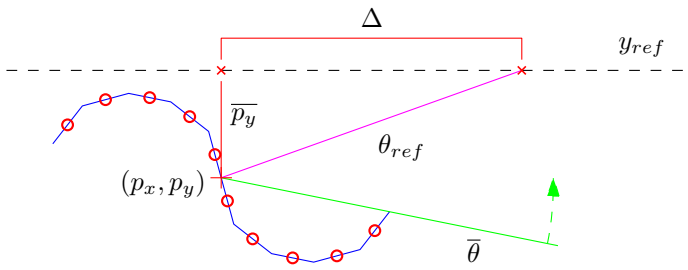


Figure 4.12: LOS principle for a straight path-following problem, approximating the vehicle with the point representing his CM.

By regulating $\overline{\theta}$ around the LOS reference angle $\overline{\theta}_{ref}$ we will fulfill the control objectives making the snake converge to (or rather, oscillate around) the desired path.

4.4.4 Heading Control

The simplest alternative for achieving the control objective is by use of a proportional controller, as proposed by [1], defined by the following control law:

$$\phi_0 = k_p(\overline{\theta} - \overline{\theta}_{ref}), \tag{4.40}$$

where $k_p > 0$ is a controller gain. We instead propose extending it to include also a derivative term, as

$$\boxed{\phi_0 = k_p(\overline{\theta} - \overline{\theta}_{ref}) + k_d(\dot{\overline{\theta}} - \dot{\overline{\theta}}_{ref}),} \tag{4.41}$$

where $k_p > 0$ and $k_d > 0$ are controller gains, $\dot{\overline{\theta}}$ is the time derivative of $\overline{\theta}$ and follows from its definition in (4.2), as

$$\dot{\overline{\theta}} = \frac{1}{N}\sum_{i=1}^{N}\dot{\theta}_i, \tag{4.42}$$

and $\dot{\overline{\theta}}_{ref}$ is the time derivative of $\overline{\theta}_{ref}$ from the LOS guidance scheme. ***Please note*** that the presented control laws in Equations (4.40) and (4.41) appear to perform positive-feedback instead of negative-one. This is because of the sign-mismatch introduced in Section 4.4.1 between ϕ_0 and its effects. The overall architecture of the path-following controller is pictured as a block-diagram in Figure 4.13.

In practical scenarios where we don't possess the knowledge of $\dot{\overline{\theta}}_{ref}$ or cannot obtain in an analytical way the control law in (4.41) can be relaxed to

$$\phi_0 = k_p(\overline{\theta} - \overline{\theta}_{ref}) + k_d\dot{\overline{\theta}}, \tag{4.43}$$

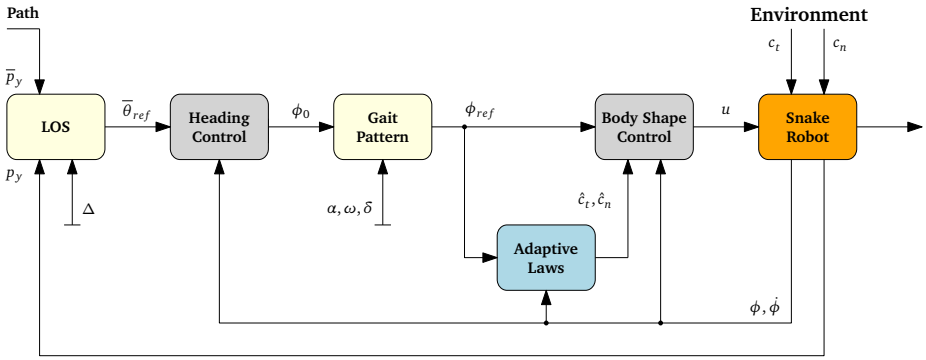


Figure 4.13: Path-following control architecture, comprehensive of a LOS guidance scheme to translate the path to a reference angle and a PD-heading controller for tracking. The body-shape inner control is managed by the adaptive controller presented in Section 4.3.

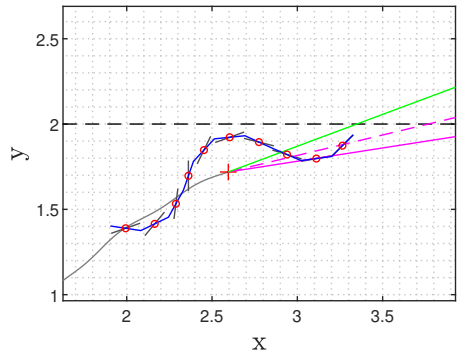
assuming that the changes in the reference angle are much slower with respect to the system dynamics, or by passing $\overline{\theta}_{ref}$ through a low-pass filter, as we present in the next section. Additionally, attempting to implement the scheme reveals an unexpected phenomenon: a loss of tracking in the joint angles, as illustrated in Figure 4.14. This behavior can be attributed to the fact that ϕ_0 is no longer a constant but varies over time, resulting in non-zero derivatives $\dot{\phi}_0, \ddot{\phi}_0 \neq 0$. Consequently, the analytical expressions for the reference derivatives given in (4.19) are no longer exact, which compromises the accuracy of the *feed-forward* terms and leads to a degraded tracking performance. The tracking error persists over time and does not vanish even once converged to the path, since the oscillatory behavior requires a continuous correction from the heading controller.

Consequently, this triggers an intricate chain reaction within the body-shape controller, since its adaptive laws are driven by the tracking errors $\tilde{\boldsymbol{\phi}}, \dot{\tilde{\boldsymbol{\phi}}}$, which are non-vanishing (see Figure 4.15a). This results in a continual adaptation of the estimation parameters that, outside of a simulation environment, may superficially resemble noise to the untrained observer (see Figure 4.15); however, it is actually attributable to a never ending descent along the Lyapunov function (see Figure 4.15b), which we would like to avoid.

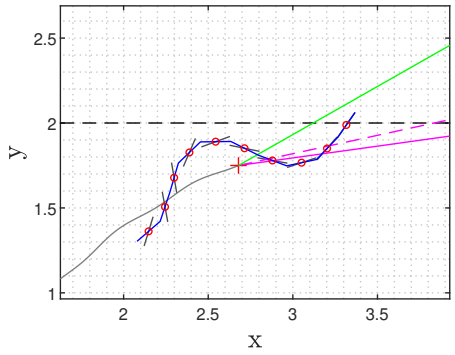
These considerations are sufficient to motivate the introduction of two reference filters: one for the LOS module to provide $\overline{\theta}_{ref}$ such that we're able to use the control law (4.41), and another for ϕ_0 to provide $\dot{\phi}_0, \ddot{\phi}_0$, thereby extending the control architecture in Figure 4.13 to the one in Figure 4.16.

4.4.5 Reference Filters

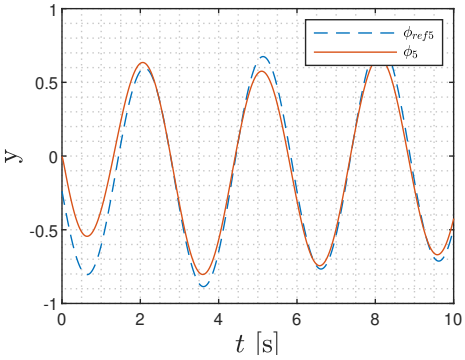
Beyond providing time derivatives of a reference signal when analytical expressions are unavailable, reference filters are useful also to (i) smooth discontinuities, such as steps in reference signals, thereby preventing unbounded responses in PD controllers and (ii) helping to keep the control input within the physical limits of



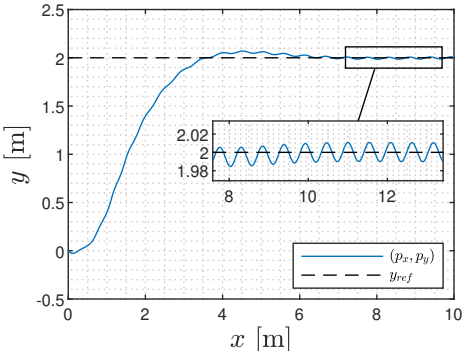
(a) Snapshot at $t = 20.0$ [s]



(b) Snapshot at $t = 20.5$ [s]



(c) Degradation in tracking performance on the 5^{th} joint angle within the time interval $t = [0-10]$ [s].



(d) Trace of the CM during the experiment. Zoom to emphasize the non-vanishing oscillatory dynamics of the CM and the heading.

Figure 4.14: Simulation experiment of the straight-line path following controller from (4.43) with a reference path $y_{ref} = 2$. The initial conditions, gait parameters and controller gains are set as previously introduced in Sections 4.2.3 and 4.3.3. For the heading controller the gains are set as $k_p = 0.3$, $k_d = 0.1$, $\Delta = N \cdot 2l$. The terrain friction coefficients are instead $c_t, c_n = [0.5, 3]$ [Ns/m] and stays constant across the simulation. Highlighted in gray under the blue links the reference angles translated from joint angles to link angles using $\boldsymbol{\theta}_{ref} = \mathbf{D}\boldsymbol{\phi}_{ref}$, and after aligned with the link absolute position $\mathbf{X}, \mathbf{Y}$, showing the loss of tracking.

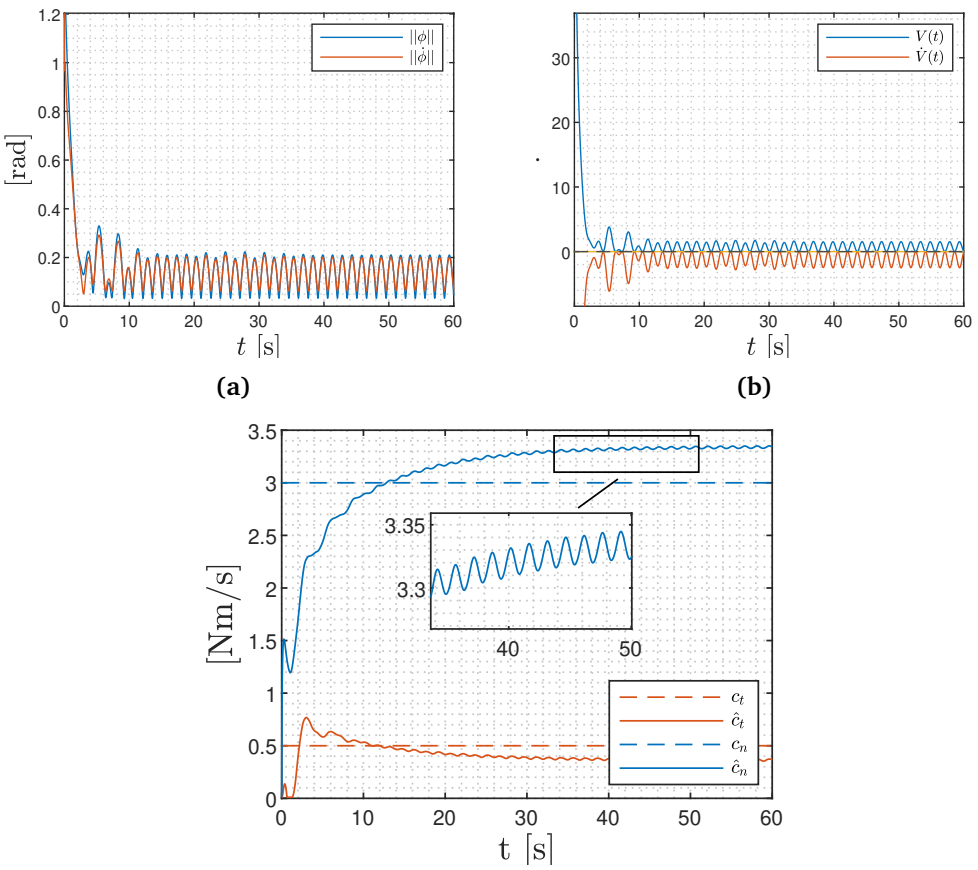


Figure 4.15: The non-vanishing gait tracking errors $\tilde{\boldsymbol{\phi}}, \dot{\tilde{\boldsymbol{\phi}}}$ induces a constant perturbation of the Lyapunov function resulting in a continuous adaptation of the estimation parameters, as well as a steady-state error in the estimate.

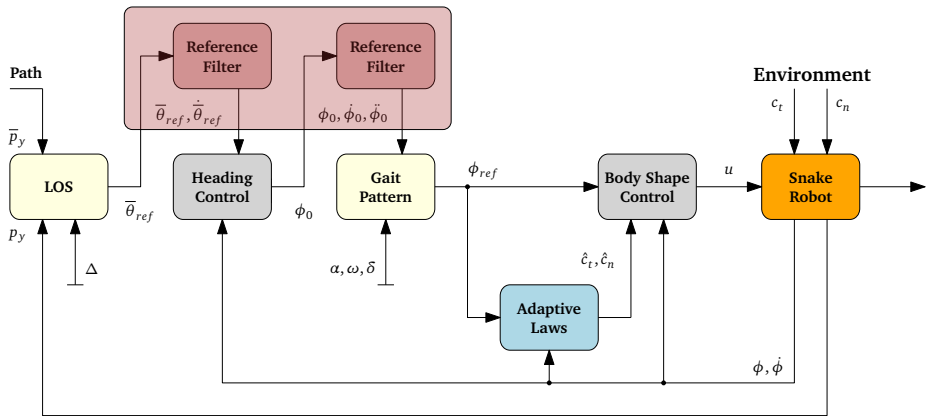


Figure 4.16: Practical implementation of the controller in Figure 4.13 by adding the reference filters.

the actuators, mitigating phenomena such as torque chattering and saturation.

A 3rd-order low-pass filter reference model is utilized to fulfill each of the two objectives previously presented, i.e. to smooth the reference signal $\phi_0 \rightarrow \phi_{0,filt}$ and provide its first and second time derivative $\dot{\phi}_0, \ddot{\phi}_0$. The same applies to $\overline{\theta}_{ref}$. A 3rd-order low-pass filter can be obtained by cascading a 2nd-order filter with a 1st-order filter, both having the same *natural frequency* ω_c, whose transfer function can be written as

$$\frac{x_{\text{ref}}}{r} = \frac{{\omega_c}^3}{s^3 + (2\zeta+1)\omega_c s^2 + (2\zeta+1){\omega_c}^2 s + {\omega_c}^3}, \tag{4.44}$$

by denoting the unfiltered reference signal as r and the filtered state-reference as x_{ref}, and where $\omega_c > 0$ and $\zeta > 0$ are respectively, the *natural frequency* of the filters and the *damping ratio* of the 2nd-order low pass filter. Following [1], we can implement he filter in a state-space form by

$$x_{\text{ref}}^{(3)} + (2\zeta+1)\omega_c x_{\text{ref}}^{(2)} + (2\zeta+1){\omega_c}^2 x_{\text{ref}}^{(1)} + {\omega_c}^3 x_{\text{ref}} = {\omega_c}^3 r, \tag{4.45a}$$

$$\frac{d}{dt}\begin{bmatrix} x_{ref} \\ \dot{x}_{ref} \\ \ddot{x}_{ref} \end{bmatrix} = \begin{bmatrix} 0 & 1 & 0 \\ 0 & 0 & 1 \\ -{\omega_c}^3 & -(2\zeta+1){\omega_c}^2 & -(2\zeta+1)\omega_c \end{bmatrix} \begin{bmatrix} x_{ref} \\ \dot{x}_{ref} \\ \ddot{x}_{ref} \end{bmatrix} + \begin{bmatrix} 0 \\ 0 \\ {\omega_c}^3 \end{bmatrix} r. \tag{4.45b}$$

A crucial aspect is achieving a proper balance between the responsiveness of the filter and the magnitude of its state variables, which is governed by the parameters ω_c and ζ. A faster filter allows x_{ref} to more closely follow r, preserving fidelity to the original unfiltered reference. However, this comes at the risk of producing excessively large derivatives $\dot{x}_{ref}$ and $\ddot{x}_{ref}$, which may result in actuator saturation or unrealistic control efforts. Conversely, a slower filter introduces delay in the control loop and may result in x_{ref} no longer representing the intended reference r with sufficient accuracy for real-time control.

We propose $\Delta T = 2\pi/\omega_c = 4$ [s] and $\zeta = 1$ (*critically damped*), such that the step response of the filter has no overshoot and a step-response rise-time 5% to 95% of approximately $3\tau = 3 \cdot 1/\omega_c \approx 1.91$ [s], which has been seen from experiments to be a good compromise.

The reference filter is implemented as in Code listing 4.7, and the one used to add the contribute of the ϕ_0 filtered derivatives to the gait references as shown in Code listing 4.8.

Code listing 4.7: 3rd order reference filter: `ref_filter`

```
% 3rd order reference filter ---------------------------------

function [x, xdot, xddot] = ref_filter(x, r, xi, deltaT)
    global dt;

```

```
    wc = 2*pi/deltaT;
    F = [0, 1, 0; 0, 0, 1; -wc^3, -(2*xi+1)*(wc^2), -(2*xi+1)*wc];
    tf = @(x,r) F*x + [0; 0; wc^3]*r;

    z = euler_step(x, r, tf, dt);
    x=z(1); xdot=z(2); xddot=z(3);
end
```

Code listing 4.8: ϕ_0 reference filter: `add_phi0`

```
% phi0, phi0_dot, phi0_ddot ---------------------------------

function [phi_ref, phi_ref_dot, phi_ref_ddot] = ...
    add_phi0(phi_ref, phi_ref_dot, phi_ref_ddot, phi0)

    % initialization
    persistent phi0filt phi0filt_dot phi0filt_ddot;
    if isempty(phi0filt)
        [phi0filt, phi0filt_dot, phi0filt_ddot] = deal(0);
    end

    %TUNING ------
    xi = 1;
    deltaT = 4;
    %-------------

    [phi0filt, phi0filt_dot, phi0filt_ddot] = ...
        ref_filter([phi0filt, phi0filt_dot, phi0filt_ddot]', phi0, xi, deltaT);

    phi_ref = phi_ref + phi0filt;
    phi_ref_dot = phi_ref_dot + phi0filt_dot;
    phi_ref_ddot = phi_ref_ddot + phi0filt_ddot;
end
```

4.4.6 Simulation results

The PD-heading controller, the LOS guidance and the reference filters can be implemented as a single module as in Code listing 4.9.

Code listing 4.9: PD-heading controller: `pd_heading`

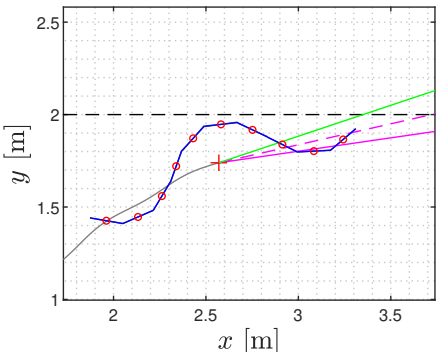
```
%% pd-heading LOS controller + filters ---------------------------------

function phi0 = pd_heading(t,x,yref)
    global Nl lookahead_dist;

    % initialization
    persistent thref_filt thref_dot thref_ddot;
    if isempty(thref_filt)
        [thref_filt, thref_dot, thref_ddot] = deal(0);
    end

    %TUNING ---------
    Kp = 0.3;
    Kd = 0.1;
    %----------------

```

```
    % LOS
    py = x(Nl+2);
    thref = -atan2(py - yref, lookahead_dist);

    %TUNING ------
    xi = 1;
    deltaT = 4;
    %-------------

    % theta_ref derivatives
    [thref_filt, thref_dot, thref_ddot] = ...
        ref_filter([thref_filt, thref_dot, thref_ddot]', thref, xi, deltaT);

    % control law
    heading = mean(x(1:Nl));
    heading_dot = mean(x(Nl+3:end-2));
    phi0 = Kp*(heading - thref_filt) + Kd*(heading_dot - thref_dot);
end
```

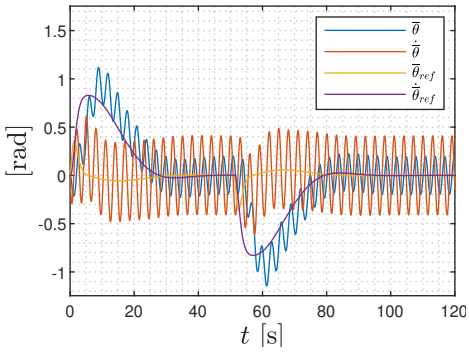
To validate the path-following scheme proposed we perform an experiment where the path is composed of two straight segments: $y_{ref} = 2$ [m] for $x < 10$ [m], while it abruptly change with a step to $y_{ref} = 0$ [m] for $x \geq 10$ [m]. We run the simulation for $T = 120$ [s], where the change occurs approximately at $t = 50$ [s]. Thanks to the reference filter introduced on the LOS module the step in the path doesn't translate into a step in the angle reference θ_{ref}, maintaining a smooth trajectory. The reference filter on the heading controller providing the derivatives of ϕ_0 also correct back the tracking issues faced before, as can be seen from Figures 4.17a and 4.17c. This translates into a correct behavior of the adaptive scheme for the body-shape control, whose behavior returns to the normal convergence to 0 of the gait-tracking errors and to the estimation errors.

4.4.7 Discussion

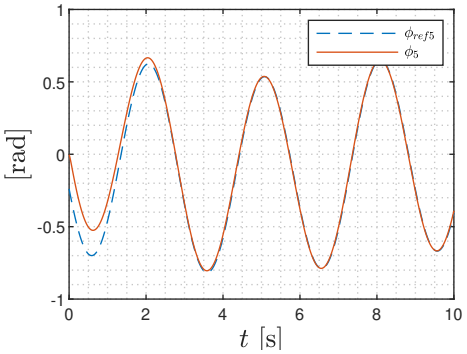
The complexity behind the *unactuated* heading, inherently linked to *actuated* joint angles, limits the design of control laws more sophisticated than a raw *PD-feedback control*, which, not being model-based results in a correction-only behavior, lacking of anticipating feed-forwards terms. This is exactly what motivates the introduction of the *control-oriented* model presented in the next chapter, whose simpler form allows a hierarchical decomposition in interconnected layers.



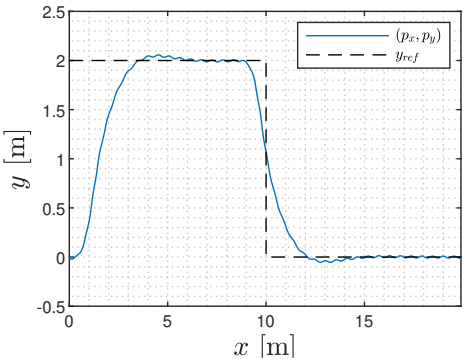
(a) Snapshot at $t = 20$ [s]. Please not the absence of joint tracking errors despite the constant perturbation through ϕ_0 and the smoothness of $\theta_{ref,filt}$ (dashed magenta) versus his unfiltered reference signal θ_{ref} (solid magenta).



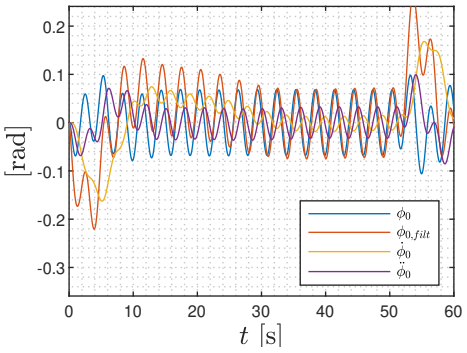
(b) Heading variables $\overline{\theta}, \dot{\overline{\theta}}$ vs their references. Note the oscillatory behavior around the reference, and also the little delay that appears to be present due to the *feedback-only* control.



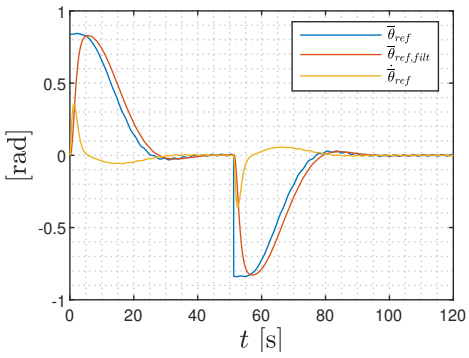
(c) Tracking performance on the 5^{th} joint angle within the time interval $t = [0-10]$ [s]. Please compare it with Figure 4.14c.



(d) Trace of the CM of the snake versus the path-reference. The turning anticipation of the step is due to the *look-ahead* distance chosen for the LOS guidance scheme.



(e) Time evolution of ϕ_0 and its derivatives. Note the little delay between the filtered reference and the unfiltered raw one. Zoom in $t = [0-60]$ [s].



(f) Time evolution of $\overline{\theta}_{ref}$ and its derivatives. The filter avoid the abrupt step at $x = 10$ [m] smoothing it out.

Figure 4.17: Simulation results on the experiment with $y_{ref} = 2$ [m] for $x < 10$ [m], $y_{ref} = 0$ [m] for $x \geq 10$ [m]. The robot initial condition are still set as $\mathbf{x}(0) = [0, \ldots, 0] \in \mathcal{R}^{2N+4}$. The heading controller gains are set as $k_p = 0.3$, $k_d = 0.1$, $\Delta = N \cdot 2l$. The terrain friction coefficients are instead $c_t, c_n = [0.5, 3]$ [Ns/m] and stays constant across the simulation. The filters parameters are set as $\Delta T = 2\pi/\omega_c = 4$ [s] and $\zeta = 1$ for both $\overline{\theta}_{ref}$ and ϕ_0.

Chapter 5

A simplified Control-Oriented Model

In this chapter, we present a simplified model for the snake robot, as derived in Liljebäck *et al.* [1, Chapter 6], which abstracts the *revolute* joints as *prismatic* – whose assumptions, details, and limitations are discussed in Section 5.1. This model admits a natural decomposition into a hierarchical cascaded structure, which is particularly well-suited for path-following control, while still retaining the *stabilizability* and *controllability* properties of the *complex model*.

We then present a partial feedback linearization control law for body-shape control in Section 5.2, followed by a model-based heading control scheme through a cascaded approach in Section 5.3, highlighting the substantial improvements over a PD-based heading controller.

In Section 5.4 the heading control law is further extended to an adaptive form with respect to the *rotational drag coefficients*, thereby addressing real-world scenarios where such parameters are unavailable (i.e., outside a simulation framework). To the best of our knowledge, no analytical method exists for selecting the rotational drag coefficients required to match the rotational motion of the *simplified* model with the of the *complex* one. In the literature, this is typically addressed through trial-and-error or data-driven approaches, comparing the behaviors of the two models until sufficient similarity is achieved.

Motivated by the goal of solving the full locomotion problem, we next develop a friction-adaptive body-shape controller also for the *CO-model* in Section 5.5, demonstrating the feasibility of a dual-layer hierarchical adaptation scheme: one for body-shape control and one for path-following.

5.1 Modeling

5.1.1 Overview of the approach

An attractive idea introduced in Liljebäck *et al.* [24] and comprehensively summarized in [1] is to base the controller synthesis on a *simplified* model of the snake robot avoiding the intricate expressions of the *complex model* in (4.12). Since the focus lies primarily on the *overall* locomotion, rather than the oscillatory behavior of heading and position, we may retain only the essential dynamic features relevant to forward motion, while preserving the same *stabilizability* and *controllability* properties of the *complex model*. Liljebäck *et al.* [1, Property 4.8] shows that lateral undulation is mainly driven by relative link displacements that are *transversal* to the direction of motion. These transversal effects dominate the contribution to forward propulsion, while the influence of tangential displacements remains secondary, provided that the relative joint angles are kept sufficiently small. Describing propulsion in terms of translational displacements between the CM of the links indeed leads to simpler equations of motion. Figure 5.1 illustrates the intuition underlying the model simplification.

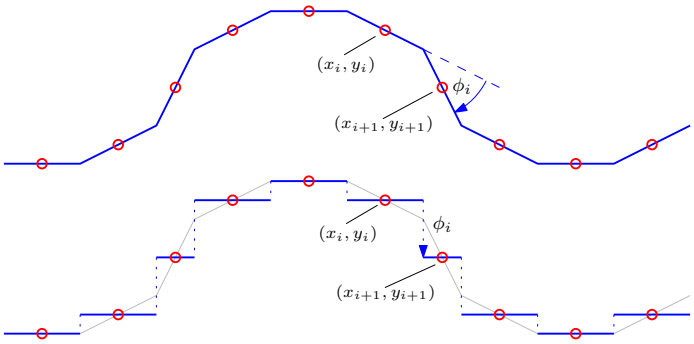


Figure 5.1: Model transformation approach

However, this approach neglects the rotational motion of the links during body-shape changes, which introduces some limitations in the derived model. These are fully discussed in Liljebäck *et al.* [1, Chapter 6]; here, we highlight the most relevant ones:

1. Consider the robot oriented along the global positive x-axis. The model retain its validity provided that the link angles θ_i of the complex model, which we seek to approximate with the *CO-model*, are kept small enough so that the prismatic joints represent a valid approximation of the rotational ones. In scenarios where the robot's heading deviates from the global positive x-axis, this can be generalized by similar reasoning on $\theta_i' = \theta_i - \overline{\theta}$, i.e. the angles that the links form with respect to the global orientation. This has to be kept in mind when one chooses the *gait pattern's* parameter α.
2. the model is only qualitatively similar, not quantitatively, and therefore is not intended as an accurate simulation model of the snake robot but only

as a base for controller design.

3. The rotational motion of a snake robot is determined by the linear and rotational velocities of the links. If the relative rotational link motion is disregarded, the resulting model cannot capture changes in orientation. To address this, the simplified model is extended with added dynamics inspired from a qualitative understanding of how rotational motion arises in the *complex model*. As discussed in Liljebäck *et al.* [1, Section 6.5.2], the rotational dynamics can be described as the sum of the torques induced by the average of the prismatic joint angles together with the forward velocity, and the one induced by viscous friction forces acting on the links. The coupling between these terms is scaled by two *artificial* rotational drag coefficients λ_1 and λ_2, acting as matching terms between the dynamics of the CO and revolute joint model. While this dynamical structure does not arise from first principles modeling, it has nevertheless been demonstrated in the literature that the CO model has utility for control design. With a proper choice of these parameters, Liljebäck *et al.* [1] conjecture that the simplified model is also *quantitatively* similar, a claim supported by the *stabilisability* and *controllability* analysis carried in [1, Sections 6.8–6.9].

5.1.2 Model Parameters and Kinematic

The snake robot is still described in the *simplified* model as series of N rigid links, each one of length l, but this time interconnected by $N-1$ *prismatic* actuated joints. Note that we denote the total link as l, whereas it was $2l$ in the complex model for notational convenience. All links has the same mass m and are modeled as zero-width links, and the actuators mass and inertia are negligible, assuming that the mass is uniformly distributed across the links such that his CM coincides with its geometrical center. The ground friction forces are still viscous and *anisotropic*, as discussed in Section 4.1 for the *complex* model.

The motion of the robot is described with respect to two different coordinate frames, shown in Figure 5.2a. The x-y frame is fixed to the global frame, while the t-n frame is aligned with the snakes orientation. In the t-n frame, the t and n axes point in the *tangential* and *normal* directions of the snake, respectively, and rotate as the snake changes its heading. The origins of both frames coincide.

The center of mass (CM) of the snake is located at (p_x, p_y) in the global frame and at (p_t, p_n) in the t-n frame. The global orientation, or heading, of the snake is denoted by θ and expressed with respect to the global x axis, with counterclockwise being positive. θ is always aligned with the forward direction of motion, providing an explicit representation of the snakes heading, in contrast to the heading definition $\overline{\theta}$ used in the complex model. The relationship between the two frames is given by

$$p_t = p_x \cos\theta + p_y \sin\theta \tag{5.1a}$$

$$p_n = -p_x \sin\theta + p_y \cos\theta \tag{5.1b}$$

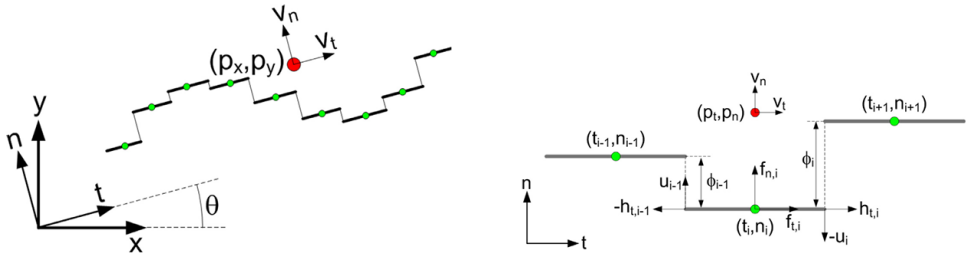


Figure 5.2: Illustration of the two coordinate frames used.

We denote the joint variable as ϕ_i, which now represents the normal-direction distance between links i and $i+1$, defined as

$$\phi_i = n_{i+1} - n_i, \tag{5.2}$$

where (t_i, n_i) is the position of the CM of link i in the t-n frame. Note that the joint variable is expressed with respect to link $i+1$ relative to link i, unlike in the *complex model*, where the convention was the opposite. This difference introduces a sign mismatch when porting controllers designed for the *CO model* to the *complex model*, requiring to flip the sign of the control law to ensure a correct implementation.

5.1.3 Complete Simplified Model

We express the complete simplified model in terms of the snake robot's state variable vector

$$\mathbf{x} = \left[\boldsymbol{\phi}^T, \theta, p_x, p_y, \mathbf{v}_\phi^T, v_\theta, v_t, v_n\right]^T \in \mathbb{R}^{2N+4}, \tag{5.3}$$

where $\boldsymbol{\phi} \in \mathbb{R}^{N-1}$ are the (translational) joint coordinates, $\theta \in \mathbb{R}$ is the absolute heading, or orientation, $(p_x, p_y) \in \mathbb{R}^2$ is the global position of the CM, $\boldsymbol{v}_\phi = \dot{\boldsymbol{\phi}} \in \mathbb{R}^{N-1}$ are the (translational) joint velocities, $v_\theta = \dot{\theta} \in \mathbb{R}$ is the angular heading velocity and $(v_t, v_n) \in \mathbb{R}^2$ are the tangential and normal direction velocities of the CM of the snake robot in the $t-n$ frame. The model derivations are omitted here for brevity, and we refer the curious reader to Liljebäck *et al.* [1, Section 6.3-6.6] for a detailed derivation of the model. The complete simplified model can be

written as

$$\dot{\boldsymbol{\phi}} = \mathbf{v}_\phi, \tag{5.4a}$$
$$\dot{\theta} = v_\theta, \tag{5.4b}$$
$$\dot{p}_x = v_t \cos\theta - v_n \sin\theta, \tag{5.4c}$$
$$\dot{p}_y = v_t \sin\theta + v_n \cos\theta, \tag{5.4d}$$
$$\dot{\mathbf{v}}_\phi = -\frac{c_n}{m}\mathbf{v}_\phi + \frac{c_p}{m} v_t \mathbf{A}\mathbf{D}^T \boldsymbol{\phi} + \frac{1}{m}\mathbf{D}\mathbf{D}^T \mathbf{u}, \tag{5.4e}$$
$$\dot{v}_\theta = -\lambda_1 v_\theta + \frac{\lambda_2}{N-1} v_t \bar{\mathbf{e}}^T \boldsymbol{\phi}, \tag{5.4f}$$
$$\dot{v}_t = -\frac{c_t}{m} v_t + \frac{2c_p}{Nm} v_n \bar{\mathbf{e}}^T \boldsymbol{\phi} - \frac{c_p}{Nm} \boldsymbol{\phi}^T \mathbf{A}\bar{\mathbf{D}}\mathbf{v}_\phi, \tag{5.4g}$$
$$\dot{v}_n = -\frac{c_n}{m} v_n + \frac{2c_p}{Nm} v_t \bar{\mathbf{e}}^T \boldsymbol{\phi}, \tag{5.4h}$$

where c_t, c_n are still the ground friction coefficient from the complex model, respectively in the tangential and normal direction, c_p is a newly introduced propulsion coefficient defined as

$$c_p = \frac{c_n - c_t}{2l}, \tag{5.5}$$

$\mathbf{u}$ is the vector of the actuation forces, λ_1 and λ_2 are the rotational drag coefficients characterizing the rotational dynamics of the snake, and the remaining quantities are defined as

$$\bar{\mathbf{e}} = [1, \ldots, 1]^T \in \mathbb{R}^{N-1}, \qquad \bar{\mathbf{D}} = \mathbf{D}^T \left(\mathbf{D}\mathbf{D}^T\right)^{-1} \in \mathbb{R}^{N\times(N-1)}. \tag{5.6}$$

An implementation of the model in (5.4) is shown in Code listing 5.1.

Code listing 5.1: Dynamics of the CO model of the snake robot: `dyn635`

```
%& dynamics using (6.35) CO-model

function dx = dyn635(x,u)
    global Nl m ct cn cp l1 l2;
    global A D D_bar e_bar nx;

    phi = x(1:Nl-1); theta = x(Nl);
    vphi = x(Nl+3:end-3); vtheta = x(end-2);
    vt = x(end-1); vn = x(end);

    dx = zeros(nx,1);
    dx(1:Nl-1) = vphi;
    dx(Nl) = vtheta;
    dx(Nl+1) = vt*cos(theta) - vn*sin(theta);
    dx(Nl+2) = vt*sin(theta) + vn*cos(theta);
    dx(Nl+3:end-3) = -cn/m * vphi + cp/m * vt * A*D' * phi + 1/m * (D*D')*u;
    dx(end-2) = -l1*vtheta + l2/(Nl-1) * vt * e_bar' * phi;
    dx(end-1) = -ct/m * vt + 2*cp/(Nl*m) * vn * e_bar' * phi + ...
                -cp/(Nl*m) * phi' * A*D_bar * vphi;
    dx(end) = -cn/m * vn + 2*cp/(Nl*m) * vt * e_bar' * phi;
end
```

5.2 Control

5.2.1 Body Shape Control

Similarly to what done for the complex model we can partially feedback linearize the $N-1$ *actuated dof* using the control law:

$$\mathbf{u} = m\left(\mathbf{D}\mathbf{D}^T\right)^{-1}\left(\overline{\mathbf{u}} + \frac{c_n}{m}\dot{\boldsymbol{\phi}} - \frac{c_p}{m}v_t\mathbf{A}\mathbf{D}^T\boldsymbol{\phi}\right), \tag{5.7a}$$

$$\overline{\mathbf{u}} = \ddot{\boldsymbol{\phi}}_{ref} - k_d(\dot{\boldsymbol{\phi}} - \dot{\boldsymbol{\phi}}_{ref}) - k_p(\boldsymbol{\phi} - \boldsymbol{\phi}_{ref}). \tag{5.7b}$$

with $k_p > 0, k_d > 0$ scalar controller gains. Equivalently as for the *complex model*, the proposed controller linearize the dynamics of the *actuated dof* with respect to the new set of control inputs $\overline{\mathbf{u}}$, yielding a joint error dynamics

$$\ddot{\tilde{\boldsymbol{\phi}}} + k_d\dot{\tilde{\boldsymbol{\phi}}} + k_p\tilde{\boldsymbol{\phi}} = \mathbf{0} \tag{5.8}$$

whose origin is *UGES* (see Proposition 1). The controller can be seen in Figure 5.3 depicted as a block-diagram scheme. The implementation of the controller can be found in Code listing 5.2.

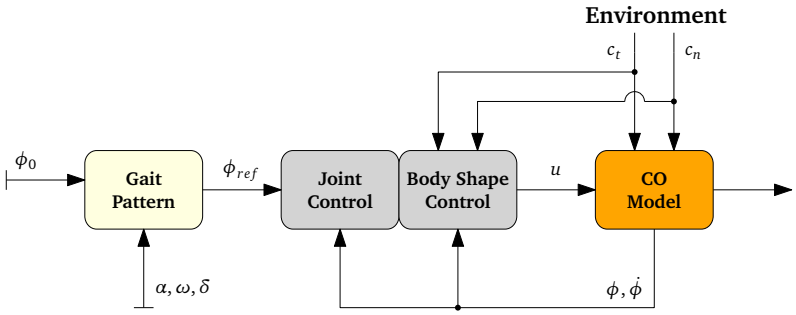


Figure 5.3: Block diagram of the partial feedback linearization body-shape controller for the CO-model. In the picture, the controller is divided into two blocks: Joint Control $\overline{\mathbf{u}}$ and Body Shape Control $\mathbf{u}$.

Code listing 5.2: Body-shape controller for the CO-model: `co_fblin`

```
%% co-fb linearizing controller

function u = co_fblin(x, phi_ref, phi_ref_dot, phi_ref_ddot)
    global Nl m cn cp A D

    %TUNING ---------
    Kp = 3; Kd = 1;
    %----------------

    phi = x(1:Nl-1);
    phi_dot = x(Nl+3:end-3);
    vt = x(end-1);

    %fb linearizing control law
```

```
15      mDDTinv = m*inv(D*D');
16      u_bar = phi_ref_ddot - Kd * (phi_dot - phi_ref_dot) - Kp * (phi - phi_ref);
17      u = mDDTinv * (u_bar + cn/m * phi_dot - cp/m * vt * A*D' * phi);
18  end
```

5.2.2 Choice of gait parameters

The reference gait pattern $\phi_{i,\text{ref}}$ remains essentially the same as for the *complex model*, but its parameters has to be translated for the simplified model. It's important that the joint angles θ_i of the approximated *revolute* joint model remains sufficiently small to ensure the validity of the assumptions made in the derivation of the *simplified* model. [1] demonstrates that this holds as long as the link angles θ_i are kept smaller than 20°. Since input torques $\mathbf{u}$ are now replaced by linear forces the amplitude α of the sinusoidal gait must be adjusted and translated accordingly to not violate the assumptions. In order to satisfy the constraint we simulate the *complex model* to determine the α_{max} satisfying $\max|\theta_i| \leq 20°$, obtaining for the current setup and model parameters in use (see Section 4.2.3) an $\alpha_{max} \simeq 16°$, which corresponds to a maximum link vertical displacement of $\Delta y_{\max} = \max y_i - \min y_i \simeq 12.45$ [cm]. The *linear* amplitude α'_{max} in the simplified model is then chosen to reproduce the same $\Delta y_{\max}$, yielding to $\alpha'_{max} \simeq 5.2$ [cm], thus ensuring consistent behavior between the models.

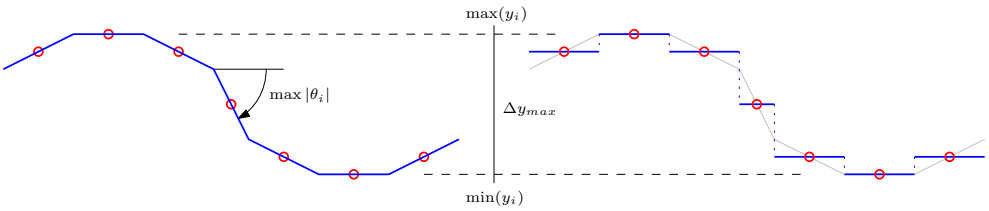


Figure 5.4: Experiment taken to determine the maximum gait parameter α that respects the CO-model assumptions.

5.3 Path-following Control through a Cascaded Approach

In this section we return to face the problem of path-following, but this time developing a heading controller which is *model-based* on the simplified model of the snake robot. The steering inherent mechanism once again relies on the parameter ϕ_0 of the *gait pattern* reference, while the controller itself follows the design proposed in Liljebäck *et al.* [1, Chapter 8]. We first briefly introduce the underlying assumptions in Section 5.3.1 and present its derivation in Section 5.3.2, as it serves as a fundamental basis for the adaptive heading controller introduced in the next section. Finally, we validate the approach through numerical simulations, after some minor, but necessary, adjustments of the reference filter settings, which is discussed in Section 5.3.3, and demonstrate its superior performance compared to a raw PD heading control law in Section 5.3.4.

5.3.1 Preliminaries

The path-following controller presented next rely on a number of assumptions. These assumptions are not strictly necessary for the understanding of the material presented here and the design of the controller itself, but rather useful for the stability analysis covered in [1], and so left out from this work. We briefly summarize them below for completeness.

Assumption 1 (Forward velocity)**.** During motion the forward velocity v_t oscillates around a non-zero average velocity that can be predetermined from the gait pattern parameters, i.e.,

$$v_t \in [V_{\min}, V_{\max}] \geq 0, \quad \forall t \geq 0.$$

Assumption 1 is generally valid, except at the initial instant $t = 0$ [s], when the snake starts from rest ($v_t = 0$) and during the initial transient of motion. In practice, it's sufficient to activate the heading and path-following controllers once $v_t > 0$.

Assumption 2 (Model transformation)**.** The joint coordinates $\boldsymbol{\phi}$ appear in the dynamics of both the angular velocity v_θ and the lateral velocity v_n. This coupling complicates the controller design, as body shape changes both affect heading and sideways motion. It's possible to virtually shift the CM backwards by a distance ϵ along the t-axis, so that body shape changes induce pure rotational motion and no sideways forces.

The coordinate transformation mentioned in assumption 2, and illustrated in [1, Figure 8.1], only modifies the dynamics of v_n, which is not object of control and therefore omitted. For details, see [1, Section 8.3.3].

Assumption 3 (Look-ahead distance)**.** Provided that assumptions 1 and 2 holds and that the look-ahead distance Δ of the LOS module is selected big enough according to [1, Theorem 8.3], the model and the controller can be written as a cascaded system where the body shape changes $\boldsymbol{\phi}$ affect the global orientation of the robot θ, which subsequently affects the cross-track error $\overline{p_y} = p_y - y_{ref}$ between the robot and the desired path. Furthermore, the heading controller in combination with the body shape controller in Equation (5.7) guarantees to $\mathcal{K}$-exponentially stabilize the snake robot to any desired straight path (see Liljebäck *et al.* [1, Definition 8.5] for the definition and [1, Sections 8.5.5–8.5.6] for the proof).

From a numerical standpoint, the bound of assumption 3 is rather weak and can be satisfied without difficulty. In fact, the previously selected look-ahead distance $\Delta = N \cdot l$, chosen as the fully outstretched length of the snake, lies well above the estimated lower limit.

5.3.2 CO Heading Control

The heading controller proposed in Section 4.4 did not attempt to suppress the oscillatory behavior of the heading and position of the snake robot. This time the simplified model allows the derivation of a *model-based* heading controller, able to not only regulate the heading and cross-track error so that they oscillate around the reference, but also so that they converge to zero. The LOS module is the same employed for the complex model in Section 4.4.3, and it's defined as

$$\boxed{\overline{\theta}_{ref} = -\arctan\left(\frac{\overline{p_y}}{\Delta}\right),} \tag{5.9}$$

where $\overline{p_y} = p_y - y_{ref}$ denotes the cross-track error and Δ is the *look-ahead distance* previously mentioned.

We will again employ the joint offset ϕ_0 to make the snake robot heading θ tracks the reference angle θ_{ref} given by the LOS module. To derive the control law for ϕ_0 we begin by rewriting the dynamics of v_θ by "unpacking" ϕ_0 from the references $\boldsymbol{\phi}_{\text{ref}}$. By defining $\overline{\phi_{i,ref}}$ as the sinusoidal contribution of the reference such that

$$\phi_{\text{i,ref}} = \overline{\phi_{\text{i,ref}}} + \phi_0 = \alpha \sin(\omega t + (i-1)\delta) + \phi_0 \tag{5.10}$$

we can rewrite (5.4f) as

$$\dot{v}_\theta = -\lambda_1 v_\theta + \frac{\lambda_2}{N-1} v_t \overline{\mathbf{e}}^T \boldsymbol{\phi} \tag{5.11a}$$

$$= -\lambda_1 v_\theta + \frac{\lambda_2}{N-1} v_t \overline{\mathbf{e}}^T \left(\overline{\boldsymbol{\phi}_{\text{ref}}} + [\phi_0, \ldots, \phi_0]^T + \widetilde{\boldsymbol{\phi}}\right) \tag{5.11b}$$

$$= -\lambda_1 v_\theta + \lambda_2 v_t \phi_0 + \frac{\lambda_2}{N-1} v_t \overline{\mathbf{e}}^T \left(\overline{\boldsymbol{\phi}_{\text{ref}}} + \widetilde{\boldsymbol{\phi}}\right). \tag{5.11c}$$

Choosing ϕ_0 as

$$\boxed{\phi_0 = \frac{1}{\lambda_2 v_t}(\ddot{\theta}_{ref} + \lambda_1 \dot{\theta}_{ref} - k_\theta \widetilde{\theta}) - \frac{\overline{\mathbf{e}}^T \overline{\boldsymbol{\phi}_{\text{ref}}}}{N-1}} \tag{5.12}$$

where $\widetilde{\theta} = \theta - \theta_{ref}$ is the heading error and k_θ is a controller gain, yields to the heading dynamics:

$$\dot{v}_\theta = -\lambda_1 v_\theta + \ddot{\theta}_{ref} + \lambda_1 \dot{\theta}_{ref} - k_\theta \widetilde{\theta} + \frac{\lambda_2}{N-1} v_t \overline{\mathbf{e}}^T \widetilde{\boldsymbol{\phi}} \tag{5.13a}$$

$$\ddot{\theta} = -\lambda_1 \dot{\theta} + \ddot{\theta}_{ref} + \lambda_1 \dot{\theta}_{ref} - k_\theta \widetilde{\theta} + \frac{\lambda_2}{N-1} v_t \overline{\mathbf{e}}^T \widetilde{\boldsymbol{\phi}} \tag{5.13b}$$

$$\ddot{\theta} - \ddot{\theta}_{ref} + \lambda_1 \dot{\theta} - \lambda_1 \dot{\theta}_{ref} + k_\theta \widetilde{\theta} = \frac{\lambda_2}{N-1} v_t \overline{\mathbf{e}}^T \widetilde{\boldsymbol{\phi}} \tag{5.13c}$$

$$\boxed{\dddot{\widetilde{\theta}} + \lambda_1 \ddot{\widetilde{\theta}} + k_\theta \dot{\widetilde{\theta}} = \frac{\lambda_2}{N-1} v_t \overline{\mathbf{e}}^T \dot{\widetilde{\boldsymbol{\phi}}}} \tag{5.13d}$$

The error dynamics of the heading θ in Equation (5.13d) and the error dynamics of the joint variables in Equation (5.8) represent a cascaded system, where the inner joint subsystem perturbs the outer heading subsystem trough the interconnection signal $\frac{\lambda_2}{N-1} v_t \overline{\mathbf{e}}^T \widetilde{\boldsymbol{\phi}}$. As anticipated before, the origin of the cascaded system is globally $\mathcal{K}$-exponentially stable if assumptions 1, 2 and 3 holds. Since the control law depends on the inverse of the forward velocity v_t, to avoid the singularity at $v_t = 0$ it is sufficient to activate the controller once the robot gained a strictly positive velocity. A block-scheme diagram of the controller is illustrated in Figure 5.5.

The derivatives $\dot{\phi}_0, \ddot{\phi}_0$ needed for the calculation of the body shape control input in (5.7), and the derivatives $\dot{\theta}_{ref}, \ddot{\theta}_{ref}$ needed for the calculation of the heading control input in (5.12), can be obtained by passing ϕ_0 and θ_{ref} trough a 3rd-order low-pass reference filter, exactly as done for the *complex model*.

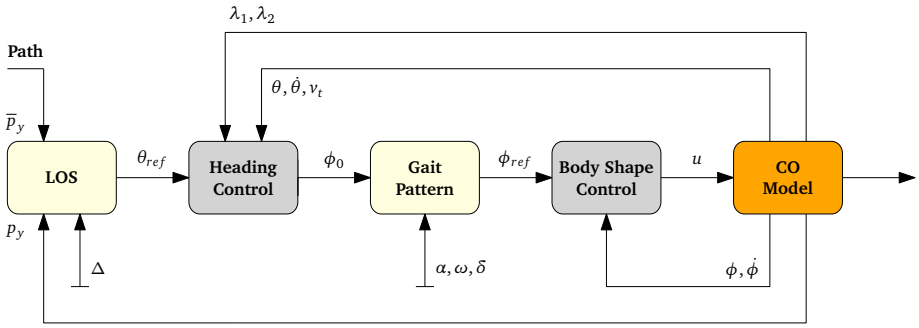


Figure 5.5: Path-following control architecture, comprehensive of the inner partial feedback linearization body-shape controller in (5.7) and the outer heading controller in (5.12). For simplicity, the reference filters and the dependence of the body-shape controller from the terrain friction coefficients are omitted from the picture.

5.3.3 A Hybrid filtering strategy

When implementing the overall control architecture in `MATLAB`, the current settings of the filters for ϕ_0 must be reconsidered. The proper balance previously found for the complex model is no longer valid, due to the new presence of a fast-varying term $\frac{\overline{\mathbf{e}}^T \overline{\boldsymbol{\phi}}_{\text{ref}}}{N-1}$ in the heading control law of ϕ_0.

Since the filters have a finite response time, the contribution of this term in the filtered derivatives $\dot{\phi}_0$ and $\ddot{\phi}_0$ gets almost entirely canceled leading to a consistent numerical error in the derivatives which, from numerical experiments, has been seen to not be negligible. This is in accordance to the layered structure of the cascaded system: the inner one is specified in terms of the joint dynamics, which is faster, while the outer one is specified in terms of the heading dynamics: which is slower. Increasing the filter responsiveness to accommodate this fast-varying component is not a viable solution, as it will produce excessively large derivatives,

leading to unrealistic control efforts and, in practical applications, input saturation and torque chattering.

A straightforward solution is to exclude this term from being filtered at all and instead use its exact derivative, which can be computed analytically given the knowledge of $\dot{\overline{\boldsymbol{\phi}}}_{\text{ref}}$ and $\ddot{\overline{\boldsymbol{\phi}}}_{\text{ref}}$. In other words, denoting the heading control law in (5.12) as

$$\phi_0 = \underbrace{\frac{1}{\lambda_2 v_t}(\ddot{\theta}_{ref} + \lambda_1 \dot{\theta}_{ref} - k_\theta \tilde{\theta})}_{\text{filtered} \rightarrow \dot{\phi}_0, \ddot{\phi}_0} \underbrace{- \frac{\bar{\mathbf{e}}^T \overline{\boldsymbol{\phi}}_{\text{ref}}}{N-1}}_{\overline{\phi_0}} \tag{5.14}$$

only the first part is passed through the reference filter, since the heading variable terms $\theta, \dot{\theta}$ and the forward velocity v_t doesn't represent a problem due to their slower dynamics, and the LOS reference angles $\theta_{ref}, \dot{\theta}_{ref}, \ddot{\theta}_{ref}$ are obtained by a reference filter with similar settings and responsiveness. The contribution to the derivatives of ϕ_0 of the second term is instead calculated exactly as

$$\dot{\overline{\phi_0}} = -\frac{\bar{\mathbf{e}}^T \dot{\overline{\boldsymbol{\phi}}}_{\text{ref}}}{N-1}, \qquad \ddot{\overline{\phi_0}} = -\frac{\bar{\mathbf{e}}^T \ddot{\overline{\boldsymbol{\phi}}}_{\text{ref}}}{N-1} \tag{5.15}$$

where we recall that $\overline{\boldsymbol{\phi}}_{\text{ref}}$ is defined as the sinusoidal contribution of the reference so that $\overline{\phi_{i,ref}} = g(i,N)\, \alpha \sin(\omega t + (i-1)\delta)$ without the ϕ_0 contribution. Finally the filtered part and the exact part of the derivatives of ϕ_0 are re-added together, which in turn is added to the sinusoidal contribution $\overline{\boldsymbol{\phi}}_{\text{ref}}$ to get the final references to be passed to the body shape controller. The process is illustrated in Figure 5.6 as a block-scheme diagram.

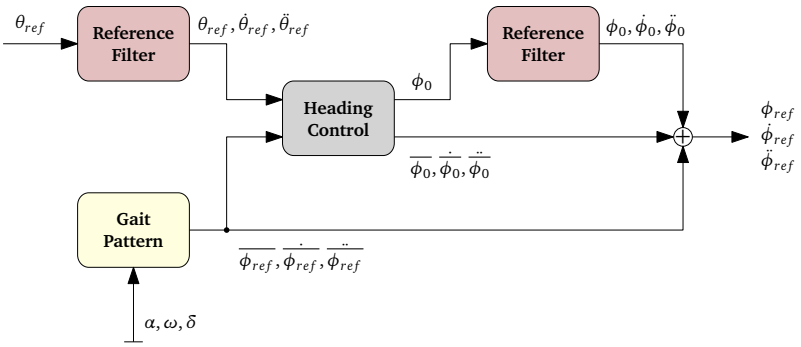


Figure 5.6: Block scheme diagram of the improvement on the structure of the 3rd-order low-pass reference filter for the calculation of $\dot{\phi}_0, \ddot{\phi}_0$

5.3.4 Simulation results

The heading controller and the adjustment taken on the filters for ϕ_0 can be implemented as in Code listings 5.3 and 5.4.

Code listing 5.3: Cascaded heading controller: `casc_heading`

```
%% cascaded heading controller -------------------------

function [phi0, phi0_bar] = casc_heading(t,x,yref)
    global Nl lookahead_dist ref;
    global l1 l2 vt_min e_bar ct cn cp;

    % initialization
    persistent thref_filt thref_dot thref_ddot;
    if isempty(thref_filt)
        [thref_filt, thref_dot, thref_ddot] = deal(0);
    end

    %TUNING ---------
    ktheta = 0.05;
    %----------------

    % LOS
    py = x(Nl+2);
    thref = -atan2(py - yref, lookahead_dist);

    %TUNING ------
    xi = 1;
    deltaT = 4;
    %-------------

    % theta_ref derivatives
    [thref_filt, thref_dot, thref_ddot] = ...
        ref_filter([thref_filt, thref_dot, thref_ddot]', thref, xi, deltaT);

    heading = x(Nl);
    heading_dot = x(end-2);
    vt = x(end-1);

    % control law
    phi0 = 1/(vt*l2) * ...
        (thref_ddot + l1*thref_dot - ktheta*(heading - thref_filt));
    % but
    if vt <= vt_min
        phi0 = 0;
    end

    sat = Nl/(2*(Nl-1))*sqrt(ct*cn)/cp; %7.25
    phi0 = max(-sat, min(phi0, sat));

    [phiref_bar, phiref_bar_dot, phiref_bar_ddot] = ref(t);
    phi0_bar(1) = -e_bar'*phiref_bar/(Nl-1);        %phi0_bar
    phi0_bar(2) = -e_bar'*phiref_bar_dot/(Nl-1);    %phi0_bar_dot
    phi0_bar(3) = -e_bar'*phiref_bar_ddot/(Nl-1);   %phi0_bar_ddot
end
```

Code listing 5.4: Hybrid reference filter: `hybrid_add_phi0`

```
% phi0, phi0_dot, phi0_ddot ---------------------------------

function [phi_ref, phi_ref_dot, phi_ref_ddot] = ...
    hybrid_add_phi0(phi_ref, phi_ref_dot, phi_ref_ddot, phi0, phi0_bar)

    % initialization
```

```
    persistent phi0filt phi0filt_dot phi0filt_ddot;
    if isempty(phi0filt)
        [phi0filt, phi0filt_dot, phi0filt_ddot] = deal(0);
    end

    %TUNING ------
    xi = 1;
    deltaT = 0.1; %0.1-0.3 with phi0filt
    %-------------

    [phi0filt, phi0filt_dot, phi0filt_ddot] = ...
        ref_filter([phi0filt, phi0filt_dot, phi0filt_ddot]', phi0, xi, deltaT);

    phi_ref = phi_ref + phi0filt + phi0_bar(1);
    phi_ref_dot = phi_ref_dot + phi0filt_dot + phi0_bar(2);
    phi_ref_ddot = phi_ref_ddot + phi0filt_ddot + phi0_bar(3);
end
```

To validate the performance of the path-following scheme proposed we compare it to a PD-heading control law of the same form of the one used for the *complex model*, i.e.

$$\phi_0 = -k_p\tilde{\theta} - k_d\dot{\tilde{\theta}}.$$

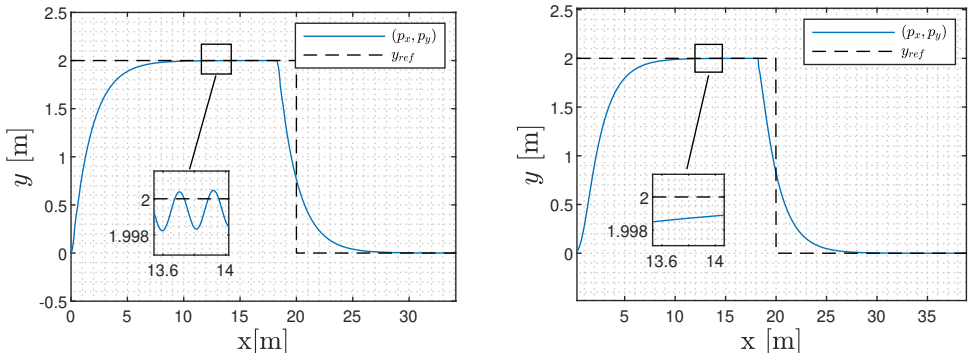
(Note the traditional *negative feedback* sign as for the simplified model the ϕ_0 contribution to the joint variable is concordant). We compare them on a piecewise straight path where $y_{ref} = 2\,[\mathrm{m}]$ for $x < 20\,[\mathrm{m}]$, and $y_{ref} = 0\,[\mathrm{m}]$ for $x \geq 20\,[\mathrm{m}]$. The *rotational drag coefficients* of the CO-model (and of the heading controller) are set as $\lambda_1 = 0.5$ and $\lambda_2 = 20$, as used in [1]. We run the simulation for $T = 480$ [s], since reducing the amplitude of the *gait pattern* to $\alpha' = 5.2$ [cm] to retain the validity of the model leads to a much slower motion. The robot initial condition are still set as $\mathbf{x}(0) = [0, \ldots, 0] \in \mathcal{R}^{2N+4}$. The PD-heading controller gains are set as $k_p = 0.4,\ k_d = 0.1$, while for our controller is set as $k_\theta = 0.05$. The look-ahead distance is set for both as the fully outstretched length of the snake $\Delta = N \cdot l$. The terrain friction coefficients are again $c_t, c_n = [0.5, 3]$ [Ns/m] and stays constant across the simulation. The filters parameters are set as $\omega = 4$ [rad/s] and $\zeta = 1$ for both the reference filters.

Furthermore, as suggested by [3, Eq. (82)], we saturate the control input ϕ_0 of both controllers to

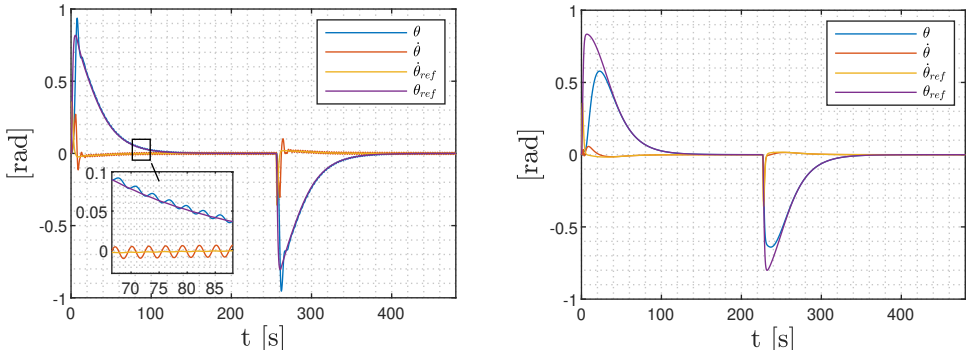
$$|\phi_0| < \frac{N}{2(N-1)}\frac{\sqrt{c_n c_t}}{c_p}. \tag{5.16}$$

This ensures that the average velocity converges exponentially to the steady-state value assumed in (1) of Section 5.3.1, while also preventing the singularity at $v_t = 0$. Nevertheless, although the singularity is avoided, excessively large control inputs may still occur at the very initial instants. To address this issue, we impose an additional threshold condition $v_t > v_{t,\min}$ below which the heading controller is not activated. In the experiments, this threshold is set to $v_{t,\min} = 2$ [cm/s]. The results are shown in Figure 5.7.

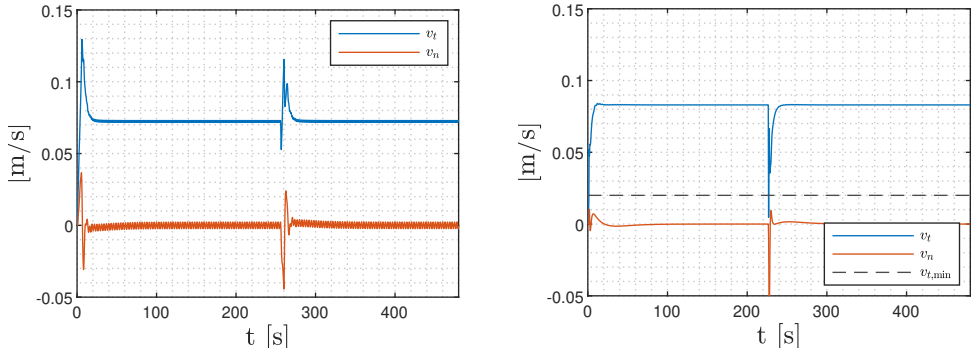
At first glance, the performance appears very similar; however, a closer inspection to Figure 5.7a reveals a noticeable difference: the PD-heading controller is



(a) Trace of the CM of the snake (p_x, p_y) versus to the path reference y_{ref}.



(b) Heading variables $\theta, \dot{\theta}$ vs the LOS reference angles θ_{ref} and $\dot{\theta}_{ref}$.



(c) Time evolution of the tangential velocity v_t and normal velocity v_n during the experiment. On the right in black it's also depicted the threshold $v_{t,\min}$ under which the Heading controller doesn't run.

Figure 5.7: Comparison of a PD-heading control law (on the left) and the cascaded heading controller developed for the CO-model (on the right).

unable to eliminate the oscillatory behavior – although small in this case, it remains present. Its limited magnitude is only due to the chosen gait parameters; increasing these would amplify the undesired effects. In Figure 5.7b it's noticeable the smoother behavior and absence of spikes in the heading variables when using the proposed controller, thanks to the anticipating *feed-forward* terms. This also reflects in a smoother behavior of the velocities, shown in Figure 5.7c (except for the spike during the step of y_{ref}).

5.4 Adaptive Heading Control

The heading control law presented in (5.12) relies on precise knowledge of the *rotational drag coefficients* λ_1 and λ_2, which poses challenges for practical implementation. Determining these coefficients so that the simplified CO-model matches the complex *revolut* model is a problem that has so far remained largely unsolved in the literature, which is traditionally solved by treating λ_1 and λ_2 as controller tuning parameters. The absence of a systematic way to determine whether the tuning of the controller is optimal represents a gap, which we aim to address by introducing an adaptive framework to estimate these *artificial* coefficients, instead of tuning them. Under ideal conditions, this approach constitutes a crucial step towards maximizing the model matching and consequently the effectiveness of the CO-based heading controller for practical implementations in real-world scenarios.

5.4.1 Approach

Analogously to the adaptive body shape controller, (5.12) is not suitable for adaptive control, since the *rotational drag coefficients* λ_1 and λ_2 do not appear in linearly separable form. Instead, they enter the expression through a more complicated multiplicative relationship involving both coefficients, which prevents the direct implementation of the law using their estimates.

In the following we present a reformulation of λ_1 and λ_2, along with an input transformation of our virtual control input ϕ_0 that enable the closed loop dynamics of the heading subsystem in (5.11c) to be reformulated as in Slotine and Li [25, Chapter 9.2.1], so that it becomes directly suitable for the application of their adaptive control methods. In their framework, the system dynamics is expressed as the product of a regression matrix Y – constructed from the state variables and reference signals – and a vector of the uncertain parameters $\mathbf{a}$. Although the dynamics are nonlinear in the states, this representation renders them linearly parameterized with respect to the uncertainties, which in turn enables the systematic design of adaptation laws based on Lyapunov theory.

5.4.2 Method

We recall that the open loop dynamics of the heading subsystem in (5.11c) is

$$\dot{v}_\theta = -\lambda_1 v_\theta + \lambda_2 v_t \phi_0 + \frac{\lambda_2}{N-1} v_t \overline{\mathbf{e}}^T \left(\overline{\boldsymbol{\phi}_{\text{ref}}} + \widetilde{\boldsymbol{\phi}} \right).$$

The last term corresponds to what we previously denoted as the *perturbation term* in the closed-loop heading dynamics. Retaining this term in the derivation is problematic, since any control law for ϕ_0 that explicitly incorporates it would render ϕ_0 an implicit function of itself, thereby making the formulation mathematically inconsistent.

We therefore conjecture that we can disregard this term in the subsequent analysis because the body-shape controller is designed to ensure convergence of the joint tracking error $\widetilde{\boldsymbol{\phi}}$, to zero. Consequently, the perturbation term, which is directly related to this error, will also converge to zero at the same rate.

Also, it's always possible to activate the heading controller only after the joint tracking error has fallen below an acceptable threshold $\|\widetilde{\boldsymbol{\phi}}\|_{\min}$, under which the perturbation is considered sufficiently small to not significantly affect the heading dynamics. Once the joint tracking error instead has converged, this approximation becomes exact, and the resulting derivation does not involve any loss of generality. We therefore make the following assumption:

Assumption 4 (Perturbation term)**.** The perturbation term in the open-loop dynamics of the heading subsystem can be neglected, since the body-shape controller guarantees convergence of $\widetilde{\boldsymbol{\phi}} \to \mathbf{0}$ to zero. This allows us to treat the model of the snake robot as a cascade[1], where the body-shape controller acts as a follower to the heading subsystem within the control chain.

Following Assumption 4 and rewriting v_θ as $\dot{\theta}$, we're left with

$$\ddot{\theta} = -\lambda_1 \dot{\theta} + \lambda_2 v_t \phi_0 + \frac{\lambda_2}{N-1} v_t \overline{\mathbf{e}}^T \overline{\boldsymbol{\phi}_{\text{ref}}} + \underbrace{\frac{\lambda_2}{N-1} v_t \overline{\mathbf{e}}^T \widetilde{\boldsymbol{\phi}}}_{\text{perturbation term}} \tag{5.17a}$$

$$= -\lambda_1 \dot{\theta} + \lambda_2 v_t \phi_0 + \frac{\lambda_2}{N-1} v_t \overline{\mathbf{e}}^T \overline{\boldsymbol{\phi}_{\text{ref}}}. \tag{5.17b}$$

In order to proceed with the Lyapunov design of the controller, we introduce an input transformation for ϕ_0, choosing it as

$$\phi_0 = \frac{1}{v_t} \left(\overline{u}_{\phi_0} - \frac{\overline{\mathbf{e}}^T \overline{\boldsymbol{\phi}_{\text{ref}}}}{N-1} \right), \tag{5.18}$$

[1] Strictly speaking, this is not exactly true, since the CM coordinate p_y is fed back to the LOS guidance module, effectively closing the control loop (see, for example, Figure 5.5). However, because p_y evolves with very slow dynamics compared to the rest of the system, it can be treated as a exogenous reference signal. This perspective allows us to regard the overall control architecture as a cascade, with the LOS guidance at the top level, followed by the heading controller, and finally the body-shape controller.

where $\overline{u}_{\phi_0}$ is a new control input to be designed at a later stage. Substituting this transformation into the system yields the closed-loop dynamics

$$\ddot{\theta} = -\lambda_1 \dot{\theta} + \lambda_2 \overline{u}_{\phi_0}, \tag{5.19a}$$

which is linear in the *drag coefficients* but still involves a multiplicative factor with the control input $\overline{u}_{\phi_0}$, making it unsuitable for a classical adaptive framework. To address this, we introduce a transformation of the coefficients by first dividing both sides of the equation by λ_2, which is always possible since $\lambda_2 > 0$ by model assumptions. This gives

$$\underbrace{\frac{1}{\lambda_2}}_{a_1} \ddot{\theta} = -\underbrace{\frac{\lambda_1}{\lambda_2}}_{a_2} \dot{\theta} + \overline{u}_{\phi_0}, \tag{5.20a}$$

$$a_1 \ddot{\theta} = -a_2 \dot{\theta} + \overline{u}_{\phi_0}, \tag{5.20b}$$

which is now linearly parameterized in the newly defined *drag parameters* a_1 and a_2, and no longer presents any multiplicative factor on the new control input $\overline{u}_{\phi_0}$.

The system is now in a suitable form for using the systematic approach introduced by Slotine and Li [25, Chapter 9.2.1]. Following Slotine and Li [25] we introduce an output variable s, also called *filtered tracking error*, defined as

$$\boxed{s = \dot{\widetilde{\theta}} + \Lambda \widetilde{\theta},} \tag{5.21}$$

where $\widetilde{\theta} = \theta - \theta_{ref}$ is the heading tracking error, $\dot{\widetilde{\theta}}$ is the angular velocity tracking error, and $\Lambda > 0$ is a tuning parameter that shapes the dynamics of the virtual first-order system in (5.21).

The introduction of s allows the second-order error dynamics in $(\widetilde{\theta}, \dot{\widetilde{\theta}})$ to be expressed as a first-order virtual system in s, simplifying both Lyapunov design and analysis. In this formulation, s plays the role of a disturbance input acting on the virtual system, so that designing a control law that drives $s \to 0$ guarantees $(\widetilde{\theta}, \dot{\widetilde{\theta}}) \to \mathbf{0}$. This is intuitively explained by the fact that $s = 0$ reduce the heading error dynamics to

$$\dot{\widetilde{\theta}} + \Lambda \widetilde{\theta} = 0, \tag{5.22}$$

which is a first-order *UGES* linear system. This implies that, once $s = 0$, the heading tracking errors satisfy $\widetilde{\theta} \to 0$ and $\dot{\widetilde{\theta}} \to 0$ exponentially. Consequently, the heading dynamics inherit the weaker of the two properties between *UGES* stability or the one achieved for s. The proof is formally carried out in Slotine and Li [25, Chapter 7.1.1], to which we refer the curious reader. We also define a *filtered reference trajectory*

$$\dot{\theta}_r = \dot{\theta}_{ref} - \Lambda \widetilde{\theta}, \tag{5.23}$$

which can be interpreted as an *augmented velocity reference trajectory*. This allows us to rewrite s as

$$s = \dot{\widetilde{\theta}} + \Lambda\widetilde{\theta} = \dot{\theta} \underbrace{-\dot{\theta}_{ref} + \Lambda\widetilde{\theta}}_{-\dot{\theta}_r} = \dot{\theta} - \dot{\theta}_r, \tag{5.24}$$

which gives an intuitive expression for s as a *velocity tracking error* with respect the augmented velocity reference trajectory $\dot{\theta}_r$. This term allows us to express the system dynamics in terms of s linearly in the unknown parameters. Indeed, rewriting the system dynamics from (5.20b) in terms of s we have that

$$a_1\dot{s} = \underbrace{a_1(\ddot{\theta}}_{(5.20b)} - \ddot{\theta}_r) = -a_2\dot{\theta} + \overline{u}_{\phi_0} - a_1\ddot{\theta}_r \tag{5.25a}$$

$$= -a_1\ddot{\theta}_r - a_2\dot{\theta} + \overline{u}_{\phi_0} \tag{5.25b}$$

$$= -\underbrace{[\ddot{\theta}_r \ \dot{\theta}]}_{:=Y}\underbrace{[a_1 \ a_2]}_{\mathbf{a}} + \overline{u}_{\phi_0} = -Y\mathbf{a} + \overline{u}_{\phi_0}, \tag{5.25c}$$

defining $Y \triangleq [\ddot{\theta}_r \ \dot{\theta}]$ as the regression matrix (its name originates from adaptive control and will be clearer later) and $\mathbf{a} = [a_1 \ a_2]^T$ as the vector of the *drag parameters*.

We now present the derivation of the adaptive controller in two steps, following their methodology. First, we design the control law assuming perfect knowledge of the *drag parameters*. Then we replace the true parameters with their estimates $\hat{\mathbf{a}}$ in the control law and design an adaptive law to guarantee (i) Lyapunov stability of the overall adaptive scheme and (ii) tracking of the heading reference angles.

5.4.3 Lyapunov Design

Proposition 5. Assume perfect knowledge of the *drag parameters* $\mathbf{a} = [a_1, \ a_2]^T$. Then, under the control law $\overline{u}_{\phi_0} = Y\mathbf{a} - k_d s$ the origin $s = 0$ of the heading error subsystem in (5.25c) is *GUAS*, where the terms in the control law are defined as

$$s = \dot{\widetilde{\theta}} + \Lambda\widetilde{\theta}, \quad \text{the filtered tracking error} \tag{5.26a}$$

$$Y = [\ddot{\theta}_r \ \dot{\theta}_r], \quad \text{the regression matrix} \tag{5.26b}$$

$$\dot{\theta}_r = \dot{\theta}_{ref} - \Lambda\widetilde{\theta}, \quad \text{the reference trajectory} \tag{5.26c}$$

$$k_d > 0, \quad \Lambda > 0, \quad \text{tuning values} \tag{5.26d}$$

Proof. Consider the *LFC* for the system in (5.25c):

$$\boxed{V_1 = \frac{1}{2}a_1 s^2,} \tag{5.27}$$

where $a_1 > 0$ is the first of the two *drag parameters*. Differentiating V_1 and substituting for the system dynamics in (5.25c) yields

$$\dot{V}_1(s) = a_1 s\dot{s} = s \cdot a_1\dot{s} \tag{5.28a}$$
$$= s(-Y\mathbf{a} + \overline{u}_{\phi_0}), \tag{5.28b}$$

Selecting the control law

$$\boxed{\overline{u}_{\phi_0} = Y\mathbf{a} - k_d s, \qquad k_d > 0,} \tag{5.29}$$

and inserting it into (5.28b) gives

$$\dot{V}_1(s) = s(-Y\mathbf{a} + Y\mathbf{a} - k_d s) = -k_d s^2 \leq \underbrace{-k_d|s|^2}_{-W_3(s)<0} < 0. \tag{5.30a}$$

V_1 in (5.27) is *positive definite, radially unbounded* and independent from time and so it can be easily bounded by

$$\underbrace{V_1(s)}_{W_1(x)} \leq V_1(s,t) \leq \underbrace{V_1(s)}_{W_2(x)}. \tag{5.31}$$

Moreover, since $\dot{V}_1$ is upper bounded by the negative definite function $-W_3(s)$, it follows from Theorem 6 that the origin $s = 0$ is *GUAS*. □

Proposition 5 requires perfect knowledge of $\mathbf{a}$, which is not available in practice. We can however still use the same control law in (5.29) replacing the true parameter values $\mathbf{a}$ with an estimate $\hat{\mathbf{a}}$. As conjectured by [1], the *drag parameters* depend on the snake robot's physical parameters, which are assumed constant[2], and on the terrain friction coefficients, which also are assumed constant (or at most slowly varying) compared to the system dynamics. Under these considerations we can assume that $\mathbf{a}$ is constant, which implies

$$\dot{\tilde{\mathbf{a}}} = \dot{\mathbf{a}} - \dot{\hat{\mathbf{a}}} = -\dot{\hat{\mathbf{a}}}, \tag{5.32}$$

where $\dot{\hat{\mathbf{a}}}$ can be freely chosen in order to fulfill the adaptive control objective. The resulting overall system can therefore be expressed as

$$\boxed{a_1\dot{s} = -Y\mathbf{a} + \overline{u}_{\phi_0},} \tag{5.33a}$$
$$\boxed{\dot{\tilde{\mathbf{a}}} = -\dot{\hat{\mathbf{a}}}.} \tag{5.33b}$$

[2]This is a reasonable assumption when the snake moves on ground. In underwater scenarios, however, unmodeled time-varying physical effects (e.g., *added mass*), may indirectly influence the *drag parameters*, potentially making them time-varying as well.

Proposition 6. The origin $(s, \widetilde{\mathbf{a}}) = \mathbf{0}$ of the system in (5.33) remains *Lyapunov stable* and the system's solutions remain bounded by selecting the control law $\overline{u}_{\phi_0} = Y\hat{\mathbf{a}} - k_d s$ and the adaptive law $\dot{\hat{\mathbf{a}}} = -\Gamma Y^T s$, with $\Gamma = \Gamma^T > 0$. Moreover, the heading tracking error $s \to 0$ as $t \to \infty$. Further, under the condition of *PE*, the entire state variable $(s, \widetilde{\mathbf{a}}) \to 0$ as $t \to \infty$.

Proof. Augment the *LFC* in (5.27) to incorporate also the estimation error, as

$$\boxed{V_2 = V_1 + \frac{1}{2}\widetilde{\mathbf{a}}^T \Gamma^{-1} \widetilde{\mathbf{a}}} \tag{5.34}$$

with $\Gamma = \Gamma^T > 0$ being a positive definite and symmetric matrix. In differentiating V_2, we restart from (5.28b), since the calculations have already been carried out, and simply add the new term introduced, getting

$$\dot{V}_2(s, \widetilde{\mathbf{a}}) = s(-Y\mathbf{a} + \overline{u}_{\phi_0}) - \widetilde{\mathbf{a}}^T \Gamma^{-1} \dot{\hat{\mathbf{a}}}. \tag{5.35a}$$

Selecting now the control law

$$\boxed{\overline{u}_{\phi_0} = Y\hat{\mathbf{a}} - k_d s, \qquad k_d > 0,} \tag{5.36}$$

that uses an estimate $\hat{\mathbf{a}}$ of the *drag parameters*, and inserting it into (5.35a) gives

$$\begin{aligned} \dot{V}_2(s, \widetilde{\mathbf{a}}) &= s(-Y\mathbf{a} + Y\hat{\mathbf{a}} - k_d s) - \widetilde{\mathbf{a}}^T \Gamma^{-1} \dot{\hat{\mathbf{a}}} && (5.37a) \\ &= s(-Y(\mathbf{a} - \hat{\mathbf{a}}) - k_d s) - \widetilde{\mathbf{a}}^T \Gamma^{-1} \dot{\hat{\mathbf{a}}} && (5.37b) \\ &= -k_d s^2 - sY\widetilde{\mathbf{a}} - \widetilde{\mathbf{a}}^T \Gamma^{-1} \dot{\hat{\mathbf{a}}}. && (5.37c) \end{aligned}$$

where $\widetilde{\mathbf{a}}$ is the estimation error vector. Selecting now the adaptation law

$$\boxed{\dot{\hat{\mathbf{a}}} = -\Gamma Y^T s} \tag{5.38}$$

and inserting it into (5.37c) leads to

$$\dot{V}_2(s, \widetilde{\mathbf{a}}) = -k_d s^2 \leq 0. \tag{5.39}$$

The *Lyapunov function* in (5.34) is positive definite, therefore lower bounded, and its derivative is negative semi-definite. This implies that $V(t) \leq V(0)\ \forall t > 0$, therefore its arguments, s and $\widetilde{\mathbf{a}}$, are bounded. Due to the definition of s, this means that also $\widetilde{\theta}$ and $\dot{\widetilde{\theta}}$ are bounded. By Theorem 1 this proves that the system in (5.33) is *Lyapunov stable* but not that the dynamics of the s converges to 0. Integrating (5.39),

$$0 \leq V(t) \leq V(0) - \int_0^t k_d s^2(\tau)\, d\tau. \tag{5.40}$$

Since $s(\cdot) \in \mathcal{L}_\infty$ and $k_d > 0$ then the quantity $k_d s^2(\tau)$ in the integral is always ≥ 0, and equal to 0 if and only if $s(\tau) = 0$. Due to the lower-boundedness this only means that

$$\exists \lim_{t\to\infty} \int_0^t k_d s^2(\tau)\, d\tau = 0 \implies \lim_{t\to\infty} s(t) = \mathbf{0} \tag{5.41}$$

by Barbalat's Lemma (Theorem 3). Finally, by Theorem 4, under *PE* also the estimates $\hat{\mathbf{a}}$ converges to their (hypothetical) "true values" $\mathbf{a}$, implying that $\tilde{\mathbf{a}} \to 0$, which concludes the proof. □

We summarize here below the full adaptive control scheme.

control law	$\phi_0 = \frac{1}{v_t}\left(\overline{u}_{\phi_0} - \frac{\overline{\mathbf{e}}^T \overline{\boldsymbol{\phi}_{\text{ref}}}}{N-1}\right), \quad \overline{u}_{\phi_0} = Y\hat{\mathbf{a}} - k_d s$	(5.42a)
adaptive law	$\dot{\hat{\mathbf{a}}} = -\Gamma Y^T s, \quad \hat{\mathbf{a}} = [\hat{a}_1\ \hat{a}_2]^T$	(5.42b)
parameter transformation	$a_1 = \frac{1}{\lambda_2}, \quad a_2 = \frac{\lambda_1}{\lambda_2}$	(5.42c)
filtered tracking error	$s = \dot{\tilde{\theta}} + \Lambda\tilde{\theta} = \dot{\theta} - \dot{\theta}_r$	(5.42d)
regression matrix	$Y = [\ddot{\theta}_r\ \dot{\theta}]$	(5.42e)
reference trajectory	$\dot{\theta}_r = \dot{\theta}_{ref} - \Lambda\tilde{\theta}$	(5.42f)
tuning values	$k_d > 0, \quad \Gamma = \Gamma^T > 0, \quad \Lambda > 0.$	(5.42g)

A block-scheme diagram of the controller is pictured in Figure 5.8.

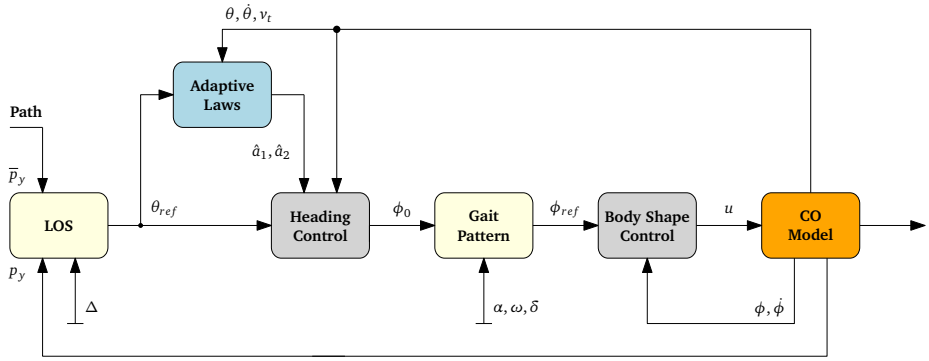


Figure 5.8: Path-following control architecture block-diagram: (i) partial feedback linearization body shape controller from (5.7) and (ii) the newly introduced adaptive heading controller from (5.42). The reference filters are omitted from the picture for simplicity.

5.4.4 Simulation results

The controller can be implemented as shown in Code listing 5.5.

Code listing 5.5: Adaptive heading controller: `adap_heading`

```
%% adaptive heading controller -------------------------

function [phi0, phi0_bar] = adap_heading(t,x,yref,cond)
    global Nl lookahead_dist ref;
    global a e_bar;
    global l1_hat l2_hat a_hat dt;

    % initialization
    persistent thref_filt thref_dot thref_ddot;
    if isempty(thref_filt)
        [thref_filt, thref_dot, thref_ddot] = deal(0);
    end

    %TUNING ---------
    Lambda = 0.1;
    Kd = 0.01;
    Gamma = diag([10,1]);
    %----------------

    % LOS
    py = x(Nl+2);
    thref = -atan2(py - yref, lookahead_dist);

    %TUNING ------
    xi = 1;
    deltaT = 4;
    %-------------

    [thref_filt, thref_dot, thref_ddot] = ...
        ref_filter([thref_filt, thref_dot, thref_ddot]', thref, xi, deltaT);

    heading = x(Nl);
    heading_dot = x(end-2);
    vt = x(end-1);

    % filtered tracking error
    s = (heading_dot - thref_dot) + Lambda*(heading - thref_filt);
    thetaR_dot = thref_dot - Lambda*(heading - thref_filt);
    thetaR_ddot = thref_ddot - Lambda*(heading_dot - thref_dot);

    % adaptive part
    global v vdot;
    a_tilde = a - a_hat;
    Gammainv = inv(Gamma);
    v(2) = 0.5*a(1)*s^2 + 0.5*a_tilde'*Gammainv*a_tilde;
    vdot(2) = -Kd*s^2;

    Y = [thetaR_ddot, heading_dot];
    a_hat_dot = @(a,s) -Gamma*Y'*s;

    if cond
        a_hat = euler_step(a_hat, s, a_hat_dot, dt);

        try
```

```
            l2_hat = 1/a_hat(1);        %try to avoid 1/0
            l1_hat = a_hat(2)*l2_hat;
        end

        u_bar = Y*a_hat - Kd*s;
        phi0 = 1/vt * u_bar;
    else
        phi0 = 0;
    end

    [phiref_bar, phiref_bar_dot, phiref_bar_ddot] = ref(t);
    phi0_bar(1) = -e_bar'*phiref_bar/(Nl-1);        %phi0_bar
    phi0_bar(2) = -e_bar'*phiref_bar_dot/(Nl-1);    %phi0_bar_dot
    phi0_bar(3) = -e_bar'*phiref_bar_ddot/(Nl-1);   %phi0_bar_ddot
end
```

A straight-line or piecewise linear path is not of major interest in this scenario for several reasons. First, such a path, and the associated dynamics, are not sufficiently exciting to guarantee PE in the adaptive scheme, nor to effectively exploit the feedforward term $Y\hat{\boldsymbol{a}}$ in the control law. Instead, the controller can rely almost exclusively on its feedback stabilizing term $-k_d s$ to track the trajectory, which, in fact, does not demand any genuine tracking capability. Consequently, the controller is not forced to adapt its parameters to meaningful values, since stabilization alone suffices to ensure the control objectives in this scenario. The proposed controller is fully capable of performing such a task, but the experiment lacks to highlight any of its relevant properties.

We then choose to validate the performance of the adaptive heading controller through a sinusoidal-path experiment, where the path is defined as

$$y_{ref}(x) = 2\sin(0.5 \cdot x) \tag{5.43}$$

and we recall that $y_{ref}(t)$ is chosen moment by moment from the position of the CM of the snake robot $p_x(t)$ + the *look-ahead distance* Δ, namely

$$y_{ref}(t) = y_{ref}(\, p_x(t) + \Delta\,). \tag{5.44}$$

All physical parameters of the model, as well as the gait parameters and reference filter settings, are kept identical to those introduced previously. The *rotational drag coefficients* of the CO-model are fixed to $\lambda_1 = 0.5$ and $\lambda_2 = 20$, as introduced before and used from [1].

The simulation is run over a slightly longer time horizon $T = 600$ [s], since the heading dynamics constitute the outer, and so *slower* loop in the control hierarchy. Moreover, due to the feedforward terms in the control law, the heading dynamics no longer exhibit oscillatory behavior and evolve only in response to variations in the reference signal. The initial estimates are set to $\hat{a}_1 = 0, \hat{a}_2 = 0$. The adaptive heading controller gains are chosen as $\Lambda = 0.1$, $k_d = 0.01$, and $\Gamma = \mathrm{diag}([10, 1])$, while for the Partial Feedback Linearization Body Shape controller the gains are $k_p = 3$, $k_d = 1$.

The heading controller is activated only when the forward velocity exceeds the threshold $v_{t,\min} = 2$ [cm/s]. Furthermore, because of the disregarded interconnection term, large undesired values of this term may destabilize the hierarchical path-following scheme during initial transients. To mitigate this, we introduce an additional threshold on the joint tracking error, $\|\widetilde{\boldsymbol{\phi}}\| \leq \|\widetilde{\boldsymbol{\phi}}\|_{\min} = 0.1$, which must be satisfied before the heading controller is enabled. This guarantees that the body-shape controller is already tracking the joint angles with sufficient accuracy, thereby ensuring safe activation of the heading controller. This same mechanism is also employed when sudden changes in terrain friction occur and tracking of the joint variables is no longer satisfied: the heading controller is temporarily deactivated until the snake re-establishes correct body-shape tracking.

All the remaining parameters not explicitly mentioned are kept identical to those of the previous simulations, except for the parameter ω_c of the ϕ_0 reference filter, which is made faster setting $\Delta T = 2\pi/\omega_c = 0.1$ [s]. This adjustment is necessary because the control input ϕ_0 now plays a crucial role also in the adaptive law, and a filter with an insufficiently fast step response would inevitably de-synchronize the actually applied $\phi_{0,\text{filt}}$ (which is filtered and thus delayed) from the nominal value ϕ_0 used in the adaptive scheme of the controller. The results of this experiment are presented in Figure 5.9.

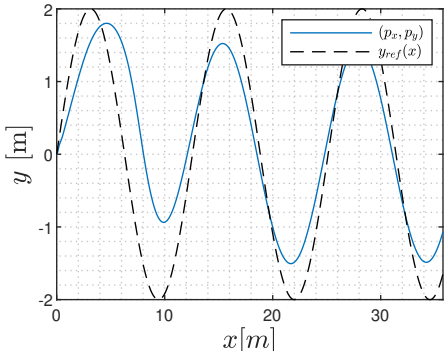
5.5 Fully-Adaptive Locomotion Control

In our pursuit of a fully adaptive path-following scheme, we aim to adapt both the heading and the body-shape controller parameters, such that the overall controller does not rely on prior knowledge of the terrain properties nor on unavailable model parameters. To this end, we first make the inner body-shape layer of the *CO-model* adaptive too and demonstrate that the combined adaptive scheme performs satisfactorily in the absence of knowledge of the friction coefficients and the model's drag parameter. We refer to this scheme as the adaptive locomotion controller, since all the two modules required for locomotion are made adaptive.

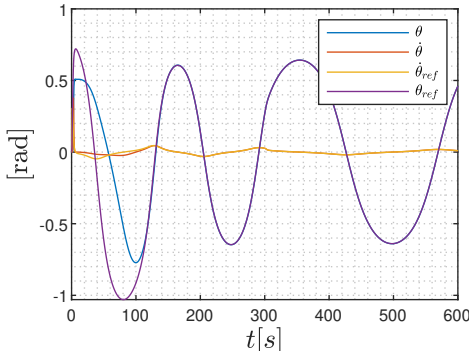
We then first introduce and derive a novel friction-adaptive body-shape controller based on the *CO-model* in Section 5.5.1 and validate it together with the adaptive heading controller from Section 5.4 in Section 5.5.2.

5.5.1 Body Shape Adaptive Control

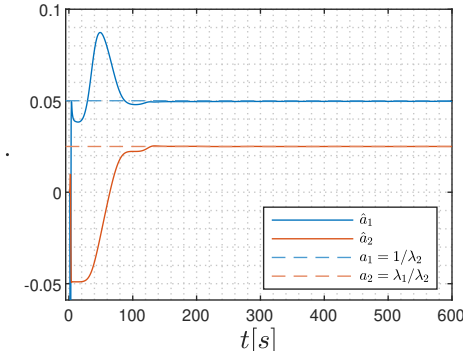
The approach undertaken is very similar to the one adopted for the adaptive body shape controller designed for the *complex model* in Section 4.3, without the need to rewrite any friction force in a regression-based form, since the present model is already well-suited for adaptive control. The control law to adapt is the one in (5.7), but instead of relying on the true values of the friction coefficients, we rather use an estimate of them, namely $\hat{c}_n, \hat{c}_p$. The "estimated" control law is then



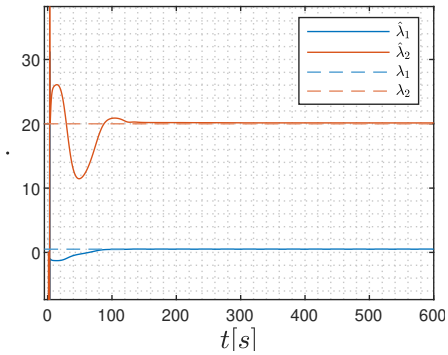
(a) Trace of the CM (p_x, p_y) versus the sinusoidal path reference $y_{ref}(x)$. Please note that $y_{ref}(x)$ is the true path, i.e., doesn't take in account the look-ahead distance: therefore the tracking appears delayed even when the estimates converge to the model parameters.



(b) Heading variables $\theta, \dot{\theta}$ vs the LOS reference angles. After an initial transient both converge smoothly to their reference.



(c) Estimated parameters $\hat{a}$ versus the model parameters $a_1 = 1/\lambda_2$ and $a_2 = \lambda_1/\lambda_2$.



(d) *drag parameter* estimates $\hat{\lambda}_1, \hat{\lambda}_2$ versus the conjectured true values λ_1, λ_2.

Figure 5.9: Simulation results of the adaptive heading controller

$$\mathbf{u} = m\left(\mathbf{D}\mathbf{D}^T\right)^{-1}\left(\overline{\mathbf{u}} + \frac{\hat{c}_n}{m}\dot{\boldsymbol{\phi}} - \frac{\hat{c}_p}{m}v_t\mathbf{A}\mathbf{D}^T\boldsymbol{\phi}\right), \tag{5.45a}$$

$$\overline{\mathbf{u}} = \ddot{\boldsymbol{\phi}}_{ref} - k_d(\dot{\boldsymbol{\phi}} - \dot{\boldsymbol{\phi}}_{ref}) - k_p(\boldsymbol{\phi} - \boldsymbol{\phi}_{ref}), \tag{5.45b}$$

where we recall that c_p is defined as

$$c_p = \frac{c_n - c_t}{2l}.$$

By substituting (5.45) in (5.4e), and rewriting $\mathbf{v}_\phi$ as $\dot{\boldsymbol{\phi}}$, the resulting closed-loop dynamics is

$$\ddot{\boldsymbol{\phi}} = -\frac{c_n}{m}\dot{\boldsymbol{\phi}} + \frac{c_p}{m}v_t\mathbf{A}\mathbf{D}^T\boldsymbol{\phi} + \overline{\mathbf{u}} + \frac{\hat{c}_n}{m}\dot{\boldsymbol{\phi}} - \frac{\hat{c}_p}{m}v_t\mathbf{A}\mathbf{D}^T\boldsymbol{\phi} \tag{5.46a}$$

$$= \overline{\mathbf{u}} - \frac{1}{m}\dot{\boldsymbol{\phi}}(c_n - \hat{c}_n) + \frac{v_t}{m}\mathbf{A}\mathbf{D}^T\boldsymbol{\phi}(c_p - \hat{c}_p) \tag{5.46b}$$

$$= \overline{\mathbf{u}} - \frac{1}{m}\dot{\boldsymbol{\phi}}\widetilde{c_n} + \frac{v_t}{m}\mathbf{A}\mathbf{D}^T\boldsymbol{\phi}\widetilde{c_p} \tag{5.46c}$$

$$= \overline{\mathbf{u}} + \underbrace{\left[-\tfrac{1}{m}\dot{\boldsymbol{\phi}} \quad \tfrac{v_t}{m}\mathbf{A}\mathbf{D}^T\boldsymbol{\phi}\right]}_{Y(\mathbf{x})} \cdot \underbrace{\begin{bmatrix}\widetilde{c_n}\\ \widetilde{c_p}\end{bmatrix}}_{\widetilde{\mathbf{c}}} \tag{5.46d}$$

$$= \overline{\mathbf{u}} + Y(\mathbf{x})\widetilde{\mathbf{c}}, \tag{5.46e}$$

where $\widetilde{c_n}$ and $\widetilde{c_p}$ denotes the estimation errors on the parameters and have been collected in the compact vector $\widetilde{\mathbf{c}}$ for convenience, while $Y(\mathbf{x})$ represent a regression matrix containing the interconnection signals between $\widetilde{\mathbf{c}}$ and the joint error dynamics. Please note that for simplicity the estimation problem is carried out directly on the pair $(\hat{c}_n, \hat{c}_p)$ rather than on $(\hat{c}_t, \hat{c}_n)$. The inverse transformation is unique and always doable. Substituting now for the linearized input $\widetilde{\mathbf{c}}$ leads to the closed loop error dynamics

$$\ddot{\boldsymbol{\phi}} = \ddot{\boldsymbol{\phi}}_{ref} - k_d\dot{\widetilde{\boldsymbol{\phi}}} - k_p\widetilde{\boldsymbol{\phi}} + Y(\mathbf{x}) \cdot \widetilde{\mathbf{c}} \tag{5.47a}$$

$$\ddot{\widetilde{\boldsymbol{\phi}}} + k_d\dot{\widetilde{\boldsymbol{\phi}}} + k_p\widetilde{\boldsymbol{\phi}} = Y(\mathbf{x}) \cdot \widetilde{\mathbf{c}}. \tag{5.47b}$$

This second-order system can be written more compactly in state-space form by introducing the augmented error vector $\mathbf{z} = [\dot{\widetilde{\boldsymbol{\phi}}},\ \widetilde{\boldsymbol{\phi}}]^T$:

$$\underbrace{\begin{bmatrix}\ddot{\widetilde{\boldsymbol{\phi}}}\\ \dot{\widetilde{\boldsymbol{\phi}}}\end{bmatrix}}_{\dot{\mathbf{z}}} = \underbrace{\begin{bmatrix}-k_d I_{N-1} & -k_p I_{N-1}\\ I_{N-1} & \mathbf{0}\end{bmatrix}}_{A<0}\underbrace{\begin{bmatrix}\dot{\widetilde{\boldsymbol{\phi}}}\\ \widetilde{\boldsymbol{\phi}}\end{bmatrix}}_{\mathbf{z}} + \underbrace{\begin{bmatrix}Y(\mathbf{x})\\ \mathbf{0}_{(N-1)\times 2}\end{bmatrix}}_{F(\mathbf{x})}\widetilde{\mathbf{c}} \tag{5.48a}$$

$$\dot{\mathbf{z}} = A\mathbf{z} + F(\mathbf{x}) \cdot \widetilde{\mathbf{c}}. \tag{5.48b}$$

Here $A < 0$ is a *Hurwitz matrix* (i.e., the real part of all its *eigenvalues* is strictly negative), so that in the absence of estimation error ($\widetilde{\mathbf{c}} = \mathbf{0}$) the system is *exponentially stable*. The estimation error $\widetilde{\mathbf{c}}$ therefore acts as a disturbance input through the matrix $F(\mathbf{x})$, which depends only on the robot state variables. Since the true coefficients are assumed constant (or slowly varying compared to the system dynamics), we have $\dot{\mathbf{c}} = 0$, which implies

$$\dot{\widetilde{\mathbf{c}}} = -\dot{\hat{\mathbf{c}}}. \tag{5.49}$$

The resulting overall system is therefore a cascade:

$$\boxed{\begin{aligned} \dot{\mathbf{z}} &= A\mathbf{z} + F(\mathbf{x}) \cdot \widetilde{\mathbf{c}}, && \text{(5.50a)} \\ \dot{\widetilde{\mathbf{c}}} &= -\dot{\hat{\mathbf{c}}}. && \text{(5.50b)} \end{aligned}}$$

This representation clearly shows that the closed-loop error dynamics is *ISS* with respect to the estimation error. The next step is to design the adaptation law for $\hat{\mathbf{c}}$ so as to make $\widetilde{\mathbf{c}}$ stable, ensuring stability of the overall cascaded system. From here on the proceeding is exactly the same as the one in Proposition 4.

Proposition 7. The origin of the cascaded system in (5.50) remains Lyapunov stable by selecting the adaptive law $\dot{\hat{\mathbf{c}}} = 2\Gamma F(\mathbf{x})^T P\mathbf{z}$, and thus the system's solutions remains bounded. Moreover, the gait tracking error $\mathbf{z} \to \mathbf{0}$ as $t \to \infty$. Further, under *PE* the entire state variable $(\mathbf{z}, \widetilde{\mathbf{c}}) \to 0$ as $t \to \infty$.

Proof. Consider the *LFC* for the system in (5.50):

$$\boxed{V(\mathbf{z}, \widetilde{\mathbf{c}}) = \mathbf{z}^T P\mathbf{z} + \frac{1}{2}\widetilde{\mathbf{c}}^T \Gamma^{-1}\widetilde{\mathbf{c}},} \tag{5.51}$$

where $\Gamma = \Gamma^T > 0$ is a positive definite matrix (i.e. always invertible) and $P = P^T > 0$ is a symmetric, positive definite and constant matrix satisfying the Lyapunov equation:

$$A^T P + PA = -Q \text{ for some } Q > 0, \tag{5.52}$$

and $A < 0$ is the stable matrix in (5.48). Q can be chosen arbitrarily provided that the positiveness condition is respected. We propose it to be

$$Q = \begin{bmatrix} k_d I_{N-1} & \mathbf{0} \\ \mathbf{0} & k_p I_{N-1} \end{bmatrix}. \tag{5.53}$$

Then, by Theorem 7 (5.52) has a unique constant solution P, which can be found analytically due to the form of A and the one chosen for Q, which is

$$P = \begin{bmatrix} \frac{1}{2}\left(\frac{1+k_d}{k_d}\right) I_{N-1} & \frac{1}{2} I_{N-1} \\ \frac{1}{2} I_{N-1} & \frac{1}{2}\left(k_p\left(\frac{1+k_d}{k_d}\right) + k_d\right) I_{N-1} \end{bmatrix}. \tag{5.54}$$

Differentiating V along the trajectories of the closed-loop dynamical system provided in (5.50) yields

$$\dot{V}(\mathbf{z},\widetilde{\mathbf{c}}) = \mathbf{z}^T P\dot{\mathbf{z}} + \dot{\mathbf{z}}^T P\mathbf{z} + \widetilde{\mathbf{c}}^T\Gamma^{-1}\dot{\widetilde{\mathbf{c}}} \tag{5.55a}$$

$$= \mathbf{z}^T P\Big(A\mathbf{z} + F(\mathbf{x})\cdot\widetilde{\mathbf{c}}\Big) + \Big(A\mathbf{z} + F(\mathbf{x})\cdot\widetilde{\mathbf{c}}\Big)^T P\mathbf{z} + \widetilde{\mathbf{c}}^T\Gamma^{-1}\dot{\widetilde{\mathbf{c}}} \tag{5.55b}$$

$$= \mathbf{z}^T \underbrace{(PA + A^T P)}_{-Q}\mathbf{z} + 2\widetilde{\mathbf{c}}^T F^T(\mathbf{x})P\mathbf{z} + \widetilde{\mathbf{c}}^T\Gamma^{-1}\dot{\widetilde{\mathbf{c}}} \tag{5.55c}$$

$$= -\mathbf{z}^T Q\mathbf{z} + 2\widetilde{\mathbf{c}}^T F^T(\mathbf{x})P\mathbf{z} - \widetilde{\mathbf{c}}^T\Gamma^{-1}\dot{\hat{\mathbf{c}}}. \tag{5.55d}$$

Selecting now the adaptation law

$$\boxed{\dot{\hat{\mathbf{c}}} = 2\Gamma F^T(\mathbf{x})P\mathbf{z}} \tag{5.56}$$

and inserting it in (5.55d) leads to

$$\dot{V}(\mathbf{z},\mathbf{c}) = -\mathbf{z}^T Q\mathbf{z} + 2\widetilde{\mathbf{c}}^T F^T(\mathbf{x})P\mathbf{z} - \widetilde{\mathbf{c}}^T\Gamma^{-1}\cdot 2\Gamma F^T(\mathbf{x})P\mathbf{z} \tag{5.57a}$$

$$= -\mathbf{z}^T Q\mathbf{z} \leq 0. \tag{5.57b}$$

The *Lyapunov function* in (5.51) is positive definite, therefore lower bounded, and its derivative in (5.57b) is negative semi-definite. This implies that $V(t) \leq V(0)\ \forall t > 0$, therefore its arguments, $\mathbf{z}$ and $\widetilde{\mathbf{c}}$, are bounded, or more formally, that $\mathbf{z}(\cdot) \in \mathcal{L}_\infty$ and also the estimation error $\widetilde{\mathbf{c}}(\cdot) \in \mathcal{L}_\infty$. This is sufficient to prove that the cascaded system in (5.50) is *Lyapunov stable* (see Theorem 1) but not that the dynamics of the *augmented error vector* $\mathbf{z}$, and so of the gait tracking errors, converges to 0. Integrating (5.57b),

$$0 \leq V(t) \leq V(0) - \int_0^t \mathbf{z}(\tau)^T Q\mathbf{z}(\tau)\, d\tau. \tag{5.58}$$

Since $\mathbf{z}(\cdot) \in \mathcal{L}_\infty$ and $Q > 0$ then the quantity $\mathbf{z}(\tau)^T Q\mathbf{z}(\tau)$ in the integral is always ≥ 0, and equal to 0 if and only if $\mathbf{z}(\tau) = 0$. Due to the lower-boundedness this only means that

$$\exists \lim_{t\to\infty} \mathbf{z}(t)^T Q\mathbf{z}(t) = 0 \implies \lim_{t\to\infty} \mathbf{z}(t) = \mathbf{0} \tag{5.59}$$

By Barbalat's Lemma (Theorem 3). In words this means that the gait tracking errors converge *asymptotically* to 0. Finally, by Theorem 4, under *PE* also the estimates $\hat{\mathbf{c}}$ converges to their true values $\mathbf{c}$, implying that $\widetilde{\mathbf{c}} \to 0$, which concludes the proof. □

Remark: The tracking error dynamics becomes UGES when the estimation error is $\widetilde{\mathbf{c}} = \mathbf{0}$ (see Proposition 1). This constitutes a remarkable advantage over the weaker *asymptotic convergence* to 0 of the joint tracking error achieved during adaptation of the estimation parameters, owing to the specific formulation in

which we set up the adaptive problem. We summarize here below the full adaptive scheme.

$$\text{control law} \qquad \mathbf{u} = m\left(\mathbf{D}\mathbf{D}^T\right)^{-1}\left(\overline{\mathbf{u}} + \frac{\hat{c}_n}{m}\dot{\boldsymbol{\phi}} - \frac{\hat{c}_p}{m}v_t\mathbf{A}\mathbf{D}^T\boldsymbol{\phi}\right), \tag{5.60a}$$

$$\text{stabilizing law} \qquad \overline{\mathbf{u}} = \ddot{\boldsymbol{\phi}}_{ref} - k_d\dot{\tilde{\boldsymbol{\phi}}} - k_p\tilde{\boldsymbol{\phi}} \tag{5.60b}$$

$$\text{control gains} \qquad k_p > 0,\ k_d > 0, \tag{5.60c}$$

$$\text{adaptive law} \qquad \dot{\hat{\mathbf{c}}} = 2\Gamma F(\mathbf{x})^T P\mathbf{z}, \quad \hat{\mathbf{c}} = [\hat{c}_n\ \hat{c}_p]^T, \quad \mathbf{z} = [\dot{\tilde{\boldsymbol{\phi}}},\ \tilde{\boldsymbol{\phi}}]^T \tag{5.60d}$$

$$F(\mathbf{x}) = \begin{bmatrix} Y(\mathbf{x}) \\ \mathbf{0}_{(N-1)\times 2} \end{bmatrix} \tag{5.60e}$$

$$\text{regression matrix} \qquad Y(\mathbf{x}) = \begin{bmatrix} -\frac{1}{m}\dot{\boldsymbol{\phi}} & \frac{v_t}{m}\mathbf{A}\mathbf{D}^T\boldsymbol{\phi} \end{bmatrix} \tag{5.60f}$$

$$\text{lyapunov solution} \qquad P = \begin{bmatrix} \frac{1}{2}\left(\frac{1+k_d}{k_d}\right)I_{N-1} & \frac{1}{2}I_{N-1} \\ \frac{1}{2}I_{N-1} & \frac{1}{2}\left(k_p\left(\frac{1+k_d}{k_d}\right)+k_d\right)I_{N-1} \end{bmatrix} \tag{5.60g}$$

$$\text{adaptation gain} \qquad \Gamma = \Gamma^T > 0. \tag{5.60h}$$

Figure 5.10 shows a block-diagram picture of the adaptive locomotion controller.

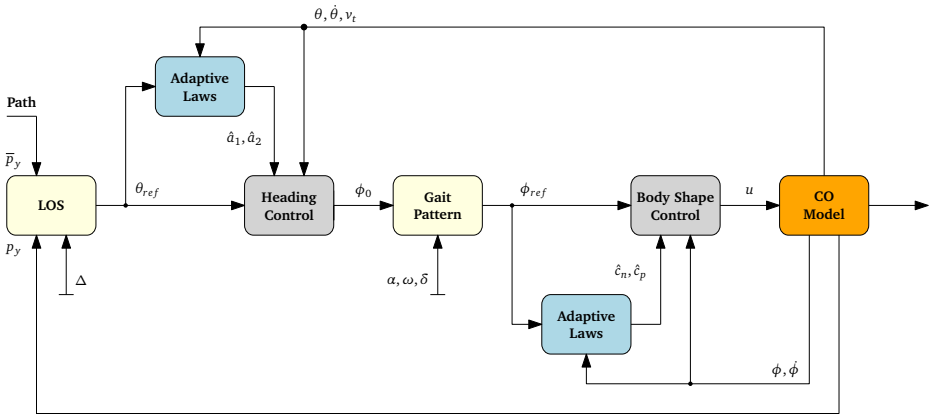


Figure 5.10: Adaptive locomotion controller block-diagram, comprehensive of the adaptive body-shape controller for the CO-model in (5.60) and the adaptive heading controller in (5.42). Again, the reference filters are omitted from the picture for simplicity.

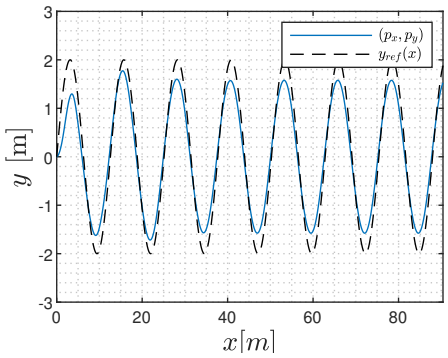
5.5.2 Simulation results

The controller in (5.60) can be implemented as shown in Code listing 5.5.

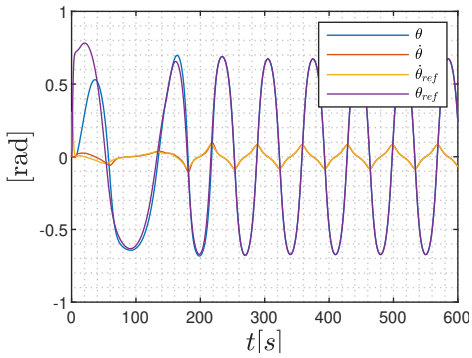
Code listing 5.6: Adaptive body-shape controller for the CO-model: `co_fblin`

```
%% adaptive fb-lin controller ---------------------------------

function u = co_adap(x, phi_ref, phi_ref_dot, phi_ref_ddot)
    global Nl l m A D;
    global dt int_method cn cp c_hat maxc minc;

    %TUNING ---------
    Kp = 3; Kd = 1;
    Gamma = diag([30,10000]);
    %----------------

    %(14b-21-22)
    Az = [-Kd*eye(Nl-1), -Kp*eye(Nl-1);
        eye(Nl-1), zeros(Nl-1, Nl-1)];
    Q = [Kd*eye(Nl-1), zeros(Nl-1,Nl-1);
         zeros(Nl-1,Nl-1), Kp*eye(Nl-1)];
    P = lyap(Az', Q);

    %some checks
    assert(isequal(P,P'), 'P is not symmetric');
    assert(isequal(Q,Q'), 'Q is not symmetric');
    assert(all(eig(P)>0), 'P is not positive def');
    assert(all(eig(Q)>0), 'Q is not positive def');
    assert(all(abs(Az'*P+P*Az - -Q) < 1e-12, 'all'), ...
        'riccati equation is not satisfied');

    phi = x(1:Nl-1);
    phi_dot = x(Nl+3:end-3);
    vt = x(end-1);

    % adaptive laws
    z = [phi_dot - phi_ref_dot; phi - phi_ref];
    Y = [-1/m * phi_dot, vt/m * A*D' * phi];
    F = [Y; zeros(Nl-1,2)];

    adap_law = (@(c,z) 2*Gamma*F'*P*z);
    c_hat = int_method(c_hat, z, adap_law, dt);

    % parameter projection
    c_hat(1) = max(minc, min(c_hat(1), maxc));              %cn
    c_hat(2) = max(0, min(c_hat(2), (maxc-minc)/(2*l)));    %cp

     % lyapunov analysis
    global v vdot;
    c_tilde = [cn;cp] - c_hat;
    Gammainv = inv(Gamma);
    v(1) = z'*P*z + 0.5*c_tilde'*Gammainv*c_tilde;
    vdot(1) = -z'*Q*z;

    %fb linearizing control law
    mDDTinv = m*inv(D*D');
    cn_hat = c_hat(1);
    cp_hat = c_hat(2);

    u_bar = phi_ref_ddot - Kd * (phi_dot - phi_ref_dot) - Kp * (phi - phi_ref);
    u = mDDTinv * (u_bar + cn_hat/m * phi_dot - cp_hat/m * vt * A*D' * phi);
end
```

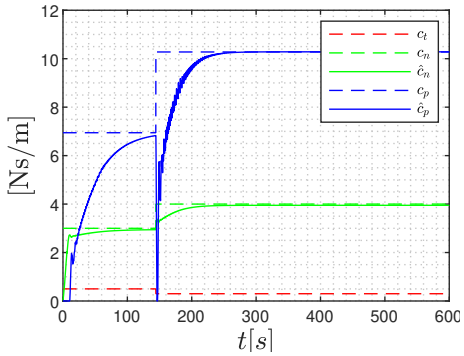
To validate the adaptive locomotion controller, we employ the same sinusoidal path introduced in Equation (5.43). All the physical parameters of the model and the control gains are kept identical to those used previously in Section 5.4.4. For the newly introduced Adaptive Body Shape controller, we set $k_p = 3$ and $k_d = 1$, as before, and choose $\Gamma = \text{diag}([30, 10^4])$. The large adaptive gain assigned to $\hat{c}_p$ is justified by its associated perturbation signal in the regressor Y, which is scaled by v_t. Since v_t is relatively small compared to the other quantities involved, a high adaptation gain is required to adequately balance its effect. The initial estimates are set to $\hat{\mathbf{c}} = [c_{\min}, c_{\min}]^T$, with $c_{\min} = 0.01$ and $c_{\max} = 100$ used to project the friction coefficients as an additional safety measure. The results of the experiment are presented in Figure 5.11.



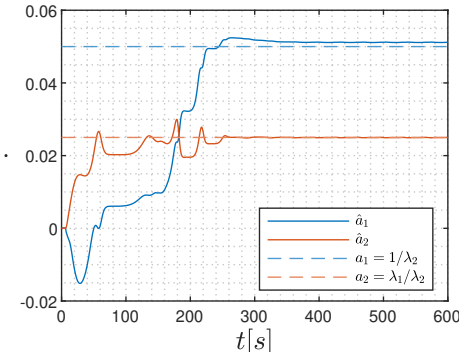
(a) Trace of the CM (p_x, p_y) versus the sinusoidal path reference $y_{ref}(x)$ employed to achieve PE in the adaptive scheme.



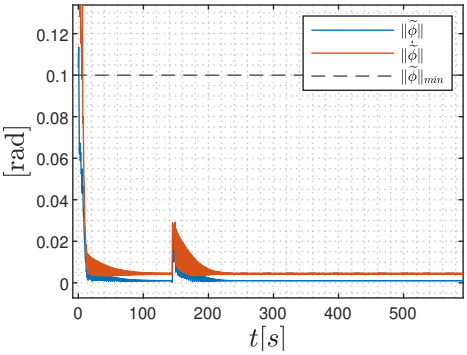
(b) Heading variables $\theta, \dot{\theta}$ vs the LOS reference angles. Note the almost perfect tracking for $t \geq 200$ [s] due to the convergence of the *drag parameters* (see below).



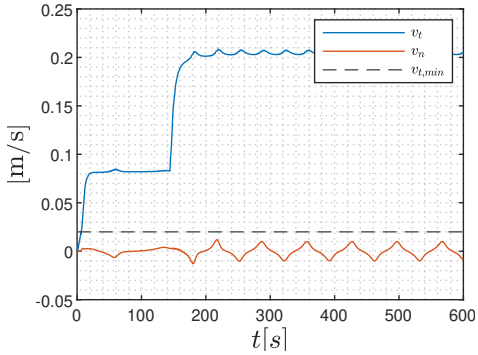
(c) Convergence of the estimate of the friction coefficients $\hat{c}_n, \hat{c}_p$ to their true values c_n, c_p. In red it's also visible the correspondent c_t.



(d) Convergence of the estimation parameters $\hat{a}$ to the model parameters $a_1 = 1/\lambda_2$ and $a_2 = \lambda_1/\lambda_2$.



(e) Norm of the joint tracking errors. The black dashed line is the acceptance threshold $\|\widetilde{\boldsymbol{\phi}}\|_{\text{min}}$ needed to activate the heading adaptive controller.



(f) Normal velocity v_t and tangential velocity v_n. The black dashed line is the minimum forward velocity $v_{t,min}$ needed to activate the heading adaptive controller.

Figure 5.11: Simulation results of the adaptive locomotion controller. For $p_x \geq 10$ [m] the friction coefficients changes from $c_t, c_n = 0.5, 3$ to $c_t, c_n = 0.3, 4$.

Chapter 6

The complete Adaptive Locomotion Controller

A natural question that arises is whether the fully-Adaptive Locomotion Controller can be translated from the control-oriented (CO) model to the full, complex model of the snake robot, how it could be implemented, and which adjustments would be necessary.

The plain-vanilla Adaptive Locomotion scheme cannot be directly applied to the complex model, and attempting to do so typically leads the system to diverge in a variety of creative ways (see Section 6.2). There are several reasons for this behavior, and we present the final solution in a problem-solution chronological order, reflecting how each issue was discovered and addressed. Each choice made to solve a particular problem often introduces new *cascaded* problems, which must then be resolved or balanced with previous choices.

It is important to remark that the issues here identified and the reasons proposed for them are not derived from a formal theoretical analysis, but rather conjectured from experimental observations and by analogy with strategies commonly adopted in the literature. Likewise, the corresponding solutions should be regarded as practical design choices when no formal proof is available.

6.1 Body Shape Adaptive Controller

The first choice made is to replace the adaptive body-shape controller designed for the *CO-model* in Section 5.5.1 with the *complex model*-dedicated controller introduced in Section 4.3. Although this appears to be a natural choice, the cascaded stability of the overall architecture is no longer guaranteed, as the original analysis was carried out on the *CO-model*. Nevertheless, since the *signal* ϕ_0 is the only interconnection with the heading controller, this replacement can be implemented without extra-care, provided that ϕ_0 is sufficiently smooth and bounded to remain trackable.

The only difference to take into account between the two controllers is *the sign*

of ϕ_0 in the gait reference. Due to the different convention adopted on the definition of the joint variable for the two models, any heading controller designed for the *CO-model* must have the sign of ϕ_0 reversed to work properly on the *complex model*.

6.2 Heading Adaptive Controller

This represents the core of the challenge. Expecting the adaptive heading controller to work as-is on the *complex model* is unrealistic (and demonstrated from the several failed numerical simulations) for a series of issues:

1. **Model's inconsistency:** The controller is designed for a model that naturally possesses the estimation parameters λ_1, λ_2, or, equivalently, the derived control parameters a_1, a_2. However it is applied to a model whose dynamics is structurally different, with only the external behavior appearing similar – while the underlying motions are governed by entirely different physical principles. As a result, the estimation parameters $\hat{a}_1, \hat{a}_2$ do not have any true values a_1, a_2 to converge to, and the entire estimation scheme relies solely on driving the filtered heading tracking error s to zero.
2. **Filtered tracking error inconsistency:** The controller is based on a model in which perfect cancellation of the oscillatory behavior of the heading θ and the CM position (p_x, p_y) is possible via feedforward terms in the control law. In this case, the filtered tracking error s represents the *true* weighted sum of the deviation of the heading from the reference $\overline{\theta_{ref}}$ and its derivative from $\dot{\overline{\theta_{ref}}}$. Conversely, in the *complex model*, the oscillatory behavior cannot be canceled due to structural differences, and the small, high-frequency ripple around the reference – occurring at the gait frequency ω – is erroneously perceived as additional tracking error. Mathematically, this translates to s becoming:

$$s = \dot{\widetilde{\theta}} + \Lambda\widetilde{\theta} + \underbrace{k \cdot \sin(f(\omega))}_{\text{HF ripple term}}, \tag{6.1}$$

   where $\widetilde{\theta} = \overline{\theta} - \overline{\theta_{ref}}$ is the heading tracking error, k is some scalar value and $f(\cdot)$ is some function of the gait pattern parameter ω. The presence of the last term leads to two undesired effects:

   a. **Feedback-induced chattering:** Being the controller unaware that this oscillation is both natural and necessary for propulsion, it attempts to cancel it by issuing high-frequency commands that alternately counter deviations to the left and right. This generates a persistent feedback loop that sustains chattering and gradually reduces forward propulsion, and the resulting decrease in v_t – which appears in the denominator of the control law for ϕ_0 – leads to progressively larger control commands until the propulsion eventually halts.

b. **Constantly perturbed adaptation:** The heading adaptive scheme is persistently perturbed by the sinusoidal component of s, preventing the adaptation process from ever settling. We therefore expect the control parameters $\hat{a}_1, \hat{a}_2$ to not converge. Even worse, the estimates may grow unbounded if the signs of regressor Y and s aligns, due to mathematical reasons, or simply due to the delay ϕ_0 has because of the reference filters.

3. **Direct high-gain feedback loop on $\dot{\overline{\theta}}$:** The regressor $Y = [\ddot{\theta}_r \ \dot{\theta}]$ in the adaptive law includes a direct dependency on the heading velocity. While this is harmless for the *CO-model*, it's not for the *complex model*. Having $\dot{\overline{\theta}}$ a high frequency oscillatory nature, even the smallest delay in the control loop – which in this case is again the reference filters for ϕ_0 – leads to a positive feedback that propagates to the control law, and also to the estimate adaptation scheme.

6.2.1 Low-Pass Filtering as State Estimator

Instead of attempting to cancel what is inherently non-cancelable, the key idea is to prevent the controller from reacting to the high-frequency gait oscillations. This is achieved by low-pass filtering $\overline{\theta}$ with a 2nd order reference filter before passing it to the controller. In this way, the feedback variables $\overline{\theta}$ and $\dot{\overline{\theta}}$ get replaced by a smoother, *average* version, which recalls the behavior of the essential dynamics of the *CO-model* while remaining blind to the structural wiggles of the *complex model*.

Conceptually, a single second-order filter should not be regarded merely as a smoother, but rather as a state estimator –especially for the heading angular velocity. This is also a very standard approach in robotics for two reasons. First, in practice one rarely has direct access to a clean velocity measurement: sensors typically provide only position (e.g. rotary encoders), while velocity has to be estimated by differentiation, which is a high-pass operation and therefore amplifies noise, whereas filtering is a low-pass operation that attenuates it. Thus, instead of applying a 'two-filter" method – differentiating a noisy position and then filtering the result – the 2nd order reference filter produce both a smooth signal for position and its derivative. This yields a more robust and reliable velocity estimate, ensuring that the controller acts on the relevant low-frequency dynamics rather than the propulsion-related high-frequency oscillations. We believe this design choice is a pragmatic advantage rather than a limitation, due to the practical orientation of this work.

This directly addresses issues 2 (**filtered tracking error inconsistency**) and 2a (**feedback-induced chattering**). However, the filter cannot be made arbitrarily slow to completely remove the oscillations at the gait frequency ω, since ω is already relatively low, and doing so would render the controller's feedback action excessively sluggish and could potentially destabilize the closed-loop system.

Consequently, a residual ripple persists in the heading variables, implying that the system still suffers from issue 2b (**constantly perturbed adaptation**).

6.2.2 Adaptive Law Compensation for Excitation Loss

Filtering $\overline{\theta}$, as in Section 6.2.1, brings some advantages but, unfortunately, also a hidden problem: a lack of excitation in the second estimate, which previously depended on $\dot{\theta}$ as its perturbation signal. With the filtering in place, this signal is now smooth and slow, so its dynamics are significantly weakened. Using a sufficiently large adaptive gain to compensate the weak excitation power would render the estimation excessively noisy and impractical. This issue can be mitigated by a slight modification of the control law. Instead of selecting it as in (5.29), we now choose it as

$$\boxed{\overline{u}_{\phi_0} = \overline{Y}\mathbf{a} - k_d s, \qquad k_d > 0,} \tag{6.2}$$

where we make use of a slightly modified regression matrix $\overline{Y} \triangleq [\ddot{\theta}_r \;\; \dot{\theta}_r]$ instead of the regression matrix $Y \triangleq [\ddot{\theta}_r \;\; \dot{\theta}]$ previously employed. We emphasize that the formal analysis is still carried out on the *CO-model* dynamics, in order to preserve fidelity with the model on which the controller is based. Substituting therefore the modified control law into (5.28b) yields

$$\begin{aligned}
\dot{V}_1'(s) &= s(-Y\mathbf{a} + \overline{Y}\mathbf{a} - k_d s) && \text{(6.3a)}\\
&= s(\begin{bmatrix}\overline{Y} - Y\end{bmatrix}\mathbf{a} - k_d s) && \text{(6.3b)}\\
&= s(\begin{bmatrix}\ddot{\theta}_r - \ddot{\theta}_r & \dot{\theta} - \dot{\theta}_r\end{bmatrix}\mathbf{a} - k_d s) && \text{(6.3c)}\\
&= s(\begin{bmatrix}0 & s\end{bmatrix}\mathbf{a} - k_d s) && \text{(6.3d)}\\
&= s(-a_2 s - k_d s) = -(k_d + a_2)s^2 < 0, && \text{(6.3e)}
\end{aligned}$$

since a_2 is assumed positive. We refer to (6.3a) as $\dot{V}_1'$ to distinguish it from $\dot{V}_1$ in (5.30a). This does not change the stability properties established by the control law in (5.29), as $\dot{V}_1'$ is still negative definite in s. To handle the adaptive case, we augment the *LFC* to also incorporate the estimation error, as previously done in (5.34). In differentiating it, we restart from (5.28b), since the calculations have already been carried out, and simply add the new term introduced, getting

$$\dot{V}_2(s, \widetilde{\mathbf{a}}) = s(-Y\mathbf{a} + \overline{u}_{\phi_0}) - \widetilde{\mathbf{a}}^T \Gamma^{-1} \dot{\hat{\mathbf{a}}}. \tag{6.4a}$$

Selecting now the control law in (6.2) and inserting it into (6.4a) gives

$$\dot{V}_2'(s,\widetilde{\mathbf{a}}) = s(-Y\mathbf{a}+\overline{Y}\hat{\mathbf{a}}-k_d s)-\widetilde{\mathbf{a}}^T\Gamma^{-1}\dot{\hat{\mathbf{a}}} \tag{6.5a}$$

$$= s(-Y\mathbf{a}\ \underbrace{+Y\hat{\mathbf{a}}-Y\hat{\mathbf{a}}}_{\text{add and subtract}}\ +\overline{Y}\hat{\mathbf{a}}-k_d s)-\widetilde{\mathbf{a}}^T\Gamma^{-1}\dot{\hat{\mathbf{a}}} \tag{6.5b}$$

$$= s(\underbrace{-Y\mathbf{a}+Y\hat{\mathbf{a}}}_{-Y\widetilde{\mathbf{a}}}+\underbrace{\overline{Y}\hat{\mathbf{a}}-Y\hat{\mathbf{a}}}_{(\overline{Y}-Y)\hat{\mathbf{a}}}-k_d s)-\widetilde{\mathbf{a}}^T\Gamma^{-1}\dot{\hat{\mathbf{a}}} \tag{6.5c}$$

$$= s(-Y\widetilde{\mathbf{a}}+\begin{bmatrix}0 & -s\end{bmatrix}\hat{\mathbf{a}}-k_d s)-\widetilde{\mathbf{a}}^T\Gamma^{-1}\dot{\hat{\mathbf{a}}} \tag{6.5d}$$

$$= -sY\widetilde{\mathbf{a}}-\hat{a}_2 s^2-k_d s^2-\widetilde{\mathbf{a}}^T\Gamma^{-1}\dot{\hat{\mathbf{a}}}, \tag{6.5e}$$

where we've used that $Y-\overline{Y}=[0,\ s]$ from (6.3b). Selecting now the adaptation law

$$\boxed{\dot{\hat{\mathbf{a}}}=-\Gamma\overline{Y}^T s} \tag{6.6}$$

and inserting it into (6.5e) leads to

$$\dot{V}_2'(s,\widetilde{\mathbf{a}}) = -sY\widetilde{\mathbf{a}}-\hat{a}_2 s^2-k_d s^2+\widetilde{\mathbf{a}}^T\overline{Y}^T s \tag{6.7a}$$

$$= -s(Y-\overline{Y})\widetilde{\mathbf{a}}-\hat{a}_2 s^2-k_d s^2 \tag{6.7b}$$

$$= -s\begin{bmatrix}0 & s\end{bmatrix}\widetilde{\mathbf{a}}-\hat{a}_2 s^2-k_d s^2 \tag{6.7c}$$

$$= \underbrace{-\widetilde{a_2}s^2-\hat{a}_2 s^2}_{-a_2 s^2}-k_d s^2 \tag{6.7d}$$

$$= -(k_d+a_2)s^2\leq 0, \tag{6.7e}$$

which, as before, is negative semi-definite and preserves the same stability properties. We summarize below the changes brought to the adaptive control scheme.

$$\boxed{\begin{aligned}
&\text{control law} && \phi_0=\frac{1}{v_t}\left(\overline{u}_{\phi_0}-\frac{\overline{\mathbf{e}}^T\overline{\boldsymbol{\phi}_{ref}}}{N-1}\right),\quad \overline{u}_{\phi_0}=\overline{Y}\hat{\mathbf{a}}-k_d s && (6.8a)\\
&\text{adaptive law} && \dot{\hat{\mathbf{a}}}=-\Gamma\overline{Y}^T s, && (6.8b)\\
&\text{regression matrix} && \overline{Y}=[\ddot{\theta}_r\ \dot{\theta}_r] && (6.8c)
\end{aligned}}$$

This directly addresses and mitigates the undesired lack of "sufficient" excitation introduced by filtering $\overline{\theta}$, and also tackles problem 3 by replacing the direct high-gain feedback loop on $\dot{\overline{\theta}}$ with the richer, more standard augmented reference velocity signal $\dot{\theta}_r$, thereby rescuing the second estimate $\hat{a}_2$ from parameter freezing.

6.2.3 Leakage Modification

The residual ripple remaining from filtering $\overline{\theta}$, along with the renewed growth of the second estimate, leaves us with the final problem 2b (**Constantly perturbed**

adaptation) yet to be solved; that is, the estimates slowly grow unbounded, even when the control scheme achieves perfect tracking performance (as observed in numerical simulations). A common solution when dealing with unmodeled dynamics (as in our case) and/or suspected parameter divergence is to introduce what's called in the literature a *leakage modification*, i.e., adding to the adaptive law a *leakage term* $-\Gamma\epsilon\hat{\mathbf{a}}$, thereby modifying it to:

$$\boxed{\dot{\hat{\mathbf{a}}} = -\Gamma\overline{Y}^T s - \Gamma\epsilon\hat{\mathbf{a}},} \tag{6.9}$$

where $\epsilon > 0$ is a damping factor that regulates the leakage rate. This acts like a "leak" in the integrator, preventing $\hat{\mathbf{a}}$ from drifting to infinity when the robot is moving straight and no new steering information is provided to the adaptive system.

If, instead of selecting the adaptation law as in (6.6), we select it as in (6.9), then (6.5e), instead of ending as in (6.7e), acquires an additional term and becomes:

$$\begin{aligned}
\dot{V}_2''(s,\tilde{\mathbf{a}}) &= -sY\tilde{\mathbf{a}} - \hat{a}_2 s^2 - k_d s^2 - \tilde{\mathbf{a}}^T\Gamma^{-1}\dot{\hat{\mathbf{a}}} && \text{(6.10a)}\\
&= \underbrace{-sY\tilde{\mathbf{a}} - \hat{a}_2 s^2 - k_d s^2 - \tilde{\mathbf{a}}^T\Gamma^{-1}\Big(-\Gamma\overline{Y}^T s}_{-(k_d+a_2)s^2} - \Gamma\epsilon\hat{\mathbf{a}}\Big) && \text{(6.10b)}\\
&= -(k_d+a_2)s^2 + \tilde{\mathbf{a}}^T\epsilon\hat{\mathbf{a}}, && \text{(6.10c)}
\end{aligned}$$

which is no longer negative semi-definite, due to the undefined term $\tilde{\mathbf{a}}\epsilon\hat{\mathbf{a}}$. Note also that we denote the modified $\dot{V}_2'$ as $\dot{V}_2''$ to distinguish it from the previous derivation. Since $\tilde{\mathbf{a}} = \mathbf{a} - \hat{\mathbf{a}}$, we can rewrite (6.10c) as

$$\begin{aligned}
\dot{V}_2''(s,\tilde{\mathbf{a}}) &= -(k_d+a_2)s^2 + \tilde{\mathbf{a}}^T\epsilon(\mathbf{a}-\tilde{\mathbf{a}}) && \text{(6.11a)}\\
&= \underbrace{-(k_d+a_2)s^2 - \tilde{\mathbf{a}}^T\epsilon\tilde{\mathbf{a}}}_{<0} + \tilde{\mathbf{a}}^T\epsilon\mathbf{a}, && \text{(6.11b)}
\end{aligned}$$

which would be negative definite if not for the last term. For this term, we know that $\mathbf{a} > 0$ and $\epsilon > 0$, but we have no guarantees on the sign of $\tilde{\mathbf{a}}$. We can handle this last term by applying *Cauchy-Schwarz* inequality, using that

$$\tilde{\mathbf{a}}^T\epsilon\mathbf{a} \leq \epsilon\|\tilde{\mathbf{a}}^T\mathbf{a}\| \leq \epsilon\|\tilde{\mathbf{a}}\|\|\mathbf{a}\|. \tag{6.12}$$

Applying this inequality to (6.11b) yields

$$\dot{V}_2''(s,\tilde{\mathbf{a}}) = -(k_d+a_2)s^2 - \tilde{\mathbf{a}}^T\epsilon\tilde{\mathbf{a}} + \tilde{\mathbf{a}}^T\epsilon\mathbf{a} \tag{6.13a}$$

$$\leq -(k_d+a_2)s^2 - \underbrace{\tilde{\mathbf{a}}^T\epsilon\tilde{\mathbf{a}}}_{\epsilon\|\tilde{\mathbf{a}}\|^2} + \epsilon\|\tilde{\mathbf{a}}\|\|\mathbf{a}\| \tag{6.13b}$$

$$\leq -(k_d+a_2)s^2 - \epsilon\|\tilde{\mathbf{a}}\|^2 + \epsilon\|\tilde{\mathbf{a}}\|\|\mathbf{a}\| \tag{6.13c}$$

$$\leq -(k_d+a_2)s^2 - \epsilon(1\underbrace{-\sigma)\|\tilde{\mathbf{a}}\|^2}_{\text{subtract}} - \epsilon\underbrace{\sigma\|\tilde{\mathbf{a}}\|^2}_{\text{add}} + \epsilon\|\tilde{\mathbf{a}}\|\|\mathbf{a}\| \tag{6.13d}$$

$$\leq -(k_d+a_2)s^2 - \epsilon\underbrace{(1-\sigma)}_{>0\,\forall\,\sigma<1}\|\tilde{\mathbf{a}}\|^2 - \epsilon\|\tilde{\mathbf{a}}\|\underbrace{\Big(\sigma\|\tilde{\mathbf{a}}\| - \|\mathbf{a}\|\Big)}_{>0\,\forall\,\|\tilde{\mathbf{a}}\|>\frac{\|\mathbf{a}\|}{\sigma},\,\sigma>0}. \tag{6.13e}$$

Define $W_3(s,\tilde{\mathbf{a}})$ as

$$W_3(s,\tilde{\mathbf{a}}) = -(k_d+a_2)s^2 - \epsilon(1-\sigma)\|\tilde{\mathbf{a}}\|^2. \tag{6.14}$$

Then:

$$W_3(0,0) = 0, \tag{6.15a}$$

$$W_3(s,\tilde{\mathbf{a}}) > 0 \qquad \forall\,(s,\tilde{\mathbf{a}}) \neq (0,0). \tag{6.15b}$$

Inserting W_3 in (6.13e), yields

$$\dot{V}_2''(s,\tilde{\mathbf{a}}) \leq W_3(s,\tilde{\mathbf{a}}) - \epsilon\|\tilde{\mathbf{a}}\|\Big(\sigma\|\tilde{\mathbf{a}}\| - \|\mathbf{a}\|\Big) \tag{6.16a}$$

$$\leq W_3(s,\tilde{\mathbf{a}}) \qquad \forall\,\|\tilde{\mathbf{a}}\| > \|\mathbf{a}\|/\sigma, \quad \sigma\in(0,1) \tag{6.16b}$$

and by Theorem 5 we can conclude that the error variable $(s(t),\tilde{\mathbf{a}}(t))$ is *GUUB* (see Definition 1), with *ultimate bound* b that can be found as

$$b = \alpha_1^{-1}(\alpha_2(\|\mathbf{a}\|/\sigma)), \qquad \sigma\in(0,1), \qquad \alpha_1,\alpha_2 \text{ as in (3.9a)}. \tag{6.17}$$

When the errors $(s,\tilde{\mathbf{a}})$ are large, i.e., when $\|\tilde{\mathbf{a}}\| > \|\mathbf{a}\|/\sigma$, the negative terms in $\dot{V}_2''$ dominate the positive term, so that $\dot{V}_2''$ is upper bounded by the negative definite function W_3. Consequently, the total energy V_2 decreases, and the errors shrink until they enter a small neighborhood around the origin, defined by the ultimate bound b, within which $\dot{V}_2''$ may become zero or positive. Once inside this neighborhood, the system's trajectories remains confined, and the errors are guaranteed to stay within the *ultimate bound* b.

The leakage-modification introduced into the adaptive law, with a properly tuned ϵ, significantly mitigates the parameter drift we were experiencing, showing convergence and boundedness of the parameter estimates, thereby definitively resolving issue 2b (**Constantly perturbed adaptation**).

6.2.4 Over-reliance on feedback terms

A final phenomenon that we should be aware of is a well-known behavior in adaptive control, often interpreted as the controller becoming *"lazy"*. The underlying mechanism is that the robust feedback term ($-k_d s$) is so effective at suppressing errors that it drives the tracking error s to 0 very easily, leaving little need for adaptation of the estimates. However, the adaptive component of the law, ($-\Gamma \overline{Y}^\top s$), requires a nonzero error to update the parameter estimates. Consequently, when s approaches zero, adaptation effectively vanishes. At this point, the leakage-modification term ($-\epsilon \hat{\mathbf{a}}$) – introduced to mitigate parameter drift – gradually drives the now "inactive" and *unnecessary* estimates toward zero. This phenomenon requires careful tuning of the k_d term: it should be low enough to rely on the estimates for performance, but large enough to provide sufficient feedback to counter external disturbances if needed.

6.2.5 Revised Adaptive Heading Controller

We summarize here and join together all the modifications made to the adaptive heading scheme introduced in the previous sections, and refer to it as the *revised* adaptive heading controller.

filtering	$\overline{\theta} \underset{\xi,\omega_n}{\longrightarrow} \theta, \dot{\theta}$	(6.18a)
filter parameters	$\xi \in (0,1], \quad \omega_n > 0$	(6.18b)
control law	$\phi_0 = \frac{1}{v_t}\left(\overline{u}_{\phi_0} - \frac{\overline{\mathbf{e}}^T \overline{\boldsymbol{\phi}_{ref}}}{N-1}\right), \quad \overline{u}_{\phi_0} = \overline{Y}\hat{\mathbf{a}} - k_d s$	(6.18c)
adaptive law	$\dot{\hat{\mathbf{a}}} = -\Gamma \overline{Y}^T s - \Gamma \epsilon \hat{\mathbf{a}}, \quad \hat{\mathbf{a}} = [\hat{a}_1 \ \hat{a}_2]^T$	(6.18d)
parameter transformation	$a_1 = \frac{1}{\lambda_2}, \quad a_2 = \frac{\lambda_1}{\lambda_2}$	(6.18e)
filtered tracking error	$s = \dot{\widetilde{\theta}} + \Lambda \widetilde{\theta} = \dot{\theta} - \dot{\theta}_r$	(6.18f)
regression matrix	$\overline{Y} = [\ddot{\theta}_r \ \dot{\theta}_r]$	(6.18g)
reference trajectory	$\dot{\theta}_r = \dot{\theta}_{ref} - \Lambda \widetilde{\theta}$	(6.18h)
tuning values	$k_d > 0, \quad \Gamma = \Gamma^T > 0, \quad \epsilon > 0, \quad \Lambda > 0.$	(6.18i)

Figure 6.1 shows a detailed block-diagram of the overall adaptive Locomotion controller implemented for the *complex model*.

6.3 Simulation results

The controller in (6.18) can be implemented as in Code listing 6.1, where a little arrow to the right (`% <--`) indicates the lines changed from Code listing 5.5.

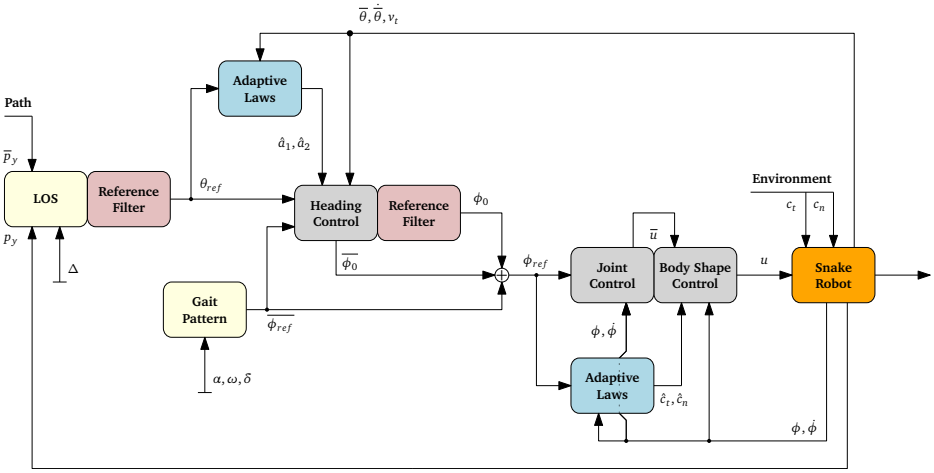


Figure 6.1: Full schematic of the adaptive locomotion controller for the complex model, including the body shape controller, the revised adaptive heading controller, the reference filters, and the LOS guidance module.

Code listing 6.1: Revised adaptive heading controller: `rev_adap_heading`

```
%% revised adaptive heading controller --------------------------

function [phi0, phi0_bar] = rev_adap_heading(t,x,yref,cond)
    global Nl lookahead_dist ref e_bar;
    global l1_hat l2_hat a_hat dt;

    % initialization
    persistent thref_filt thref_dot thref_ddot;
    persistent head_filt head_dot_filt;                                 % <---
    if isempty(thref_filt)
        [thref_filt, thref_dot, thref_ddot] = deal(0);
        [head_filt, head_dot_filt] = deal(0);                           % <---
    end

    %TUNING ---------
    Lambda = 0.1;
    Kd = 0.01;
    Gamma = diag([10,1]);
    epsilon = 0.001;                                                    % <---
    wc = 1.1;                                                           % <---
    %-----------------

    % LOS
    py = x(Nl+2);
    thref = -atan2(py - yref, lookahead_dist);

    %TUNING ------
    xi = 1;
    deltaT = 4;
    %-------------

    [thref_filt, thref_dot, thref_ddot] = ...
        ref_filter([thref_filt, thref_dot, thref_ddot]', thref, xi, deltaT);

```

```
    heading = mean(x(1:Nl));                                                % <---
    %heading_dot = mean(x(Nl+3:end-2));                                     % <---
    vt = x(end-1)*cos(heading) + x(end)*sin(heading);                       % <---

    deltaT = 2*pi/wc;                                                       % <---
    [head_filt, head_dot_filt] = ...
        ref_filter_2nd([head_filt, head_dot_filt]', heading, 1,deltaT);     % <---

    % filtered tracking error
    s = (head_dot_filt - thref_dot) + Lambda*(head_filt - thref_filt);      % <---
    thetaR_dot = thref_dot - Lambda*(head_filt - thref_filt);               % <---
    thetaR_ddot = thref_ddot - Lambda*(head_dot_filt - thref_dot);          % <---

    Y = [thetaR_ddot, thetaR_dot];                                          % <---
    a_hat_dot = @(a,s) -Gamma*Y'*s - Gamma*epsilon*a;                       % <---

    if cond
        a_hat = euler_step(a_hat, s, a_hat_dot, dt);

        try
            l2_hat = 1/a_hat(1);        %try to avoid 1/0
            l1_hat = a_hat(2)*l2_hat;
        end

        u_bar = Y*a_hat - Kd*s;
        phi0 = 1/vt * u_bar;
    else
        phi0 = 0;
    end

    [phiref_bar, phiref_bar_dot, phiref_bar_ddot] = ref(t);
    phi0_bar(1) = -e_bar'*phiref_bar/(Nl-1);        %phi0_bar
    phi0_bar(2) = -e_bar'*phiref_bar_dot/(Nl-1);    %phi0_bar_dot
    phi0_bar(3) = -e_bar'*phiref_bar_ddot/(Nl-1);   %phi0_bar_ddot
    phi0 = -phi0;                                                           % <---
end

...

% 2rd order reference filter ---------------------------------

function [x, xdot] = ref_filter_2nd(x, r, xi, deltaT)
    global dt;

    wc = 2*pi/deltaT;
    F = [0, 1; -wc^2, -2*xi*wc];
    tf = @(x,r) F*x + [0; wc^2]*r;

    z = euler_step(x, r, tf, dt);
    x=z(1); xdot=z(2);
end
```

We validate the performance of the final control architecture in `MATLAB R2023b` to demonstrate the effectiveness of the proposed modification. The complete simulation settings, including all parameters of the snake, environment, and tunings of the controllers, are reported here below in their full extent.

The snake robot's physical parameters remain those used throughout this work and have not been modified. For completeness, they are summarized in Table 6.1.

Table 6.1: Snake robot physical parameters.

Parameter	**Symbol**	**Value**	**Unit**
Number of links	N	10	–
Length of each link	$2l$	$2 \cdot 0.09$	m
Mass of each link	m	1.56	kg
Moment of inertia of each link	$J = \frac{1}{3}ml^2$	$\simeq 0.0042$	kg·m^2

We extended the simulation duration to allow for the adaptation of the rotational drag parameters, which operate on a slower timescale within the control hierarchy, and also due to the reduced propulsion because of the reduced gait pattern amplitude (see Table 6.3). This also provided sufficient time for the path-following architecture to demonstrate its tracking performance. Refer to Table 6.2 for the simulation parameters.

Table 6.2: Simulation parameters.

Parameter	**Symbol**	**Value**	**Unit**
Simulation time	T	1000	s
Integration solver	`int_method`	`RK4`	–
Time-step	dt	0.01	s
Robot's initial conditions	$\mathbf{x}(0)$	$[0, \ldots, 0]^T \in \mathbb{R}^{2N-2}$	–

The gait parameters are chosen as in Section 5.2.2 to respect the assumption of small link angles and to maintain fidelity with the *CO-model*, and are summarized in Table 6.3.

Table 6.3: Reference gait pattern parameters.

Parameter	**Symbol**	**Value**	**Unit**
Amplitude	α	$16\pi/180$	rad
Angular frequency	ω	$120\pi/180$	rad/s
Phase shift between successive joints	δ	50	rad

We simulate the system on a bi-partitioned terrain, with c_t and c_n as in Table 6.4 changing at $x = 15\,\text{m}$. For simplicity and consistency with the model, the parameter change is applied simultaneously to all links when the center of mass (p_x, p_y) crosses this boundary. Rather than using a simple piecewise-linear reference path, which would not provide persistent excitation, we select a sinusoidal path that is sufficiently rich to both challenge the heading controller and support adaptation of the heading estimation parameters. See Table 6.4.

The *look-ahead distance* Δ of the LOS guidance module is chosen as the outstretched length of the snake, while we recall that $y_{ref}(t)$ is determined at each

Table 6.4: Environment parameters and reference path.

Parameter	**Symbol**	**Value**		**Unit**
		for $x < 15\ m$	for $x \geq 15\ m$	
Tangential friction coefficient	c_t	0.5	4	Ns/m
Normal friction coefficient	c_n	3	17	Ns/m
Reference path	$y_{\text{ref}}(x)$	$4\sin(0.02\,x) + \sin(0.3\,x)$		m

instant from the position of the CM of the snake robot, projected forward by the *look-ahead distance*, as described in Table 6.5.

Table 6.5: LOS guidance module parameters.

Parameter	**Symbol**	**Value**	**Unit**
Look-ahead distance	Δ	$N \cdot 2l = 1.8$	m
Reference target	$y_{\text{ref}}(t)$	$y_{\text{ref}}(p_x(t) + \Delta)$	m

The tunings of the body-shape adaptive controller are reported in Table 6.6, where the adaptation gains Γ have been slightly increased to compensate for the reduced propulsion (i.e., lower speed and weaker excitation power).

Table 6.6: Adaptive body-shape controller parameters.

Parameter	**Symbol**	**Value**	**Unit**
Control gains	k_p	30	–
	k_d	37	–
Adaptive gain	Γ	$\begin{bmatrix} 36 & 0 \\ 0 & 45 \end{bmatrix}$	–
Initial estimates	$\hat{c}_t(0)$	$c_{\min}$	Ns/m
	$\hat{c}_n(0)$	$c_{\min}$	Ns/m
Parameter projection bounds	$c_{\min}$	0.01	Ns/m
	$c_{\max}$	100	Ns/m

For the adaptive heading controller, the feedback gain k_d is chosen small enough to avoid the "lazy-fication" phenomenon, while the leakage rate ϵ is selected large enough to prevent parameter drift, yet not so large as to significantly alter the estimation dynamics. Refer to Table 6.7 for all the tunings.

In the choice of the natural frequency of the heading filter from Section 6.2.1, ω_n must be set sufficiently small to attenuate oscillations at the gait frequency ω, yet large enough to preserve the bandwidth Λ of the *filtered tracking error* first-order system $s = \dot{\widetilde{\theta}} + \Lambda\widetilde{\theta}$. Formally, this can be expressed as

$$\Lambda \ll \omega_n \ll \omega. \tag{6.19}$$

Given the selected parameters and gains, we have $\Lambda = 0.1$ and $\omega \simeq 2.1\,\text{rad/s}$.

Table 6.7: Revised adaptive heading controller parameters.

Parameter	Symbol	Value	Unit
Feedback control gain	k_d	0.01	–
Filtered tracking error tuning value	Λ	0.1	–
Adaptation gain	Γ	$\begin{bmatrix} 10 & 0 \\ 0 & 1 \end{bmatrix}$	–
Leakage rate	ϵ	0.001	–
Forward velocity threshold	$v_{t,\min}$	0.02	m/s
Joint tracking error threshold	$\|\tilde{\boldsymbol{\phi}}\|_{\min}$	0.1	rad
Initial estimates	$\hat{\mathbf{a}}(0)$	$[0,0]^T$	–

Therefore, the condition in (6.19) cannot be satisfied exactly. A reasonable compromise is to set it midway, at $\omega_n = 1.1\,\text{rad/s}$.

For the filters damping ratio, we again choose $\xi = 1$ to avoid overshoot in the step response. The reference filter for ϕ_0 is instead the third-order hybrid reference filter introduced in Section 5.3.3. Refer to Table 6.8 for all the tunings.

Table 6.8: Reference filters settings.

Filter	Parameter	Symbol	Value	Unit
LOS reference angle $\overline{\theta_{\text{ref}}} \rightarrow \theta_{\text{ref}}, \dot{\theta}_{\text{ref}}, \ddot{\theta}_{\text{ref}}$	Period	ΔT	4	s
	Natural frequency	ω_n	$2\pi/\Delta T$	rad/s
	Damping ratio	ξ	1	–
ϕ_0 hybrid filter $\phi_0 \longrightarrow \phi_{0,\text{filt}}, \dot{\phi}_0, \ddot{\phi}_0$	Period	ΔT	4	s
	Natural frequency	ω_n	$2\pi/\Delta T$	rad/s
	Damping ratio	ξ	1	–
Heading filter $\overline{\theta} \longrightarrow \theta, \dot{\theta}$	Period	ΔT	$2\pi/\omega_n$	s
	Natural frequency	ω_n	1.1	rad/s
	Damping ratio	ξ	1	–

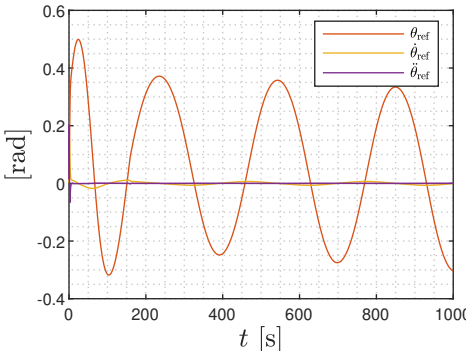
The results of the simulation are shown in Figures 6.2 and 6.3 and discussed here. As can be seen from Figure 6.2c, the estimate $\hat{\mathbf{a}}$ does not drift indefinitely as desired and the whole control architecture revealed to work as expected. Instead, it seems to stabilize around two steady-state values, around which it slightly oscillates because of the *leakage modification* that is constantly pulling them back. This behavior is most noticeable for the estimate $\hat{a}_2$, likely because it is a combination of the two artificial parameters λ_1 and λ_2, and is therefore more sensitive to the issues mentioned in Section 6.2.

For reference, the corresponding estimates $\hat{\lambda}_1$ and $\hat{\lambda}_2$ are shown in Figure 6.2d. The final values they reach are completely consistent with what was conjectured by Liljebäck *et al.* [1], which proposed $\lambda_1 = 0.5$ and $\lambda_2 = 20$ in its original work. Our final estimated values, $\hat{\lambda}_1 \simeq 1.08$ and $\hat{\lambda}_2 \simeq 16.63$, are of the same order of magnitude, suggesting they are very promising results.

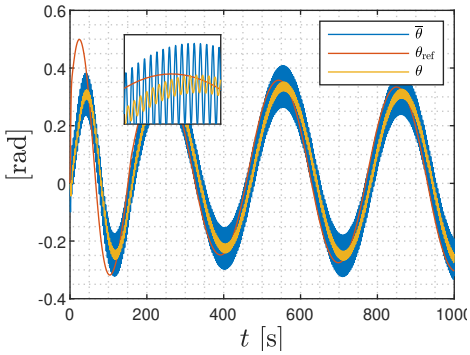
Figure 6.2b shows how the filtered heading θ is, as expected, still oscillating, but significantly less than the original heading $\overline{\theta}$ defined as the average of the link angles, and the ripple appears to be completely manageable as a remaining modeling error.

In Figures 6.3a and 6.3b are instead shown the results concerning the body-shape adaptive controller: the estimates $\hat{c}_t$ and $\hat{c}_n$, shown to converge to their true values for both the terrains, and the gait tracking errors, overcoming the perturbation threshold only during friction changes of the terrains.

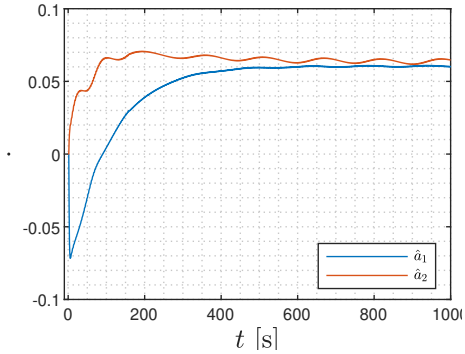
Finally, Figure 6.3c shows the path-following capabilities of the overall control architecture, revealing excellent performance after an initial transient.



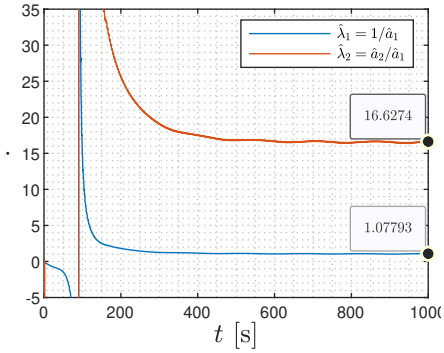
(a) Filtered reference angles from the LOS guidance module. As can be noticed, the majority of the contribution is given by θ_{ref}, due to the slow variation of the sinusoidal path $y_{\text{ref}}(x)$.



(b) Time evolution of the heading $\overline{\theta}$ and the filtered heading θ compared to the reference angle θ_{ref}. The angular velocities are omitted for improved clarity of the picture, as they always oscillate around zero.

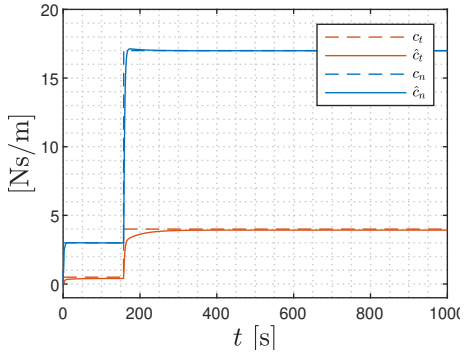


(c) Time evolution of the estimate $\hat{a}$.

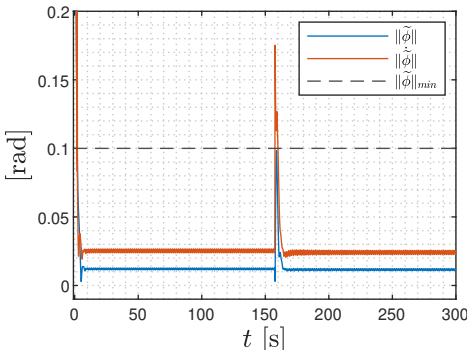


(d) Time evolution of the estimates $\hat{\lambda}_1$ and $\hat{\lambda}_2$. Highlighted the final value reached at $t = 1000$ [s].

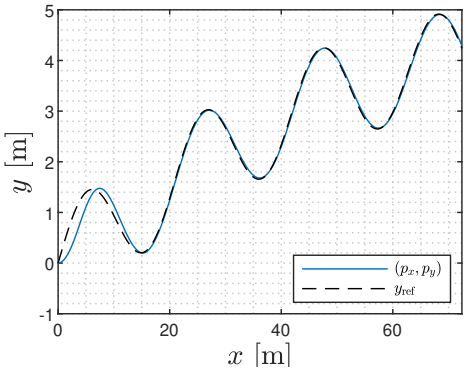
Figure 6.2: Simulation results of the overall final control architecture (1of2)



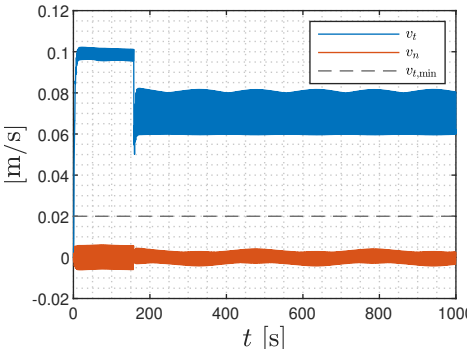
(a) Convergence of the estimate of the friction coefficients $\hat{c}_t, \hat{c}_n$ to their true values c_t, c_n.



(b) Norm of the joint tracking errors. The black dashed line is the acceptance threshold $\|\widetilde{\boldsymbol{\phi}}\|_{\min}$ needed to activate the heading adaptive controller. Zoom in $t \in [0, 300]$ [s].



(c) Trace of the CM (p_x, p_y) versus the sinusoidal path reference $y_{ref}(x)$ employed to achieve PE in the adaptive scheme.



(d) Normal velocity v_t and tangential velocity v_n. The black dashed line is the minimum forward velocity $v_{t,min}$ needed to activate the heading adaptive controller.

Figure 6.3: Simulation results of the overall final control architecture (2of2)

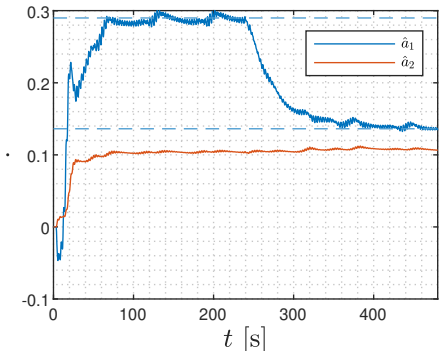
6.4 Discussion

Due to the many elements that, albeit slightly, affect the dynamics of the overall architecture – such as the reference filter delays, leakage modification, and the use of filtered heading variables – it is difficult to determine whether the estimated drag parameters can be considered fully accurate. Nonetheless, it is still possible to conduct experiments using the working adaptive heading controller as a parameter estimator to further explore how these parameters vary and to verify some of the conjectures initially formulated by Liljebäck *et al.* [1].

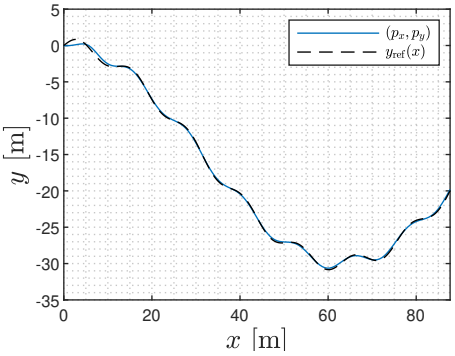
The authors suggested that λ_1 and λ_2 are functions of the friction coefficients, to which we suggest adding the influence of the gait pattern parameters. This can be intuitively explained by the fact that the snake's "maneuverability" – in the sense of its ability to track tighter curved paths – is also influenced by the forward velocity and the amplitude of the gait pattern. A wider gait amplitude, or a higher speed, leads to greater steering power.

We validated this reasoning by conducting an experiment with the original gait amplitude, $\alpha = 40°$, instead of the reduced value, $\alpha = 16\circ$, which was chosen to respect the CO-model assumptions. This showed that the estimates stabilized at a different steady-state value, $\lambda_1 \simeq 0.14$ and $\lambda_2 \simeq 7.15$. By analyzing these results from a purely physical perspective by reasoning on the CO-model, the smaller coefficients indicate that the robot appears to be subject to reduced rotational drag friction in this second experiment. This translates to greater steering "freedom" and "maneuverability", thereby confirming our conjecture on the dependence of λ_1 and λ_2 from the gait parameters. Increasing the gait amplitude also revealed a noticeable difference between the two terrains, as the estimates now tend to stabilize around different values (see Figure 6.4a and recall that $a_2 = \lambda_1/\lambda_2$, so when $a_1 = 1/\lambda_2$ changes but a_2 stays fixed, λ_1 still varies accordingly), confirming the conjecture of Liljebäck *et al.* [1] that both depends on the friction coefficients.

This also revealed another important result: the heading controller continues to operate effectively (see the performance in Figure 6.4b) even when losing the assumption of small angles, as the gait amplitude used (40°) far exceeds the theoretical limitation assumed of 16°.



(a) Time evolution of the estimates $\hat{a}_1$ and $\hat{a}_2$. Notice how they tend to stabilize at different values depending on the friction coefficient values, which change around $t = 250$ [s].



(b) Trace of the CM (p_x, p_y) versus the sinusoidal path reference $y_{ref}(x)$, which is now stressed a little bit more to challenge the heading controller choosing $y_{\text{ref}}(x) = -15 + 15\sin(0.05x + \pi/2) + \sin(0.5x)$.

Figure 6.4: Simulation results with increased gait amplitude.

Chapter 7

Conclusion

7.1 Summary

This thesis investigated adaptive locomotion control for planar snake robots in scenarios where key model parameters are uncertain or unavailable. Two adaptive schemes were developed: (i) an adaptive body-shape controller compensating for unknown or varying friction coefficients, and (ii) an adaptive heading controller estimating the *artificial* drag parameters of the simplified *CO-model*, which suffers from structural mismatch with the complex model.

The closed-loop stability of the system under the proposed controllers was rigorously analyzed using Lyapunov-based methods, and the results of the theoretical analysis were validated through extensive simulations on both simplified and complex models. Results confirmed that the body-shape controller ensures convergence of joint tracking errors, maintaining desired propulsion despite frictional uncertainty, while the heading controller adapts to unknown drag parameters to enable accurate path-following. The overall architecture, integrating also the LOS guidance module and the reference filters, demonstrated stable and robust performance. Nevertheless, some challenges arose when transferring the heading controller to the complex model, which required structural adjustments to account for the different dynamics. In particular, the oscillatory dynamics of heading – not accounted for in the design of the heading controller, but present in both the complex and the real physical model – necessitated filtering of the heading state variable and the introduction of a leakage modification to prevent parameter drift.

The development process followed a progressive structure, reflected in the organization of this thesis. We began by introducing an adaptive body-shape controller for the complex model to address frictional uncertainty in Chapter 4. Next, we explored solutions to achieve path-following control by embedding the body-shape controller within a cascaded architecture. This effectively highlighted the limitations imposed by the complexity of the model, which prevented the direct design of a fully *model-based* heading controller. To overcome this, we tempor-

arily shifted to work on the *CO-model*, which, however, suffers from structural mismatch with the complex model and relies on two *artificial* parameters for modeling rotational dynamics – parameters that are unavailable in real-world applications. To address this we developed an adaptive heading controller capable of estimating these rotational drag coefficients in Chapter 5. Finally, the scheme was reintegrated into the complex model, resulting in the overall adaptive locomotion control architecture presented and validated in Chapter 6.

Overall, this work contributes to bridging the gap between two mathematical models that are commonly used in the snake robot research community. It demonstrates the potential of adaptive strategies to handle uncertainty not only in physical parameters, such as terrain friction, but also in *artificial* coefficients, such as drag parameters, and provides a theoretical foundation for implementing these controllers in real-world snake robot locomotion. Furthermore, it offers insights on model transfer and structural mismatch, highlighting practical challenges and solutions when adapting controllers in the presence of *unmodeled dynamics*. Moreover, it empirically confirms the dependence of drag parameters on frictional coefficients, as conjectured by Liljebäck *et al.* [1], and offer a tool to further study parameter matching between the two models, by use of the adaptive heading controller as parameter estimator.

7.2 Future work

While this thesis presents a robust framework for adaptive locomotion, several avenues for future research exist:

- **Formal stability analysis:** Some practical solutions developed to address the challenges of applying the CO-model-based controller to the complex model were largely based on experimental observations and analogies from the literature. A formal theoretical analysis of the whole system including also the reference filters as part of the control chain would be a valuable and significant next step.
- **Velocity control:** The original control architecture which this work is based on does not include velocity as a control variable, allowing for a simpler control structure as a foundation for the adaptive extension. We believe that extending our work to include maneuvering control could further complete the desired locomotion control objectives, for example, by referring to the work of Mohammadi *et al.* [9] and Chitikena *et al.* [5].
- **Underwater locomotion validation:** The simulations and models used in this work focused on ground locomotion. Extending them to underwater locomotion – a domain of major industrial relevance – remains an ongoing effort. In particular, implementation in a high-fidelity Computational Fluid Dynamics (CFD) simulator from the snake robot research group at Norwegian University of Science and Technology (NTNU) is an ongoing activity but not reported here due to time constraints. Such an extension would in-

troduce additional challenges related to fluid dynamics, added mass, and other *unmodeled* effects.

- **Physical experimental validation:** The proposed adaptive schemes were validated only through numerical simulations. Due to time constraints and the original project planning, implementation on a physical snake robot was not pursued. Experimental deployment in a real-world setting would represent the ultimate validation of the controllers effectiveness, robustness, and applicability.
- **Model-matching by use of the adaptive heading controller:** It could be interesting to analyze the model matching between the complex model and the CO-model by means of the *drag parameters*, by use of the adaptive controller as an estimator rather than as a controller. This approach could provide valuable insights into the correspondence between the two models and help quantify the limitations and validity of the simplified CO-model.

7.3 Personal thoughts

I am very satisfied with the results obtained in this thesis, as they not only validate my work as... a meaningful work, but also taught me two important lessons. First, I realized that a future in pure academic research, where much of the work has no tangible outcome, may not align with my personal interests; I would prefer to see and interact with concrete results.

Second, while working on this, especially when approaching the panorama of snake robots for the first time, I noticed a gap in current teaching practices: too often, simulations are reduced to inspecting lines on a graph, such as the y-position of a snake robot, without providing an intuitive understanding of the actual motion of the snake robot. This can often result in issues hard to debug, being the user blind to the overall scenario, since he's looking only to specific variables, and often, neither in an intuitive way.

Motivated by this, I am developing a real-time GUI simulator in python that implements the results of this thesis as an educational tool. This simulator allows both the research group and newcomers to visualize and experiment with snake locomotion directly, making the effects, for example, of gait parameters on motion immediately tangible and comprehensible.

A more utopian thought is to evolve this simulator as a generalized dynamic tool with separate modules such as visualization, simulator, and controllers, which is lightweight and customizable, allowing both teachers and students to interact with it. For example, when developing a controller for a particular system, students will have access only to the control module, while the simulator and the dynamics of the system remains hidden, providing a clear and structured learning experience.

Bibliography

[1] P. Liljebäck, K. Y. Pettersen, Ø. Stavdahl and J. T. Gravdahl, *Snake Robots: Modelling, Mechatronics, and Control*. Springer London, 2013.

[2] N. Li, F. Wang, S. Ren, X. Cheng, G. Wang and P. Li, 'A Review on the Recent Development of Planar Snake Robot Control and Guidance,' *Mathematics*, vol. 13, no. 2, 2025, ISSN: 2227-7390. DOI: 10.3390/math13020189. [Online]. Available: https://www.mdpi.com/2227-7390/13/2/189.

[3] K. Y. Pettersen, 'Snake robots,' *Annual Reviews in Control*, vol. 44, pp. 19–44, 2017, ISSN: 1367-5788. DOI: https://doi.org/10.1016/j.arcontrol.2017.09.006. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S1367578817301050.

[4] T. Owen, 'Biologically inspired robots: Snake-like locomotors and manipulators by shigeo hirose oxford university press, oxford, 1993, 220pages, incl. index (č40),' *Robotica*, vol. 12, no. 3, pp. 282–282, 1994. DOI: 10.1017/S0263574700017264.

[5] H. Chitikena, I. Gravdahl, K. Y. Pettersen, A. Mohammadi, F. Sanfilippo, Ø. Stavdahl and S. Ma, 'Adaptive Manoeuvring Control for Planar Snake Robots in Uncertain Friction Environments,' in *2024 American Control Conference (ACC)*, 2024, pp. 2844–2850. DOI: 10.23919/ACC60939.2024.10644564.

[6] J. Gray, 'The mechanism of locomotion in snakes,' *Journal of Experimental Biology*, vol. 23, no. 2, pp. 101–120, Dec. 1946, ISSN: 0022-0949. DOI: 10.1242/jeb.23.2.101. eprint: https://journals.biologists.com/jeb/article-pdf/23/2/101/2205775/jexbio_23_2_101.pdf. [Online]. Available: https://doi.org/10.1242/jeb.23.2.101.

[7] P. Liljebäck, K. Y. Pettersen, O. Stavdahl and J. T. Gravdahl, 'A review on modelling, implementation, and control of snake robots,' *Robotics and Autonomous Systems*, vol. 60, no. 1, 2012, ISSN: 09218890. DOI: 10.1016/j.robot.2011.08.010.

[8] P. Liljebäck, I. Haugstuen and K. Pettersen, 'Path following control of planar snake robots using a cascaded approach,' *Control Systems Technology, IEEE Transactions on*, vol. 20, pp. 111–126, Feb. 2012. DOI: 10.1109/TCST.2011.2107516.

[9] A. Mohammadi, E. Rezapour, M. Maggiore and K. Y. Pettersen, 'Maneuvering Control of Planar Snake Robots Using Virtual Holonomic Constraints,' *IEEE Transactions on Control Systems Technology*, vol. 24, no. 3, 2016, ISSN: 10636536. DOI: 10.1109/TCST.2015.2467208.

[10] J. Mukherjee, S. Mukherjee and I. N. Kar, 'Sliding Mode Control of Planar Snake Robot with Uncertainty Using Virtual Holonomic Constraints,' *IEEE Robotics and Automation Letters*, vol. 2, no. 2, 2017, ISSN: 23773766. DOI: 10.1109/LRA.2017.2657892.

[11] J. Mukherjee, I. N. Kar and S. Mukherjee, 'Adaptive sliding mode control for head-angle and velocity tracking of planar snake robot,' in *2017 Asian Control Conference, ASCC 2017*, vol. 2018-January, 2018. DOI: 10.1109/ASCC.2017.8287227.

[12] B. M. Patel and S. K. Dwivedy, 'Virtual holonomic constraints based super twisting sliding mode control for motion control of planar snake robot in the uncertain underwater environment,' *Proceedings of the Institution of Mechanical Engineers. Part I: Journal of Systems and Control Engineering*, vol. 237, no. 8, 2023, ISSN: 20413041. DOI: 10.1177/09596518231153253.

[13] A. Zhang, S. Ma, B. Li, M. Wang, X. Guo and Y. Wang, 'Adaptive controller design for underwater snake robot with unmatched uncertainties,' *Science China Information Sciences*, vol. 59, no. 5, 2016, ISSN: 18691919. DOI: 10.1007/s11432-015-5421-8.

[14] J. Mukherjee, S. Roy, I. N. Kar and S. Mukherjee, 'A double-layered artificial delay-based approach for maneuvering control of planar snake robots,' *Journal of Dynamic Systems, Measurement, and Control*, vol. 141, no. 4, p. 041 012, 2019.

[15] J. Mukherjee, S. Roy, I. N. Kar and S. Mukherjee, 'Maneuvering control of planar snake robot: An adaptive robust approach with artificial time delay,' *International Journal of Robust and Nonlinear Control*, vol. 31, no. 9, 2021, ISSN: 10991239. DOI: 10.1002/rnc.5430.

[16] G. Wang, W. Yang, Y. Shen, H. Shao and C. Wang, 'Adaptive Path Following of Underactuated Snake Robot on Unknown and Varied Frictions Ground: Theory and Validations,' *IEEE Robotics and Automation Letters*, vol. 3, no. 4, 2018, ISSN: 23773766. DOI: 10.1109/LRA.2018.2864602.

[17] H. Khalil, *Nonlinear Systems* (Pearson Education). Prentice Hall, 2002, ISBN: 9780130673893. [Online]. Available: https://books.google.it/books?id=t_d1QgAACAAJ.

[18] J.-J. E. Slotine't and W. Li, 'Composite Adaptive Control of Robot Manipulators* The performance of direct adaptive controllers can be enhanced by incorporating a prediction error on the joint torques or on the power input into the tracking-error-driven adaptation law,' Tech. Rep. 4, 1989, pp. 509–519.

[19] J. Hopsdal, D. T. Nguyen and H. Schmidt-Didlaukies, 'Adaptive Control of Underwater Snake Robot,' Ph.D. dissertation, 2019.

[20] M. W. Spong, 'Partial feedback linearization of underactuated mechanical systems,' in *IEEE International Conference on Intelligent Robots and Systems*, vol. 1, 1994. DOI: 10.1109/IROS.1994.407375.

[21] Y. L. Gu and Y. Xu, 'A normal form augmentation approach to adaptive control of space robot systems,' *Dynamics and Control*, vol. 5, pp. 275–294, 3 Jul. 1995, ISSN: 09254668. DOI: 10.1007/BF01968678/METRICS. [Online]. Available: https://link.springer.com/article/10.1007/BF01968678.

[22] M. Reyhanoglu, A. van der Schaft, N. Mcclamroch and I. Kolmanovsky, 'Dynamics and control of a class of underactuated mechanical systems,' *IEEE Transactions on Automatic Control*, vol. 44, no. 9, pp. 1663–1671, 1999. DOI: 10.1109/9.788533.

[23] E. Kelasidi, P. Liljebäck, K. Y. Pettersen and J. T. Gravdahl, 'Experimental investigation of efficient locomotion of underwater snake robots for lateral undulation and eel-like motion patterns,' *Robotics and Biomimetics 2015 2:1*, vol. 2, pp. 1–27, 1 Dec. 2015, ISSN: 2197-3768. DOI: 10.1186/S40638-015-0029-4. [Online]. Available: https://jrobio.springeropen.com/articles/10.1186/s40638-015-0029-4.

[24] P. Liljebäck, K. Y. Pettersen, Ø. Stavdahl and J. T. Gravdahl, 'A simplified model of planar snake robot locomotion,' in *2010 IEEE/RSJ International Conference on Intelligent Robots and Systems*, 2010, pp. 2868–2875. DOI: 10.1109/IROS.2010.5649110.

[25] J. Slotine and W. Li, *Applied Nonlinear Control* (Prentice-Hall International Editions). Prentice-Hall, 1991, ISBN: 9780130400499. [Online]. Available: https://books.google.it/books?id=HddxQgAACAAJ.

Appendix A

MATLAB Simulator

The simulator code used throughout this work is fully available in the project repository at `github.com/albertomors/snekloc`. An overview of its workflow is illustrated in Figure A.1, while Code listing A.1 reports in full the initial configuration, the main simulation loop, and the visualization loop for the *complex model*. The code for the *CO-model* is largely equivalent, differing only in the visualization loop – due to the alternative drawing approach – and in the definition and usage of the state variables, where link angles are replaced by joint translational coordinates, while heading θ and velocities v_t, v_n are directly available in the state vector. Post-processing plots and functions already presented earlier are omitted here.

Code listing A.1: `snek_locomotion.m`

```
clc; clf; clear all; close all;

% ======================
% SIM CONFIG
% ======================

% snake params
global Nl l m J;
Nl = 10;
l = 0.09;
m = 1.56;
J = (1/3)*m*l^2;

% sim params
global T dt N_steps;
T = 480;
dt = 0.01;
N_steps = round(T/dt);
t = linspace(0,T,N_steps);

global int_method ref;
online_vis = true;
dynamics = @dyn233;                % dyn233 / dyn241
controller = @adap;                % pd / fblin / adap
int_method = @RK4;                 % RK4 / euler_step
ref = @lat_ond;                    % lat_ond / eel_like
```

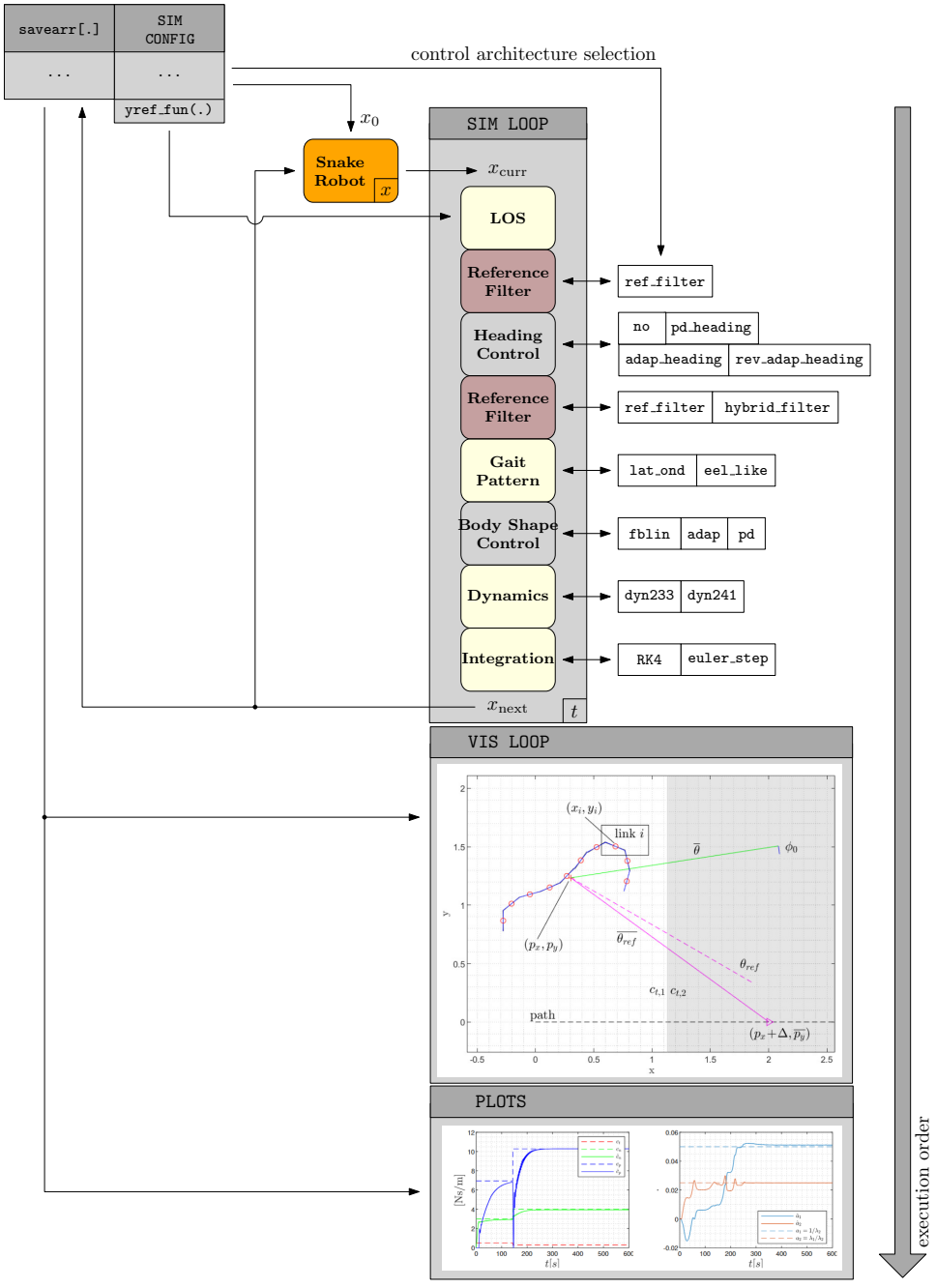


Figure A.1: Code structure

```
heading_law = @rev_adap_heading;    % no / pd_heading / rev_adap_heading
rand_init_pose = false;
rand_c_hat0 = false;
rand_l_hat0 = false;

% gait pattern
global alpha omega delta;
alpha = 40*pi/180;
omega = 1*120*pi/180;
delta = 50*pi/180;

% los settings
global lh_dist
lh_dist = 2*Nl*l;

% terrain
global ct cn c_hat maxc minc
minc = 0.01;
maxc = 100;
ct = 0.5;
cn = 3;
c_hat = [minc;minc];
if rand_c_hat0
    c_hat = rand(2,1)*(maxc - minc) + minc;
end

% lyap analysis
global v vdot;
[v, vdot] = deal([0,0]);

% path
yref_fun = @(x) -15+15*sin(0.05*x+pi/2) + 1*sin(0.5*x);
%yref_fun = @(x) 2*heaviside(x-15)

% heading controller params
global vt_min iss_min;
vt_min = 0.02;
iss_min = 0.1;
cond = false;

% CO model params
global l1_hat l2_hat a_hat
a_hat = [0;0];
[l1_hat, l2_hat] = deal(NaN);

%% initial conditions -----------------------------------

init.angs = ones(Nl,1)*0;
init.pos = [0;0];
if rand_init_pose
    init.angs = -pi/4 + rand(Nl,1)*2*pi/4;
    init.pos = [0; rand()*2];
end
init.ang_vel = zeros(Nl,1);
init.vel = [0;0];

init.x0 = [init.angs; init.pos; init.ang_vel; init.vel];
x = init.x0;

%% derived params ---------------------------------------
```

```
87
88  global nx nu e e_bar A D D_bar K V H;
89  nx = 2*Nl+4;
90  nu = Nl-1;
91  e = ones(Nl,1);
92  e_bar = ones(Nl-1,1);
93
94  %(2.34)
95  A = diag(ones(Nl, 1)) + diag(ones(Nl-1, 1), 1);
96  A = A(1:Nl-1,:);
97  D = diag(ones(Nl, 1)) + diag(-ones(Nl-1, 1), 1);
98  D = D(1:Nl-1,:);
99  D_bar = D'*inv(D*D');
100 K = A'*inv(D*D')*D;
101 V = A'*inv(D*D')*A;
102 H = triu(ones(Nl,Nl));
103
104 % savearr ------------------------------------------------
105
106 sv.x = zeros(N_steps,nx);
107 [sv.phi_ref, sv.phi_dot_ref, sv.phi, sv.phi_dot] = deal(zeros(N_steps,nu));
108 [sv.v, sv.vdot] = deal(zeros(N_steps,2));
109 sv.target = zeros(N_steps,2);
110 sv.heading = zeros(N_steps,1);
111 [sv.threfs, sv.phi0s] = deal(zeros(N_steps,4));
112 terrain.c_trues = zeros(N_steps,2)
113 [sv.c_hat, sv.l_hat, sv.a_hat] = deal(zeros(N_steps,2));
114 sv.cond = zeros(N_steps,1);
115
116 % ========================
117 % SIM LOOP
118 % ========================
119
120 last_percent = 0;
121 for k = 1:N_steps
122
123     percent = floor(k/N_steps * 100);
124     if percent > last_percent
125         fprintf('Processing...%d%%\n', percent);
126         last_percent = percent;
127     end
128
129     px = x(Nl+1);
130     if px >= 30         %change terrain
131         ct = 4;
132         cn = 17;
133     end
134     assert(cn >= ct)    %for propulsion
135
136     % control loop
137     target = [px + lh_dist, yref_fun(px+lh_dist)];
138     [heading, phi0, phi0_bar, threfs] = heading_law(t(k), x, target(2), cond);
139     [phi_ref, phi_ref_dot, phi_ref_ddot] = ref(t(k));
140     [phi_ref, phi_ref_dot, phi_ref_ddot, phi0s] = ...
141         hybrid_add_phi0(phi_ref, phi_ref_dot, phi_ref_ddot, phi0, phi0_bar);
142     u = controller(x, phi_ref, phi_ref_dot, phi_ref_ddot);
143     x = int_method(x, u, dynamics, dt);
144
145     % save stuff
146     sv.x(k,:) = x;
```

```
147     sv.phi(k,:) = D*x(1:Nl);
148     sv.phi_dot(k,:) = D*x(Nl+3:end-2);
149     sv.phi_ref(k,:) = phi_ref;
150     sv.phi_dot_ref(k,:) = phi_ref_dot;
151     sv.heading(k) = heading;
152     sv.threfs(k,:) = threfs;
153     terrain.path(k,:) = [px, yref_fun(px)];
154     sv.target(k,:) = target;
155     sv.phi0s(k,:) = phi0s;
156     terrain.c_trues(k,:) = [ct,cn];
157     sv.c_hat(k,:) = c_hat;
158     sv.a_hat(k,:) = a_hat;
159     sv.l_hat(k,:) = [l1_hat; l2_hat];
160     sv.v(k,:) = v;
161     sv.vdot(k,:) = vdot;
162     sv.cond(k) = cond;
163 
164     % tresholds to run the heading controller
165     vt = x(end-1)*cos(heading) + x(end)*sin(heading);
166     err = vecnorm(D*x(1:Nl) - phi_ref, 2, 1);
167     cond = (err <= iss_min && vt > vt_min);
168 end
169 
170 % ======================
171 % VIS LOOP
172 % ======================
173 
174 speed_mult = 1;
175 target_speed = speed_mult;
176 speed_corrected = false;
177 
178 if online_vis
179     figure(1);
180     handles.title = title('', 'FontSize', 8);
181     hold on;
182 
183     % for closing figure
184     set(gcf, 'KeyPressFcn', @(src, event) setappdata(src, 'keyPressed', event.Key));
185     setappdata(gcf, 'keyPressed', '');
186 
187     % path
188     handles.path = plot(terrain.path(:,1), terrain.path(:,2), 'k--');
189     handles.trace = plot(NaN, NaN, '-', 'Color', [0.5 0.5 0.5]);
190 
191     % ref snek
192     handles.reflinks = plot(NaN, NaN, '-', 'Color', [0.3 0.3 0.3]);
193     handles.target_dir = plot(NaN, NaN, 'm-');
194     handles.target_spot = scatter(NaN, NaN, 'm>');
195 
196     % snek
197     handles.pos = scatter(NaN, NaN, 'r+');
198     handles.thref_dir = plot(NaN, NaN, 'm--');
199     handles.heading_dir = plot(NaN, NaN, 'g-');
200     handles.phi0_filt_dir = plot(NaN, NaN, 'b-');
201     handles.links = plot(NaN, NaN, 'b-');
202     handles.dots = scatter(NaN, NaN, 6, 'ro');
203 
204     grid minor; axis equal; hold off;
205 
206     for j=1:N_steps
```

```
207
208         % handle eventual closing
209         key = getappdata(gcf, 'keyPressed');
210         if strcmp(key, 'q')
211             disp('<q> pressed. Exiting visualization');
212             break;
213         end
214         setappdata(gcf, 'keyPressed', '');
215
216         % speed regulation
217         tic;
218         speed_mult = speed_mult - dt*(speed_mult - target_speed);
219         k=max(1,min(round(j*speed_mult),N_steps));
220
221         % snek
222         theta = sv.x(k,1:Nl)';
223         heading = sv.heading(k);
224         px = sv.x(k,Nl+1);
225         py = sv.x(k,Nl+2);
226
227         % links COM
228         X = -l*K'*cos(theta) + e*px;
229         Y = -l*K'*sin(theta) + e*py;
230
231         % compute start and end of every link
232         pis = zeros(Nl+1,2);
233         pis(1,:) = [X(1), Y(1)] - l*[cos(theta(1)), sin(theta(1))];
234         for i = 2:Nl+1
235             pis(i,:) = pis(i-1,:) + 2*l*[cos(theta(i-1)), sin(theta(i-1))];
236         end
237
238         % update plots
239         set(handles.trace, 'XData', sv.x(1:k,Nl+1), 'YData', sv.x(1:k,Nl+2));
240         set(handles.links, 'XData', pis(:,1), 'YData', pis(:,2));
241         set(handles.dots, 'XData', X, 'YData', Y);
242         set(handles.pos, 'XData', px, 'YData', py);
243         set(handles.heading_dir, 'XData', [px px + 2*l*Nl*cos(heading)], ...
244                                  'YData', [py py + 2*l*Nl*sin(heading)]);
245         set(handles.thref_dir, 'XData', [px, px + lh_dist*cos(sv.threfs(k,2))], ...
246                                'YData', [py, py + lh_dist*sin(sv.threfs(k,2))]);
247         set(handles.phi0_filt_dir, 'XData', px + 2*l*Nl*cos(heading) + ...
248                                     [0, -0.5*sv.phi0s(k,2)*cos(heading+pi/2)], ...
249                                    'YData', py + 2*l*Nl*sin(heading) + ...
250                                     [0, -0.5*sv.phi0s(k,2)*sin(heading+pi/2)]);
251
252         % ref snek
253         thref = H*[sv.phi_ref(k,:)'; theta(Nl)]; %(2.37)
254         pis = zeros(3*Nl,2);
255
256         % detachs end of previous and begin of next link using NaN
257         for i = 0:Nl-1
258             pis(3*i+1,:) = [X(i+1), Y(i+1)] - l*[cos(thref(i+1)), sin(thref(i+1))];
259             pis(3*i+2,:) = [X(i+1), Y(i+1)] + l*[cos(thref(i+1)), sin(thref(i+1))];
260             pis(3*i+3,:) = [NaN, NaN];
261         end
262
263         % update plot
264         set(handles.reflinks, 'XData', pis(:,1), 'YData', pis(:,2));
265         set(handles.target_dir, 'XData', [px, sv.target(k,1)], ...
266                                 'YData', [py, sv.target(k,2)]);
```

```
267         set(handles.target_spot, 'XData', sv.target(k,1), 'YData', sv.target(k,2));
268 
269         set(handles.title, 'String', sprintf('>>%.2fx, t: %.2f s, ...
270 ␣␣␣␣␣␣␣␣␣␣␣␣c: [%.2f %.2f], chat: [%.2f %.2f], model: %s, contr: %s, ...
271 ␣␣␣␣␣␣␣␣␣␣␣␣heading:%s, int: %s', speed_mult, t(k), ...
272             terrain.c_trues(k,:), sv.c_hat(k,:), func2str(dynamics), ...
273             func2str(controller), func2str(heading_law), func2str(int_method)));
274 
275         pause(dt);
276         if j*dt >= 0.1 && ~speed_corrected % a little to stabilize
277             target_speed = toc/dt;
278             speed_corrected = true;
279         end
280 
281         if k>=N_steps
282             break;
283         end
284     end
285 end
286 
287 % ======================
288 % PLOTS
289 % ======================
290 
291 ...
```